STRUCTURED EVENT REASONING WITH LARGE LANGUAGE MODELS

Li Zhang

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2024

Supervisor of Dissertation

Chris Callison-Burch, Associate Professor, University of Pennsylvania

Graduate Group Chairperson

Mayur Naik, Professor, University of Pennsylvania

Dissertation Committee

Dan Roth (chair), Professor, University of Pennsylvania
Marianna Apidianaki, Senior Researcher, University of Pennsylvania
Mark Yatskar, Assistant Professor, University of Pennsylvania
Rada Mihalcea, Professor, University of Michigan
Graham Neubig, Associate Professor, Carnegie Mellon University

STRUCTURED EVENT REASONING WITH LARGE LANGUAGE MODELS

Dedicated to Professor Dragomir Radev, whose generosity and enthusiasm drive me forward.

ACKNOWLEDGEMENT

My foremost gratitude goes to Chris Callison-Burch, my advisor, who is certainly the primary factor of my success in the PhD program. I am most grateful of the academic freedom Chris gave me so that I could completely unleash my creativity, while he was always there for me in need. To me, this was truly an ideal advisor-student relationship. I learned so much from Chris that I am determined to be a professor like him.

I only decided to pursue a PhD after Rada Mihalcea, my former mentor, saw the potential in me. Rada's positive influence upon me transcends research ability – she makes me a better version of myself. She taught me to be patient and self-content during my early years of urgency and anxiety. Her mindset has always driven me through the most arduous challenges.

I am forever indebted to Dragomir Radev, my first ever mentor in NLP, AI, and research. As Rada would say, Drago sees value in people when few would. I never fully understand why Drago would sit a freshman like me down in his office and teach me basic Linux commands, but I am truly glad he did. I hope I have made him proud.

I never take the unwavering support of my parents, Fan Zhang and Hong Sun, for granted. It is easier said than done to put one's child's happiness and well-being before theirs, and I will always respect them for always being by my side.

I thank my partner Vivienne Chi, the primary source of joy in my life, not only for her companionship over my PhD years, but also for her extraordinary wisdom of living. My time with her is like memories of pure light, which accompanied me through the best and worst times.

I thank Veronica Lyu, Niket Tandon, and Shuyan Zhou for being amazing collaborators that worked side by side with me for multiple projects, as well as lots of inspiring conversations about life, career, and philosophy. They are the reason why my PhD journey is not lonely at all.

I thank Peter Clark, Graham Neubig, Steven Wilson, Peter Jansen, Dan Roth, Rui Zhang, Marianna

Apidianaki, and Mark Yatskar for their invaluable mentorship on different fronts at different stages of my career.

I thank Ziyang Li and Jiani Huang for being my most stalwart friends in the last five years. Playing music, petting cats, lamenting at poor reviewers, debating about nerusymbolic methods, and slurping Terakawa ramen with you - these are just some of my favorite things.

I thank Jeffrey Cho, Hainiu Xu, Joey Hou, and Tianyi Zhang for the opportunity of mentoring them. Seeing their success is often more stimulating than my own.

I thank former and present lab members, Ellie Pavlick, Reno Kriz, Daphne Ippolito, Jie Gao, Aditya Kashyap, Artemis Panagopoulou, Yue Yang, Bryan Li, Alyssa Hwang, Liam Dugan, Ajay Patel, Samar Haider, for their mentorship, collaboration, and friendship.

I thank my collaborators and audience of The Protagonists band and Haz Studio channel for their camaraderie and the joy of music, an unexpected windfall of my PhD journey.

ABSTRACT

STRUCTURED EVENT REASONING WITH LARGE LANGUAGE MODELS

Li Zhang

Chris Callison-Burch

Reasoning about real-life events is a unifying challenge in AI and NLP that has profound utility in a variety of domains, while fallacy in high-stake applications could be catastrophic. Able to work with diverse text in these domains, large language models (LLMs) have proven capable of answering questions and solving problems. However, I show that end-to-end LLMs still systematically fail to reason about complex events, and they lack interpretability due to their black-box nature. To address these issues, I propose three general approaches to use LLMs in conjunction with a structured representation of events. The first is a language-based representation involving relations of sub-events that can be learned by LLMs via fine-tuning. The second is a semi-symbolic representation involving states of entities that can be predicted and leveraged by LLMs via few-shot prompting. The third is a fully symbolic representation that can be predicted by LLMs trained with structured data and be executed by symbolic solvers. On a suite of event reasoning tasks spanning commonsense inference and planning, I show that each approach greatly outperforms end-to-end LLMs with more interpretability. These results suggest manners of synergy between LLMs and structured representations for event reasoning and beyond.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Consider this one-sentence story:

> “At the start of a band practice, the violinist was tuning her instrument, while the drummer started playing an elaborate solo.”

This minimal but eventful narrative showcases that natural language frequently involve the communication of *events*. An event is a semantic concept that describes *something that happens* in the world, in some space and time (e.g., a band practice, tuning a violin, playing a drum solo). We humans possess a deep understanding of these events that transcends their face-value. For instance, we can deduce much implicit information, such as the locale (practice room), participants (musicians), their intents (prepare for a show), preceding events (arrival at practice), involved entities (violin, drumsticks), and so on. While some of the above information might be obvious, obtaining others requires *reasoning*. For example, if one were to ask about the outcome of the above event, in the form of a question:

> “What would the violinist likely say in this situation?”

A human who has never been asked this question before but has some background knowledge can likely answer correctly, for example, by making the following deductions:

1. I know that drum playing is loud.
2. I know that violin tuning requires a quiet environment.
3. Based on 1 and 2, the violinist cannot effectively tune.
4. Therefore, the violinist will be annoyed.

Eventually, they may arrive at a possible answer:

> “Could you wait until I finish tuning?”

In this process, the human is not only drawing upon their knowledge of the world, but also systematically integrating pieces of such knowledge to arrive at the correct answer. Naturally, such an ability is highly sought after in artificial intelligence (AI), machine learning (ML) and natural language processing (NLP) systems.

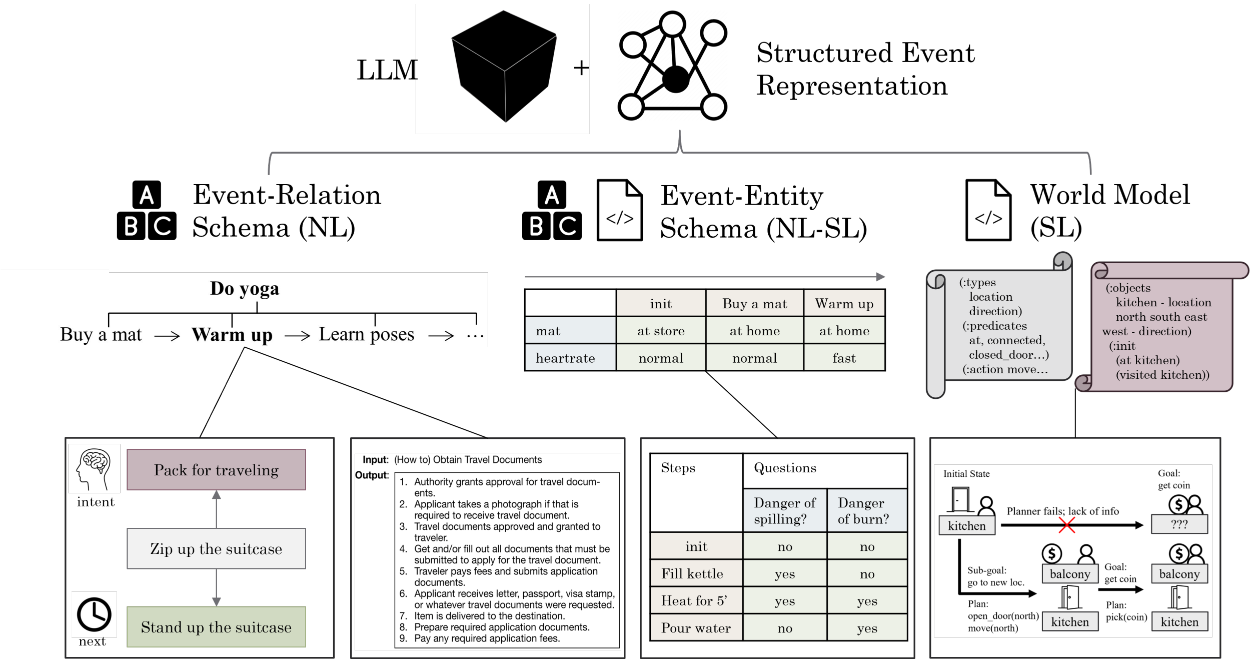
However, if the same question is posed towards ChatGPT[1], its response is:

> “That was an amazing solo! You really know how to rock it!”

an unlikely response from any violinist in this situation. Notwithstanding the rapid advancement of models like ChatGPT trained on massive online data, failures like such are still common when reasoning about events, especially those that are less commonly represented in the training data (i.e., the *long-tail problem*). Apart from the toy example above, the lack of reasoning ability in AI models can be catastrophic in high-stake applications. As technology continues to advance, it is increasingly likely that a user will rely on AI models for legal, financial, or medical advice. In these scenarios, each individual’s case would be significantly different from and more complex than the available training data. If the model fails to generalize and reason correctly, it will suggest unreasonable actions, leading to a loss of freedom, property, or even life. Less drastically, failure to reason like humans do would result in a lack of trust, where users do not feel confident making decisions using AI models.

For decades, the AI, ML, and NLP communities have been honing the ability of intelligent systems to reason over events. Historically, such reasoning was performed using symbolic methods. These are highly interpretable but require domain-specific annotations which can be hard to generalize to new domains, especially to the long-tail problems. In contrast, modern efforts have favored data-driven, neural methods that can effectively adapt to domains that are sufficiently represented in the training data, but still struggle with out of domain or out of distribution situations. Moreover,

[1] A state-of-the-art AI model at the time of writing (chat.openai.com); the experiment was done in December 2023; response may vary due to the model’s non-deterministic nature.



Input: (How to) Obtain Travel Documents

Output:

1. Authority grants approval for travel documents.
2. Applicant takes a photograph if that is required to receive travel document.
3. Travel documents approved and granted to traveler.
4. Get and/or fill out all documents that must be submitted to apply for the travel document.
5. Traveler pays fees and submits application documents.
6. Applicant receives letter, passport, visa stamp, or whatever travel documents were requested.
7. Item is delivered to the destination.
8. Prepare required application documents.
9. Pay any required application fees.

	init	Buy a mat	Warm up
mat	at store	at home	at home
heartrate	normal	normal	fast

Steps	Questions	
	Danger of spilling?	Danger of burn?
init	no	no
Fill kettle	yes	no
Heat for 5'	yes	yes
Pour water	no	yes

```
(:types
  location
  direction)
(:predicates
  at, connected,
  closed_door...)
(:action move...
```

```
(:objects
  kitchen - location
  north south east
west - direction)
 (:init
  (at kitchen)
  (visited kitchen))
```

Next-event prediction & intent detection

Script generation

Causal reasoning of events and entities

Classical planning

Figure 1.1: An overview of work discussed in this thesis.

neural methods provide little in the way of interpretability, nor can their output be easily verified and improved, leading to a lack of trustworthiness.

In my work, I attempt to combine the best of both worlds by reasoning about events using a combination of neural and symbolic methods. I extensively leverage large language models (LLMs), arguably the most powerful tool for most text-based applications. Instead of using LLMs in an end-to-end fashion, I propose a neurosymbolic synergy that combines LLMs with a *structured representation* of events. I explore different forms of such representation as well as different ways to integrate it with LLMs through a variety of downstream tasks (Figure 1.1 and Figure 1.2).

In natural language, an event is often described in relation to other events. Therefore, in Chapter 3, I study a natural language representation by decomposing a complex event into relations of its sub-events. For example, the event "*do yoga*" can be represented as a sequence of its sub-events, "*buy a mat*", "*warm up*", "*learn poses*", and so on. The collection of these sub-steps is essentially a procedure with the goal "*do yoga*", while each can be seen as a step, thus forming a goal-step

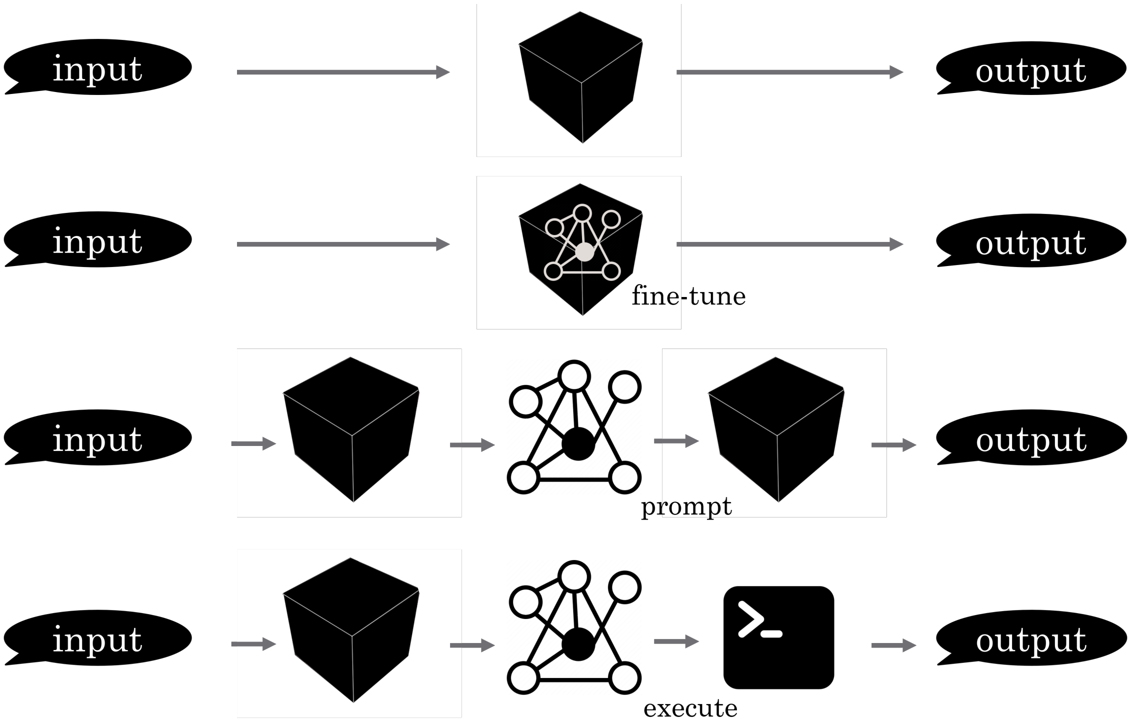


Figure 1.2: Four approaches of using LLMs for structured reasoning: (from upper to lower) end-to-end usage (no structure), fine-tuning with structure, prompting with structure, and neurosymbolic usage.

relation. Additionally, any pair of two steps specifies the order in which they happen, thus forming a step-step temporal relation. These two relations constitute an **event-relation schema**, where each event is still expressed with a natural language phrase, while the relations of them are modeled in a structured fashion.

With these event relations explicitly modeled, I now explore a way to inject such a structured representation into LLMs. In Section 3.1, I fine-tuning LLMs on a dataset targeting the two event relations. Taking the goal-step relation as an example, a model is given a goal "*do yoga*" and must infer the most reasonable step among candidates "A) *buy a suitable mat*", "B) *buy some weights*", "C) *deep clean a mat*", and "D) *buy a house*." While for humans the answer is trivially A, such prediction was non-trivial for LLMs at the time. To train LLMs, I construct a multiple-choice dataset using procedural data from the web. To ensure that the dataset is both clean and challenging, I devise a negative sampling strategy based on LLM-based word vectors to select candidates that are semantically similar but not identical. The resulting dataset is manually verified to have high quality and shows that LLMs are able to learn these event relations, though there is still a performance gap compared to humans.

To ascertain whether the event-relation schema is useful in downstream applications, I apply the previously fine-tuned LLMs on two reasoning tasks in Section 3.2. The first is next-event prediction, a task of commonsense inference. For example, given an event "*the pianist rest her fingers on the keys*", a model must choose the most reasonable subsequent event among "*she started playing*", "*she stood up*", "*she chuckled nervously*", and "*she sang a marvelous tune.*" The second is intent detection, a task that dialog systems like Alexa would need to perform to select the appropriate subroutine. For example, given an utterance "*I want to make restaurant-quality fried rice*", a model must choose the most intent among "*find a recipe*", "*recommend a restaurant*", and so on, before it can call upon an API to fulfill the request. On an array of datasets on both tasks, the models fine-tuned with goal-step and step-step temporal relations, respectively, greatly outperform end-to-end LLMs in few-shot settings. This result demonstrates the benefit of a structured event representation in low-data, long-tail scenarios.

After showing the utility for each individual event relation, I attempt to use both event relations in the event-relation schema at once. In Section 3.3, I tackle the challenging task of script generation. Given a goal such as "*obtain travel documents*", the model must generate all the reasonable steps that are not only pertinent but also correctly ordered, such as "*prepare materials*", "*pay fees*", "*submit application*", and so on. I modularize the task into two sequential stages in a pipeline, each tackled by one of the LLMs fine-tuned with a relevant event relation. By both automatic and manual evaluation of the generated scripts, I again show that structured approach is superior to the end-to-end one on a variety of languages. These experiments collectively take advantage of the flexibility of a natural language representation of events and the specificity of the underlying structure, namely the event relations. Despite the improved performance, a natural language representation that interacts with LLMs via fine-tuning lacks trustworthiness. An end-user would have little idea of how the LLMs produce the output, nor can they effectively improve or correct the models without extensively modifying the data.

Previously, I have chosen to express events using natural language because of their richness and volatility in real-life texts. In comparison, the entities involved in the events are much more finite and grounded. Therefore, in Chapter 4, I switch to a semi-symbolic representation by decomposing a complex event into dynamic state changes of entities. For example, the event "*do yoga*" can be represented as the states of involved entities: the *mat* was *in the store* before the step "*buy a mat*", but *at home* afterwards. The three-dimensional matrix with the axes being the steps, entities, and attributes and the value being the states is referred to as an **event-entity schema**. Although each entity is superficially expressed with a natural language phrase, they are finite and can be grounded in an embodied environment.

How can one equip LLMs with the knowledge of entities, just like that of event relations discussed above? In Section 4.1, I contribute in constructing a dataset of entity state tracking in procedural texts. I facilitate the grounding of entities by canonicalizing them into symbols. Specifically, this entails clustering their mentions. For example, *coffee maker*, *espresso dispenser*, or simply *machine* may refer to the same entity in the procedure of "*make coffee with an espresso machine*." This task is much more challenging than typical coreference resolution or paraphrase detection because procedural texts are highly contextual. By prompting state-of-the-art LLMs with in-context learning, I fully canonicalize an entity tracking dataset that can evaluate entity state tracking models more fairly. These models can thus predict an event-entity schema given a procedure.

Intuitively, the states of entities causally give rise to the occurrence of events. To see if LLMs can similarly leverage the event-entity schema to make predictions about events, in Section 4.2, I define the task of causal reasoning task of events and entities. For example, over the course of "*boiling water*", it will be *dangerous to overturn a kettle* when the kettle is filled and heated up, but not before the kettle is heated or after the water is poured out from the kettle. The above judgement causally hinges on whether the kettle contains hot water. Naturally, I leverage a predicted event-entity schema by providing it to LLMs as an in-context prompt. Notwithstanding the symbolic nature of the schema, I find that a pseudo-code form (one that expresses events and entities as Python functions and class objects) respectively achieves much better performance than a natural language form (e.g., at the end of step "*heat up the kettle*", the *kettle* is *filled with hot water*). In addition to illustrating the potency of the event-entity schema, this observation leaves the open-question of whether such a code-like form better evokes the reasoning ability of modern LLMs trained on a mixture of code and text.

To answer this question, in Section 4.3, I design prompts, both in a typical natural language or textual form and the previously proposed symbolic language or code-like form, on a dozen of general NLP tasks beyond event reasoning. However, despite some contemporaneous work that argues for the idea, I observe no conclusive trend. Nevertheless, regardless of which form to take, the event-entity schema assumes a symbolic nature, and is proven effective when fed to LLMs as in-context input. This mechanism of synergy between the structured representation and LLMs is highly flexible, and provides more interpretability and user control over the fine-tuning mechanism in the last Chapter.

In the two previous Chapters, my proposed structured representation, either in natural or symbolic language, is eventually provided to LLMs as input. However, I have observed that as the reasoning task becomes more complex, LLMs fall short and might not be the optimal tool. In Chapter 5, I introduce an alternative, neurosymbolic methodology by which the structured representation and LLMs interact. Instead of feeding the LLM-generated symbolic representation into another LLM, I rather use an algorithmic solver that can parse and execute it. Because the solver is both deterministic and well-constructed, its output is guaranteed to be correct provided that the input symbolic representation is correct. This way, LLMs are relieved of the task of problem solving, but are only responsible for generating the correct interim representation of the context, or namely a **world model**. By construction, this approach is more trustworthy than the two previous alternatives, in that a user can verify, interpret, and correct the output by interacting with the structured world model. Because the world model is no longer input to LLMs but rather to symbolic solvers, the representation also must be fully symbolic. In other words, the LLM that generates the world model must ground the involved concepts (e.g., events, entities, etc.) to some provided environment such that they can be consumed by the symbolic solver.

An ideal downstream task to instantiate the above idea is classical planning, explored in Section 5.2. At any point of time, a state of the environment is symbolically defined by a collection of entity states (e.g., `in(you, kitchen)`, `closed_door(kitchen,balcony)`, `in(coin,balcony)`. Also defined is a domain model of permitted actions with pre-conditions and effects. Collectively, these two pieces of information constitute a world model. The goal of classical planning is thus to find a sequence of such actions that drive the state of the environment in a given way. In the real life, the environment is rarely defined symbolically, but in natural language (e.g., you are in the kitchen, the kitchen is connected to the balcony by a closed door, a coin is on the balcony, here are a list of actions you can perform...). Given such a description, my approach is not to have LLMs generate a plan, but to generate a representation in planning domain definition language (PDDL) that can be deterministically solved.

Can LLMs indeed generate a symbolic world model given textual description? In Section 5.3, I focus on planning for procedural texts. Given a textual description as above, I use LLMs to generate a domain model before feeding it into a PDDL solver to arrive at the plan. Even though the LLMs' task is reduced to just translating the environment description to a world model but not actual planning, such generation turns out to be highly challenging and even impossible for many LLMs given their weakened ability to generate low-resource, domain-specific languages. Using a combination of approaches to modularize the prompting process, I end up with a model that can generate solvable domain model more than 30% of the time.

In Section 5.4, I shift my focus to planning for interactive textual environments that emulate a robotic application. Here, the environment itself is a symbolically defined state-transition model. Therefore, my approach is to learn a world model through exploration and interaction with the environment. However, unlike the previous setup where the entirety of the environment has been described, many entity states are initially unobserved. Thus, the world model cannot be completely learned, and a plan cannot be derived. To tackle this challenge, I decompose the end goal into sub-goals that *can* be planned for via the current partial world model. Through making progress towards the sub-goals, more of the environments are explored, resulting in a more complete world model, that eventually can support planning for the end goal. I show that this neurosymbolic approach is drastically more effective than having end-to-end LLMs predicting the plan in various interactive environments.

In summary, in this thesis, I introduce three structured event representations: a language-based event-relation schema, a semi-symbolic event-entity schema, and a fully-symbolic world model. I explored ways that these representations may interact with LLMs, including fine-tuning with them, in-context learning with them, and generating them to be solved symbolically. On a variety of downstream tasks of event reasoning, I show that my proposed approaches are superior to end-to-end LLMs, both on performance and trustworthiness.

1.1. Thesis Statement

I argue that using LLMs in conjunction with structured representations of events lead to improved performance in reasoning tasks. This thesis focuses on three types of such structured representations: (1) a language-based event-relation schema, (2) a semi-symbolic event-entity schema, and (3) a fully-symbolic world mode. Through experiments on a variety of approaches to leverage these representations and a diverse set of downstream tasks, I show that the benefit of structured event reasoning using LLMs.

1.2. Publications Presented

The work described in this thesis has been published as the following conference papers in the Association for Computational Linguistics (ACL) Anthology[2]:

(*equal contribution; †mentored student)

1. Zhang, L.*, Lyu, Q.*, and Callison-Burch, C. (2020b). Reasoning about goals, steps, and temporal ordering with WikiHow. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4630–4639, Online. Association for Computational Linguistics.

2. Zhang, L., Lyu, Q., and Callison-Burch, C. (2020a). Intent detection with WikiHow. In Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, pages 328–333, Suzhou, China. Association for Computational Linguistics.

3. Lyu, Q.*, Zhang, L.*, and Callison-Burch, C. (2021). Goal-oriented script construction. In Proceedings of the 14th International Conference on Natural Language Generation, pages 184–200, Aberdeen, Scotland, UK. Association for Computational Linguistics.

4. Zhang, L.*, Xu, H.*†, Yang, Y., Zhou, S., You, W., Arora, M., and Callison-Burch, C. (2023). Causal reasoning of entities and events in procedural texts. In Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia. Association for Computational Linguistics.

5. Zhang, L.*, Dugan, L.*, Xu, H.*†, and Callison-Burch, C. (2023). Exploring the curious case of code prompts. In Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE), Toronto, Canada. Association for Computational Linguistics.

[2] https://aclanthology.org/

6. Zhang, L., Xu†, H., Kommula, A., Zhou, Callison-Burch, C, and Tandon, N. (2024). OpenPI2.0: An Improved Dataset for Entity Tracking in Texts. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, St. Julians, Malta. Association for Computational Linguistics.

7. T. Zhang*†, L. Zhang*, Z. Hou, Z. Wang, Y. Gu, P. Clark, C. Callison-Burch, and N. Tandon. PROC2PDDL: Open-Domain Planning Representations from Texts. In preprint.

8. L. Zhang, P. Jansen, P. Clark, C. Callison-Burch, and N. Tandon. PDDLego: Iterative Planning in Textual Environments. In *SEM 2024.

These publications are licensed under Creative Commons 4.0 BY, listed below. Therefore, parts of the relevant discussions are quoted directly from said publications, with the explicit approval of all co-authors, my thesis committee, and the Graduate Group Chair. None of these publications have been or will be extensively discussed in any of my collaborators' theses. All work was completed jointly with collaborators at the University of Pennsylvania, Carnegie Mellon University, and Allen Institute for Artificial Intelligence. At the end of each chapter, I summarize my primary contribution to the work.

CHAPTER 2

RELATED WORK

Assuming basic knowledge of machine learning, this thesis pertains to three concepts:

1. Events and procedures, the task
2. Large language models, the tool
3. Machine reasoning, the framework

This chapter provides background knowledge to these concepts, focusing on empirical rather theoretical aspects.

2.1. Events and Procedures

2.1.1. Events

Event is a historic and important concept in linguistics and NLP. In this thesis, an event is something that *happens* at some space and time. For example, “play drums”, “jazz concert”, “starts raining”, or “glacier movement” are all natural language expressions that describe events, ubiquitously in texts. Regardless of whether these are noun phrases or verb phrases, the common thread is that all of them can *happen*. Some non-examples include “drummer”, “venue”, “raindrop”, or “Eurasian Plate”. These cannot *happen*, and are in fact **entities** that *participate* in events, which we will study in more details later.

In literature, the concept of event has been thoroughly studied, leading to many possible definitions. In NLP, the closest neighboring field is semantic role labeling (SRL), the task of assigning roles to participants (similar to entities) according to a predicate (similar to events) (Jurafsky and Martin, 2000). The definitions of events in SRL is similar to what I adopt, but I do not focus exclusively on the arguments of events. In theoretical linguistics and formal semantics, the event semantics have a history of decades (Tenny and Pustejovsky, 2000; Maienborn et al., 2011). One of the most acceptable formalism of events is the Davidsonian event semantic (Davidson, 1967), defining events as “concrete particulars with a location in space and time.” Later in this thesis, I will show some work that targets the time-feature of events. The mainstream event formalism was later replaced by the Neo-Davidsonian formalism (Higginbotham, 1985), which expands the definition of events on several fronts (such as not restricting events to verb phrases. Later, this expanded definition also encompasses processes (or procedures) and states, which I will discuss in details (Mourelatos, 1978; Bach, 1986). In this thesis, I do not focus on any particular formalism, but intentionally take on a loose and pragmatic definition with the purpose of solving problems and tackling tasks in NLP.

Events can be multimodal (e.g., “alarm clock ringing” can not only be described, but also be seen and experienced), but I will only discuss them in textual terms under the context of NLP. Because events are a semantic construct, the study of natural language events often falls under the umbrella disciplines of semantics or natural language understanding. As an example of a task that is out of my scope, part-of-speech tagging of the phrase “alarm clock ringing” is not in our scope because it does not pertain to semantics, nor is translating this phrase to French because it does not focus on any particular feature of an event.

My work primarily falls under the sub-field of *event-centric NLP* (Chen et al., 2021b), which includes efforts that span a number of fronts. To start with, information extraction or semantic parsing (Mausam et al., 2012; Liu et al., 2018; Yang et al., 2019a; Du and Cardie, 2020), the task of extracting relational tuples from texts, is largely concerned with events. For example, for the text “a pirate ship was sunken by the coastal guards”, a model might need to identify an *attack* event with the guard and pirate as the *belligerents.* Typically, systems perform information extraction with a given ontology, specifying the information of interest. In case of event extraction, there is often a categorization of events (e.g., material, vehicle, weapon) and parameters associated with each (e.g., agent, beneficiary, manner). In my work, I do not attempt to extract events from texts, but rather assume that they are readily available either via web resources or crowdsourced datasets. Instead, I focus on applying those available event information to downstream applications. However, the formulation of relational tuples aligns with my notion of structured representation, especially those described in Section 3. Notably, unlike typical information retrieval tasks, I do not assume any given ontology with regard to said relational representation.

Similarly, much work has focused on tracking and summarizing events over time (Allan et al., 1998; Laban and Hearst, 2017; Saravanakumar et al., 2021). For example, over the *COVID pandemic* that lasted years, a line of events such as *universities shut-down*, *stay-at-home ordinance*, *dispensing sanitizers*, *masking*, *vaccination* can be collectively summarized as the precautions that were adopted over time. Similar to this line, my work also emphasizes a collection of events bound by temporal relations instead of singular events. In contrast to referenced work that primarily studies events in news, my work focuses on procedures (discussed later), in which events happen in a much smaller scale, but are more intentional and tighter-knit in a more qualified environment. Similar to the discussion on information extraction, this line of work is often concerned with extracting information from unstructured text, while my work is not.

Many efforts branded as *commonsense inference* (Mostafazadeh et al., 2016b; Zellers et al., 2018; Zhou et al., 2020) mainly target event reasoning, the family of tasks that are core to this thesis. These tasks are diverse in nature. For example, the next-event prediction task might provide an event such as *the host taps on the glass in a banquet hall* as input, and a model needs to select the most reasonable subsequent event from *everyone goes silent*, *the guests start singing*, *the crowd bursts into a roar of laughter*, and so on. Because 'commonsense' encompasses additional reasoning beyond the event reasoning that I focus on –including things like social commonsense or physical commonsense–, I refrain from this term. While my exact notion of reasoning will be further defined in Section 2.3, this line of work is concerned with different aspects of events described in the beginning of Chapter 1. Many of the involved datasets are also discussed and used in following sections.

A *script* is a standardized sequence of events about stereotypical activities (Feigenbaum et al., 1981). For example, "*go to a restaurant*" typically involves "*order food*", "*eat*", "*pay the bill*", etc. Such script knowledge has long been proposed as a way to enhance AI systems (Schank, 1977). Most work in script learning focuses on narrative scripts, where declarative or descriptive knowledge is distilled from narrative texts like news or stories (Mujtaba and Mahapatra, 2019b). Such scripts are descriptions of sequential events (e.g. a traffic accident involves a collision, injuries, police intervention, etc.). Chambers and Jurafsky (2008) introduced the classic Narrative Cloze Test, where a model is asked to fill in the blank given a script with one missing event. Following the task, a few papers made extensions on representation (Chambers and Jurafsky, 2009; Pichotta and Mooney, 2014) or modeling (Jans et al., 2012; Pichotta and Mooney, 2016a,b), achieving better performance on Narrative Cloze. Meanwhile, other work re-formalized Narrative Cloze as language modeling (LM) (Rudinger et al., 2015) or multiple-choice (Granroth-Wilding and Clark, 2016) tasks. Alternative to the narrative scripts, procedural scripts are those whose events are unified under a common goal. Those are equivalent to procedures and will be discussed in details later. In Section 3.3, I tackle the task of generating a procedural script.

To summarize, the event-centric NLP targets the special semantics of events, which span various conventional sub-fields in NLP. Zooming into events has much practical impact financially (Ding et al., 2015), scientifically (Berant et al., 2014), and medically (Zhang et al., 2020d). Naturally, it has attracted much attention in the community in the recent years.

2.1.2. Procedures

Most of my thesis work focuses on *procedures*, a concept that may be thought of as a subset of events. A procedure, or a process, is a compound event, (e.g., "learn about NLP"), which can be broken down into multiple events (e.g., "read papers", "take classes", "attend seminars", ...). A procedure consists of a **goal** (or intent, motivation) event, and some **step** events to achieve this goal. Studying procedures has many additional benefits than studying general events (Zhang, 2022), especially from a human-centered perspective, placing an emphasis on humans' behavior and cognition.

The earliest work on procedures in NLP dates back to Miller (1976); Momouchi (1980). At that time, procedural understanding is closely tied to AI *planning* (Schank, 1977). Specifically, a substantial body at that time focused on natural language generation from plans (Mellish and Evans, 1989; Wahlster et al., 1993), many relying on instructional texts as data (Kosseim and Lapalme, 1994; Paris et al., 1995), which are structured and limited in scope. However, such work did not attempt learning from procedural texts, until later when (Paris et al., 2002) created a human-in-the-loop tool for procedural knowledge acquisition. Note that such knowledge extracted from procedural texts can be transformed to plans (the reverse of "natural language generation from plans" mentioned above), which is a substantial body of subsequent research (MacMahon et al., 2006; Branavan et al., 2009; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013; Kiddon et al., 2015). One primary use of such procedural language generation is to answer how-to questions, the second most sought-after type of queries on the internet at that time (de Rijke et al., 2005). To that end, there were an array of efforts to identify **instructional texts** on the web (Takechi et al., 2003), automatically generate them (Paris et al., 2005), converting them to executables (Gil et al., 2011; Fritz and Gil, 2011), study their linguistic idiosyncrasies (Kosseim and Lapalme, 2000; Bielsa and Donnell, 2002; Aouladomar, 2005; Gil, 2015), and extract components such as titles (Delpech and Saint-Dizier, 2008).

Much like information extraction in events discussed above, much work extracts structured knowledge from procedures (Lau et al., 2009; Addis and Borrajo, 2011). Zhang et al. (2012) proposes a standard representation of procedures that emphasizes goals, steps, pre- and post-conditions, much like my formulation in this thesis (Section 3.1 and Section 5.2), but was much simpler and set the basis of procedural representation in subsequent research. (Maeta et al., 2015; Kiddon et al., 2015) took a different approach, and focused on the relations among entities, which are also a center piece in information extraction (Doddington et al., 2004; Ellis et al., 2014). The entity-based representation heavily inspires my work in Chatper 4.

Under the context of script learning, a line of work focuses on procedural scripts, where events happen in a scenario, usually in order to achieve a goal. For example, to “visit a doctor”, one should “make an appointment”, “go to the hospital”, etc. To obtain data, Event Sequence Descriptions (ESD) are collected usually by crowdsourcing, and are cleaned to produce procedural scripts. Thus, most such datasets are small-scale, including OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), the Li et al. (2012) corpus, and DeScript (Wanzare et al., 2016). The evaluation tasks are diverse, ranging from event clustering, event ordering (Regneri et al., 2010), text-script alignment (Ostermann et al., 2017) and next event prediction (Nguyen et al., 2017). There are also efforts on domain extensions (Yagcioglu et al., 2018; Berant et al., 2014) and modeling improvements (Frermann et al., 2014; Modi and Titov, 2014). All of the above are possible candidates for the data source to study procedures.

In our work, the primary source of procedural data is **wikiHow**[3] (previously eHow[4]), a website of how-to instructions for many tasks. As the structure and writing style are consistent, wikiHow has been leveraged by NLP researchers since its inception (Perkowitz et al., 2004; Addis and Borrajo, 2011; Pareti et al., 2014a). In recent years, wikiHow has grown massively in size (now more than 110k articles), diversity (19 languages, hundreds of categories) and quality (editorial process[5]).

[3] wikihow.com

[4] https://www.wikihow.com/wikiHow:History-of-wikiHow

[5] https://www.wikihow.com/wikiHow:Delivering-a-Trustworthy-Experience

As a result, it has fueled much research on procedures from various aspects (Pareti, 2018), such as linking actions (Pareti et al., 2014b; Chernov et al., 2016; Lin et al., 2020a; Donatelli et al., 2021; Zhou et al., 2022), what-if reasoning (Tandon et al., 2019; Rajagopal et al., 2020), entity tracking (Tandon et al., 2020), next-event prediction (Nguyen et al., 2017; Zellers et al., 2019b; Zhang et al., 2020a), intent reasoning (Dalvi et al., 2019), goal-step reasoning (Zhou et al., 2019; Park and Motahari Nezhad, 2018; Zhang et al., 2020c; Yang et al., 2021), procedure generation (Sakaguchi et al., 2021a; Lyu et al., 2021), simulation (Puig et al., 2018), summarization (Koupaee and Wang, 2018; Ladhak et al., 2020) and so on.

As more work has explored procedures (Mujtaba and Mahapatra, 2019a), research agendas become more diverse.

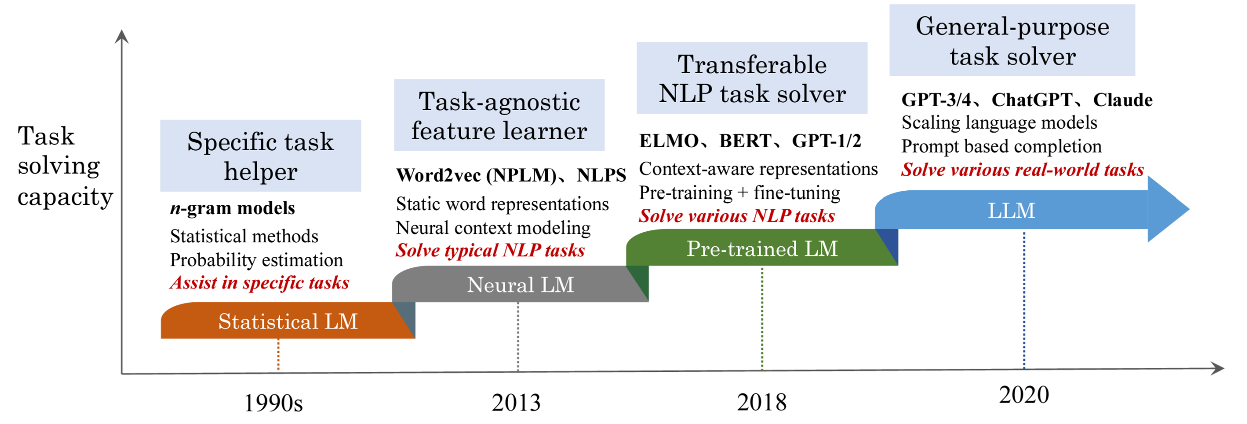


Figure 2.1: The evolution of language models (Zhao et al., 2023).

2.2. Large Language Models

2.2.1. A Brief History of Language Models

For a complete survey on LLMs and their predecessors, readers are redirected to Zhao et al. (2023). In essence, a language model (LM) is one that assigns generative likelihood of vocabulary in a passage. Its revolution over the past decades might be summarized as the following four phases.

Around 1990, **statistical LMs**, or n-gram models, based on the Markov assumption, were proven useful in very specific tasks such as part-of-speech tagging (Thede and Harper, 1999). One typically use these statistical LMs for **probability estimation**: e.g., predicting the tag with the highest probability associated with a token. In these tasks, the probabilistic distribution of the in-domain vocabulary was simple enough to be modeled by the limited order of these statistical LMs. In much more complex tasks, the magnitude of the transition probability becomes intractable.

Around 2013, said limitation was addressed by **neural LMs**. Deep learning approaches such as word2vec (Mikolov et al., 2013) enabled the concept of a distributed representation, condensing information from the context. One typically use these neural LMs for **feature extraction**: e.g., concerting text into a vector representation which is then fed to a model, such as KNN or LSTM. Due to their task-agnostic nature, these neural LMs had gained dominance in many NLP tasks. However, supervised neural models are data-hungry, and thus it is challenging to annotate sufficient data for most tasks, especially the low-resource ones.

Around 2018, this pain point was alleviated by **pre-trained LMs**. Exemplified by ELMO (Peters et al., 2018), BERT (Devlin et al., 2019), and GPT-2 (Radford et al., 2019), these general-purpose models were trained with large-scale unlabeled corpora and could effectively transfer to individual tasks requiring a relatively small amount data. This is referred to as the **pre-train then fine-tune paradigm** which had since become mainstream. With the new-gained power of domain adaptation, pre-trained LMs redefined state-of-the-art for many NLP tasks. Notably, the might of these LMs to preserve knowledge during large-scale pre-training has long been attributed to the Transformer architecture (Vaswani et al., 2017), which has persisted for most offspring NLP models. However, in computer vision, there has been competing arguments that scale plays a more important role than architecture (Smith et al., 2023).

Around 2020, pre-trained LMs (e.g., BERT with 330M parameters) were scaled up to become **large language models** (e.g., GPT-3 with 175B parameters, PaLM (Chowdhery et al., 2023) with 540B parameters). The leap in model size showed emergent abilities (Wei et al., 2022a). The abilities allow for solving complex tasks that had not been possible before (e.g., many of the reasoning tasks will be discussed in this thesis). Also importantly, these LLMs that are pre-trained on a mixture of code and text can generate well-formed symbolic data (e.g., Python, Javascript, JSON), an ability that will be extensively discussed in Section 4.2.5 onward. Moreover, they enable the **in-context learning paradigm**, where previously required fine-tuning data is reduced to just a handful of few-shot learning exemplars, in some cases, zero-shot. In addition to the remarkable performance in many mainstream NLP tasks on par with the data-heavy fine-tuning paradigm, this new paradigm of interacting with large language models is a game-changer for many practitioners, for science and business alike, who have very limited budget for data annotation. In 2023, when I started writing this thesis, products like ChatGPT[6] have become a term as commonplace as ‘iPhone’ for many ordinary people, and a go-to solution provider for many businesses.

It is plain to see that in the past decade alone, LMs have evolved from a probability estimator

[6] https://openai.com/blog/chatgpt

to a problem solver, drifting away from the original perception of "language modeling." Indeed, most task formats in NLP can be reduced to generative language modeling. For example, solving a question answering (QA) example is equivalent to generating the answers given the question as the context. Conversely, generating text using LMs is also equivalent to answering the question "what would be the appropriate response to ..." Hence, the task *formats* often discussed in the NLP community (e.g., QA, dialog, generation, information retrieval, etc.) is not nearly as important as the *core* of the tasks (e.g., event-centric learning, logical reasoning, etc.).

This thesis includes work that spans 2018 to 2024, and thus will primarily focus on the last two types of language models, collectively referred to as LLMs[7]. Due to the highly empirical nature of work discussed in this thesis, it is sufficient to understand the usage of these LLMs (i.e., the two paradigms) without grasping the internal details of these models.

2.2.2. LLMs are Black-Boxes

A black-box is a mechanism that takes in an input, produces an output, while the user cannot interpret what goes on within. Regardless of performance, a black-box gives rise to multiple problems at once:

1. Trustworthiness: a user cannot trust the black-box by knowing that its underlying mechanism aligns with their expectation[8];

2. Verifiability: a user cannot systematically examine if the output is correct;

3. Improvability: a user cannot do things differently for a better performance.

As discussed above, LLMs in the scope of this thesis are neural models, which are long known to be known as black-boxes (Benítez et al., 1997). Notwithstanding decade-long efforts on looking *inside* the black-box (i.e., understanding what neural models do under the hood) (Dayhoff and DeLeo, 2001), I instead take a more macro view and focus on the interpretability of the higher-order

[7] After all, 330 million (BERT's number of parameters) is *large* by most people's standards.

[8] Though a user might still trust a black-box that empirically performs satisfactorily (e.g., by statistical or anecdotal evidence).

pipeline that involves the black-box. In other words, I am interested in *how to use* black-box LLMs so that a user gains more empirical insights into their decision-making process.

The most intuitive and least interpretable approach to use an LLM for most NLP tasks is **end-to-end**. The two “ends” here specify the input and output, agnostic of the task format. For a QA format, the input is the question while the output is the answer; for a dialog format, the input is the conversation history while the output is the utterance; for a translation format, the input is a source passage while the output is a target passage. The end-to-end usage is also agnostic to the two paradigms, pre-train then fine-tune and in-context learning. For the former, both the training and testing data are tuples of input and output; for the latter, both the in-context and to-infer examples are tuples of input and output. With the advancement of large-scale pretraining, the straightforward end-to-end usage demonstrates impressive performance and flexibility, as demonstrated in the big tables of results in mainstream LLM papers (Devlin et al., 2019; Liu et al., 2019; Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). It is arguably the default way to interact LLMs for most. Naturally, the end-to-end usage falls short on the three criteria above. On improvability, specifically, a user’s hands are frustratingly tied. They may either increase the quantity or quality of the fine-tuning data (for the first paradigm), or engineer a different prompt (for the second paradigm), both of which come at an obvious cost.

Later, I will discuss existing attempts to address these issues.

2.3. Machine Reasoning

2.3.1. Reasoning Frameworks

Reasoning is an intriguing and crucial ability of human cognition (Rips, 1990). Naturally, machine reasoning is a well sought-after ability in AI systems to demonstrate human-like intelligence. In the communities of mathematics, AI, machine learning, and NLP, the concept of reasoning is historical and varies greatly (McCarthy, 1959; Pearl, 1988; Duan et al., 2020). For a brief survey on the topic, readers are referred to Duan et al. (2020). For the purpose of this thesis, it is crucial to understand two frameworks for machine reasoning: symbolic reasoning and textual reasoning.

Symbolic reasoning works by manipulating knowledge in the form of symbolic logic using inference algorithms. Such inference algorithm can either be deterministic (Good, Old-Fashioned AI) or probabilistic (Pearl, 1988; Richardson and Domingos, 2006). Either way, the input must be formalized as symbols, which in itself is a non-trivial process for many applications. On the other hand, since the age of deep learning and LLMs, **textual reasoning** that only works with textual input has been made possible. By 2023, hundreds of papers with 'reasoning' in their titles have been published[9] but very few use any symbolic reasoning technique, but primarily use LLMs to answer *reasoning questions* (Dalvi Mishra et al., 2023). These include commonsense reasoning (Davis and Marcus, 2015), numerical reasoning (Cobbe et al., 2021), and of course event-centric reasoning that will be the focus on this thesis.

As discussed before, the default methodology is end-to-end language modeling, which, in conjunction with various techniques, has achieved dominant performance in many of these reasoning tasks.

Comparing the two frameworks for machine reasoning, symbolic reasoning is highly interpretable and may even guarantee correctness given a well-defined symbolic input, much unlike the finicky black-box neural models. Conversely, symbolic reasoning is also highly rigid, requiring extensive training or annotation within a domain, a style, and even a singular dataset, much unlike modern LLMs which can work with rich expressions in diverse domains. Such an impasse of formal vs.

[9] https://aclanthology.org/

neural methods is historical and ongoing.

Neural-symbolic reasoning attempts to combine the best of both worlds and has a long history in literature (Hitzler and Sarker, 2022). Indeed, there are many ways in which one can simultaneously work with symbols and neural networks, including knowledge graphs (Bordes et al., 2013; Teru et al., 2020), neural semantic parsing (Dong and Lapata, 2016; Finegan-Dollak et al., 2018), etc. In this thesis, the notion of *structured reasoning* is a subset of neural-symbolic reasoning, specifically designed to tackle event-centric reasoning. The core idea is to combine LLMs with structured representation of events. We will explore this approach in details later.

2.3.2. Reasoning Tasks

In this thesis, I will focus on reasoning tasks in a broad and empirical sense. In NLP literature, 'reasoning' is a highly overloaded term: what counts as reasoning as what does not is far from clear. What I have in mind is close to the sub-field of **multi-hop reasoning**, either QA from a passage (Welbl et al., 2018; Talmor and Berant, 2018; Yang et al., 2018) or utilizing multihop information in the form of symbolic data (De Cao et al., 2019; Ding et al., 2019; Cao et al., 2019; Fang et al., 2020; Thayaparan et al., 2019). Here, a question has to be answered (or, a problem has to be solved) using at least two pieces of knowledge and one algorithmic operation. However, reasoning processes are subjective. To answer the question "does Ella Fitzgerald typically sing with swing[10]" probably requires just a single hop for jazz lovers, but at least two hops in addition to acquiring missing knowledge for the rest. In this thesis, I take an eclectic view and do not enforce what counts or does not count as reasoning, with much overlap with tasks like inference, entailment, commonsense, and QA.

This thesis focuses on the family of tasks called **event reasoning** that targets either extracting the knowledge or answering the questions described above. In NLP, this work includes includes extracting knowledge from instructional texts (Paris et al., 2002; Delpech and Saint-Dizier, 2008; Zhang et al., 2012), reasoning about relations among events (Takechi et al., 2003; Tandon et al., 2019; Rajagopal et al., 2020), event knowledge-base construction (Jung et al., 2010; Chu et al.,

[10] A rhythmic feel that is iconic in jazz music.

2017; Park and Motahari Nezhad, 2018; Zhou et al., 2022), or various downstream applications (Zhang et al., 2020a; Dalvi et al., 2019; Chen et al., 2020).

2.4. Interplay

In summary, my work discussed in this thesis can be situated in the overlap of the three concepts above: LLMs, events, and reasoning. To tackle event reasoning tasks, I use a conjunction LLMs and structured event representation.

CHAPTER 3

EVENT-RELATION SCHEMA: A NATURAL LANGUAGE REPRESENTATION

Event reasoning is an umbrella term for many tasks and applications. A few event reasoning tasks that I will discuss in this Chapter are: predicting the next event, predicting the intent of an action, and predicting the steps to take in a procedure. In human communication, these tasks almost always involve input and output in the form of natural language. Traditionally, for machines to consume the data and learn the task, it has been imperative to represent events in a symbolic fashion (discussed in Section 2.1). For example, previous work has explored **event schemata** (Baker et al., 1998; Li et al., 2020) or **procedure schemata** (Momouchi, 1980; Zhang et al., 2012; Kiddon et al., 2015) that identify the key components in representing events and procedures. While these schemata are focused and effective for certain tasks, their construction is prone to errors and their ability to generalize to unseen domains is limited. On the other hand, by the time this research project was performed, LLMs have shown superior performance on many of the event reasoning tasks exemplified above. Seemingly, LLMs eliminate the need for any such structured representation, because they can take natural language as input and generate natural language texts directly. Nevertheless, I will show that end-to-end LLMs cannot take advantage of specific knowledge that could be encoded in the schemata, so their performance still has room for improvement.

To push the limit of automated event reasoning, I attempt to bridge the two attempts by defining a language-based structured representation that not only retains a considerable amount of the structured knowledge of events, but also can be consumed and leveraged by LLMs (see Figure 3.1). Previously described symbolic schemata (for example, see Figure 3.2) are not a good fit with LLMs which are trained to exclusively work with natural language. To strike a balance between a symbolic vs. natural language representation, I propose a minimal **relational event schema** that includes two important event relations: hierarchical GOAL-STEP RELATION between an event and its sub-events, and a STEP-STEP TEMPORAL RELATION among the sub-events. The idea is explored under the DARPA KAIROS project[11] and its related work (Li et al., 2021) (see Figure 3.3 for an

[11] https://www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas

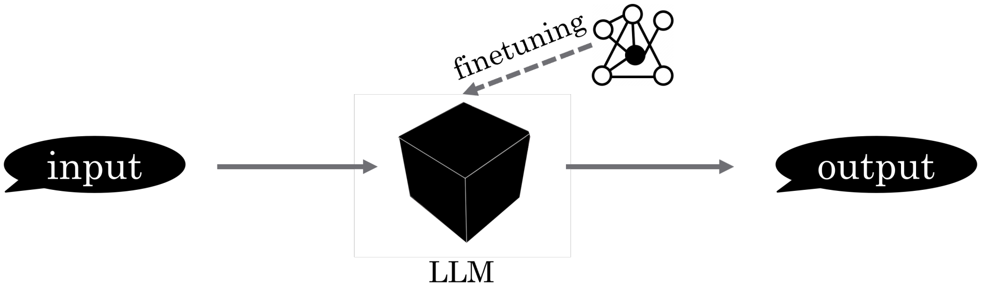
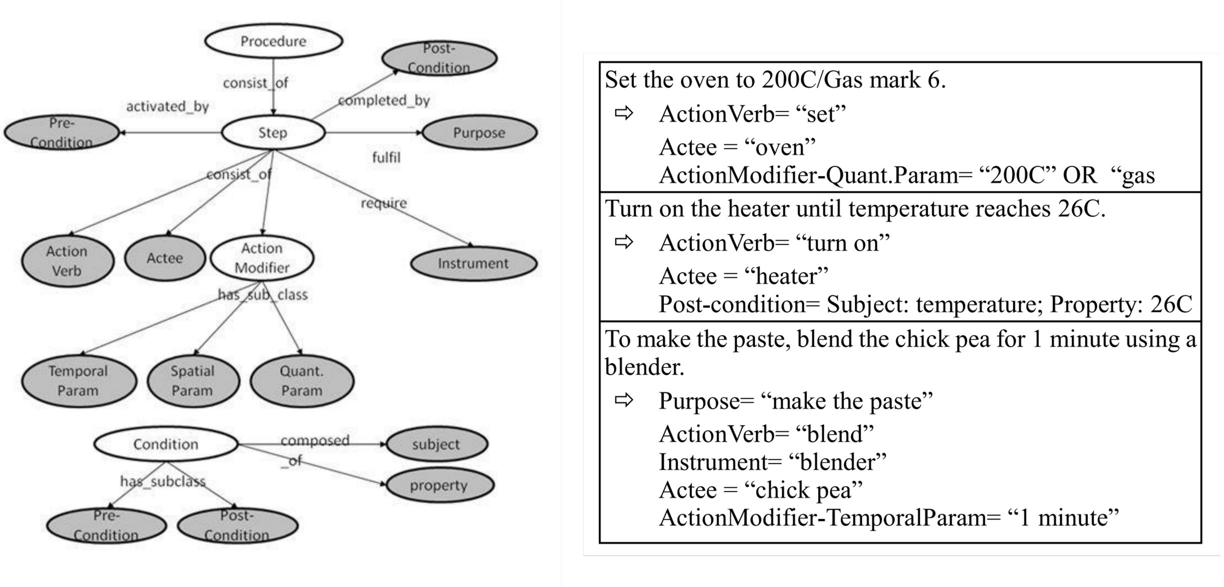


Figure 3.1: An illustration of my proposed pipeline leveraging a natural language representation of entities. The LLM is fine-tuned with data of that representation.



Set the oven to 200C/Gas mark 6.
⇨ ActionVerb= "set"
Actee = "oven"
ActionModifier-Quant.Param= "200C" OR "gas

Turn on the heater until temperature reaches 26C.
⇨ ActionVerb= "turn on"
Actee = "heater"
Post-condition= Subject: temperature; Property: 26C

To make the paste, blend the chick pea for 1 minute using a blender.
⇨ Purpose= "make the paste"
ActionVerb= "blend"
Instrument= "blender"
Actee = "chick pea"
ActionModifier-TemporalParam= "1 minute"

Figure 3.2: A fine-grained procedure representation (schema) proposed by Zhang et al. (2012).

illustration). Compared to existing event schemata such as Figure 3.2, the proposed one has two advantages. First, it is lightweight but versatile, which I will demonstrate through improvements on three downstream tasks later in this Chapter. Second, the operands of each relation (i.e., the atomic unit) are events simply expressed as natural language phrases, which are much more higher-level and flexible than symbolic atomic units. As these natural language event expressions can be consumed and interpreted by LLMs, generalization among domains becomes much more likely.

Next, I describe work done by my collaborators and myself to first learn such a relational event schema data from the web (Section 3.1). We use this data to equip LLMs with such structured

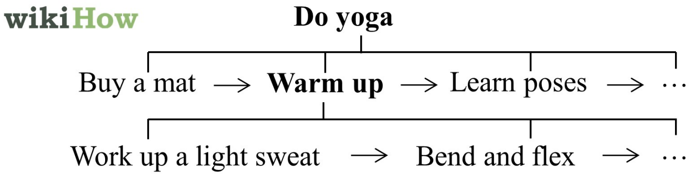


Figure 3.3: Goals and steps (slightly paraphrased) from wikiHow articles “How to Do Yoga” and “How to Warm Up”. The lines denote goal-step relations; the arrows denote step-step temporal relations.

knowledge via fine-tuning and apply them to downstream tasks (Section 3.2 and Section 3.3).

3.1. Learning Event Relations

To construct a large-scale dataset of procedures, we crawl wikiHow[12] which is a how-to website containing more than 110k procedures at the time the work was done in 2020. It has been leveraged by researchers since its inception (Perkowitz et al., 2004; Addis and Borrajo, 2011; Pareti et al., 2014a). Procedural texts are suitable data for inducing hierarchical and temporal relations, because the hierarchical relation exist organically between a **goal** and its **steps** (i.e., GOAL-STEP RELATION), and the TEMPORAL RELATION can be found among the steps. Each wikiHow article represents a procedure and contains a title, which we extract as a goal, and some step paragraphs, of which we extract the headlines as steps (see the top part of Figure 3.3 for an example). Additionally, not used in this work, our data includes related articles, references, Q&A, tips and warnings, links to images, and videos aligned with texts.

wikiHow has articles from a broad range of domains, with 19 top-level categories: Arts and Entertainment, Cars & Other Vehicles, Computers and Electronics, Education and Communications, Family Life, Finance and Business, Food and Entertaining, Health, Hobbies and Crafts, Holidays and Traditions, Home and Garden, Personal Care and Style, Pets and Animals, Philosophy and Religion, Relationships, Sports and Fitness, Travel, Work World, and Youth. We plot the distribution of the top eight categories in Figure 3.4.

Next, our goal is to learn an relational event schema including two relations: hierarchical relation

[12] wikihow.com

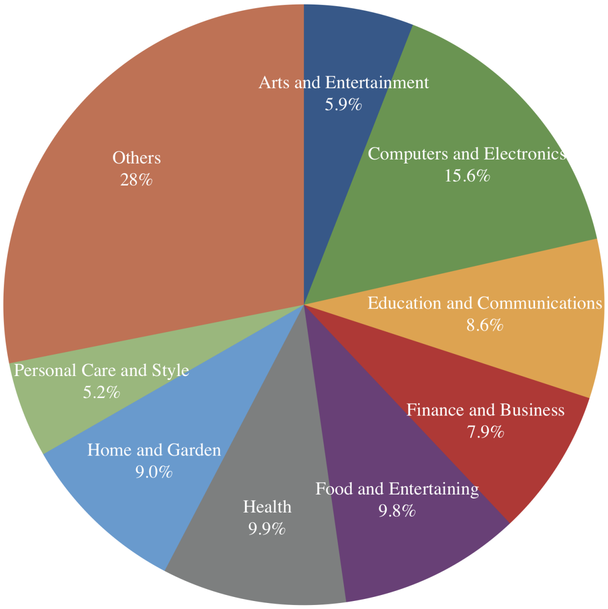


Figure 3.4: Category distribution of wikiHow articles.

between an event and its sub-events, and temporal relation among the sub-events. In procedures, these are equivalent to the GOAL-STEP relation, and the step-step temporal relation. Using the wikiHow corpus, we construct a dataset of three event relation inference tasks.

3.1.1. Step Inference Task

We first introduce the *Step Inference* task, targeting GOAL-STEP relations between events. We formulate this as a 4-choose-1 multiple choice format evaluated using accuracy.

In this task, a system is given a goal and 4 candidate steps and needs to choose the step that helps achieve the goal. For example, given the goal "Prevent Coronavirus" and the candidate steps:

A. wash your hands B. wash your cat

C. clap your hands D. eat your protein

the correct step would be A.

Obtaining the goal and the positive candidate is straightforward, as we sample them iteratively from each how-to article. However, it is challenging to sample negative candidates (Chao et al., 2018; Zellers et al., 2019a) which should have high semantic relatedness with the positive candidate (or the question becomes trivial) while being incorrect answers. We first map each step in wikiHow to a vector representation by taking the average of the BERT embeddings (Devlin et al., 2019) of the verbs. Given the positive step, we then choose 3 steps under different wikiHow categories with the highest cosine similarity to it as the negative candidates. The nearest-neighbors are computed using FAISS (Johnson et al., 2017).

It has recently become clear that the latest NLP models can exploit statistical artifacts from a dataset (Poliak et al., 2018; Si et al., 2019; Zellers et al., 2019b). To prevent the model from learning the negative sampling strategy and relying on just the candidates, we randomly reassign one of the candidates as positive, and the others as negative. Then, we replace the provided goal with the goal attached to the new positive candidate. This strategy ensures that any model performs no better than chance when given access to only the candidates and not the context.

For each step in wikiHow, we create an example by using it as the positive candidate, followed by the negative sampling and label reassignment processes as described above. Then, we apply a collection of hand-crafted filters to remove low-quality examples (Appendix A.1).

3.1.2. Goal Inference Task

Next, we introduce the *Goal Inference* task, formulated in a similar way as Step Inference.

In this task, a system is given a prompt step and 4 candidate goals and needs to choose the correct goal which the step helps achieve. For example, given the step “choose a color of lipstick” and the candidate goals:

A. Get Pink Lips B. Read One’s Lips
C. Lip Sync D. Draw Lips

the correct goal would be A.

For each goal in wikiHow, we create the set of 4 candidates by using it as the positive candidate, followed by the negative sampling, label reassignment, and filtering processes as in Step Inference.

For each positive candidate goal, we use each of its steps to create an example.

3.1.3. Step Ordering Task

Finally, we introduce the *Step Ordering* task, targeting STEP-STEP TEMPORAL relations between events. This task is in a 2-choose-1 multiple choice format evaluated using accuracy.

In this task, given a prompt goal and 2 steps, a system needs to determine which step temporally precedes the other. For example, given the goal "Clean Silver" and the steps:

A. dry the silver B. handwash the silver

the correct answer would be B precedes A.

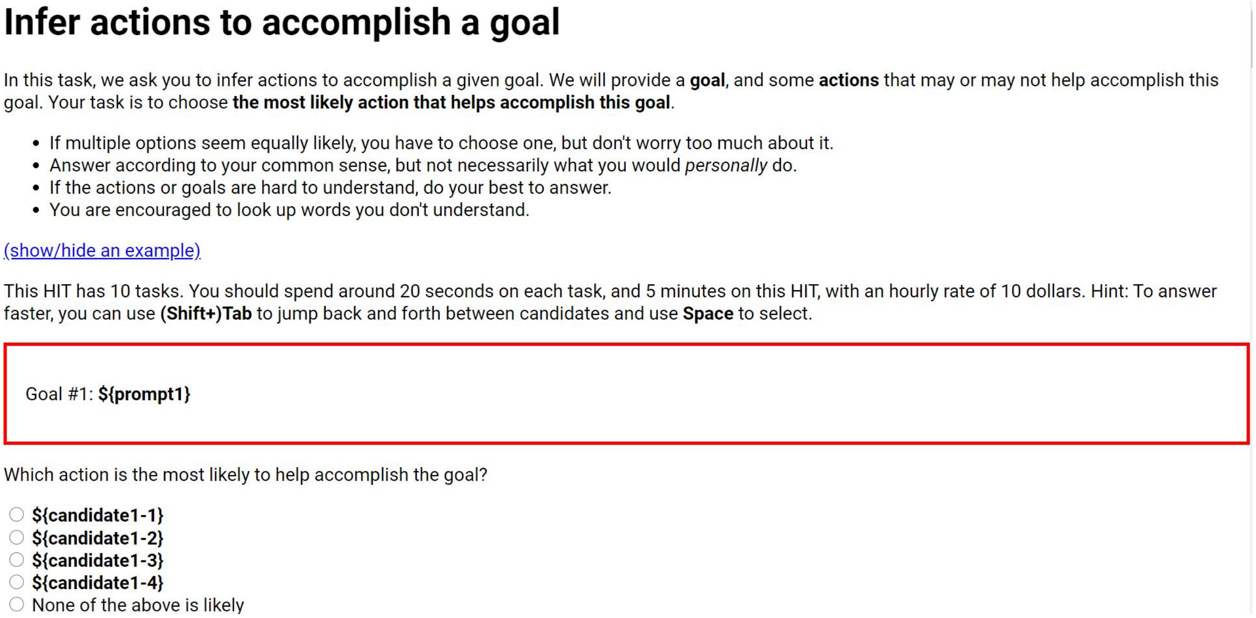
Unfortunately, not all steps in every wikiHow article follow an underlying order. We observe that there are 2 types of wikiHow articles. One is *unordered*, where the steps are parallel alternatives, such as ways to "Stay Healthy" ("develop an exercise routine", "get enough sleep", "eat a healthy diet", etc.). The other is *ordered*, such as recipes for cooking or manuals for fixing appliances, where most steps should be taken sequentially.

We ask 3 annotators to label 1,000 wikiHow articles as ordered or not as a coarse-grained approximation for whether their steps are ordered. We finetune a pre-trained RoBERTa model using 5-fold cross-validation, finding an average precision of 88%. We then ask a 4$^{\text{th}}$ annotator to label another 40 articles as the held-out test set, where the finetuned model achieves 100% precision. Finally, we only consider articles that the model predicts as ordered (around 40%) for the Step Ordering task.

For each goal in wikiHow, we create a set of examples by using it as the prompt and sampling every pair of its adjacent steps as candidates. Then, we randomly shuffle the candidates, so each appears first with 50% chance.

3.1.4. Test Set Construction and Validation

There exists some noise in our automatically generated examples, because some of them do not have a single correct answer. Errors can be introduced when a sampled negative candidate is in



Infer actions to accomplish a goal

In this task, we ask you to infer actions to accomplish a given goal. We will provide a **goal**, and some **actions** that may or may not help accomplish this goal. Your task is to choose **the most likely action that helps accomplish this goal**.

- If multiple options seem equally likely, you have to choose one, but don't worry too much about it.
- Answer according to your common sense, but not necessarily what you would *personally* do.
- If the actions or goals are hard to understand, do your best to answer.
- You are encouraged to look up words you don't understand.

(show/hide an example)

This HIT has 10 tasks. You should spend around 20 seconds on each task, and 5 minutes on this HIT, with an hourly rate of 10 dollars. Hint: To answer faster, you can use **(Shift+)Tab** to jump back and forth between candidates and use **Space** to select.

Goal #1: **${prompt1}**

Which action is the most likely to help accomplish the goal?

○ **${candidate1-1}**
○ **${candidate1-2}**
○ **${candidate1-3}**
○ **${candidate1-4}**
○ None of the above is likely

Figure 3.5: Screenshot of the HIT design for the Step Inference task.

fact correct. For example, in the Goal Inference task, consider an example where the give step is “practice swings”, the expected positive candidate step is “Play Golf”, and a candidate negative example is “Play Drums”. “Play Drums” is sampled due to its high embedding similarity with “Play Golf” and is also a reasonable goal for “practice swings (of the drumsticks)”. This is an ambiguous example and should be excluded from the test set. Therefore, we ask crowd workers to validate a subset of the examples.

We perform crowdsourcing on Amazon Mechanical Turk, requiring Master Qualification and a lifetime HIT approval rate over 90%. The HIT design is shown in Figure 3.5.

For each of Step Inference and Goal Inference, we randomly sample 4,800 examples as input, and for each example we ask 3 crowd workers to choose the most likely candidate. Every HIT includes 15 examples with a pay of $0.83, estimated to be completed in 5 minutes, equivalent to an hourly rate of $9.96.

For Step Ordering, we randomly sample 9,300 examples, and for each example we ask 3 crowd workers to order the events (with a “neutral” option). Every HIT includes 30 examples with a pay

	Step Infer.	Goal Infer.	Step Ordering
Train size	374,278	185,231	836,128
Test size	2,250	1,703	3,100
BERT	.874	.798	.819
XLNet	.867	.783	.826
RoBERTa	.882	.820	.835
GPT-2	.836	.686	.801
Human	.965	.980	.975

Table 3.1: The accuracy of state-of-the-art models on the test sets after being finetuned on our training sets.

of $0.83, estimated to be completed in 5 minutes, equivalent to an hourly rate of $9.96.

In the test set, we only keep examples where all 3 workers agree with the gold label as our benchmark. We remove all examples from the automatically generated ones whose prompt or candidates appear in the test set, and use the remaining data as the training set.

3.1.5. In-Domain Evaluation

To see if LLMs can effectively learn to predict thse two event relations, we finetune pretrained BERT (Devlin et al., 2019), XLNet (Yang et al., 2019b), RoBERTa (Liu et al., 2019) and GPT-2 (Radford et al., 2019) models on the training set and report accuracy on the test set. To benchmark human performance, two authors each annotate 100 random examples from the test set and report the average accuracy. The results are shown in Table 3.1, indicating a satisfactory performance and still a gap of 10% to 20% between human and models trained on all available in-domain data.

At the time this project was carried out, the above models represented the state-of-the-art. Later LLMs with in-context learning abilities would demonstrate an even stronger performance (Srivastava et al., 2022). However, as these larger models are trained on internet data with a later cut-off date than this project, one cannot rule out the possibility that the strong performance is attributed to data contamination (i.e., the models have already seen the test set and labels of our dataset on the web).

3.1.6. Open-Ended Examples

In addition to quantitatively evaluating models on our multiple-choice tasks, we perform qualitative evaluation on some open-ended examples from wikiHow unseen during training, using RoBERTa.[13]

For Step Inference, we rank 100 steps with high embedding similarity for their likelihood of helping achieve a given goal. For example, for the goal "Eat in Islam", the top 3 ranked steps are "understand what type of meats are permissible" (correct), "start by adding mild spices to your food," and "gather supplies and ingredients." Similarly for Goal Inference, we rank 100 goals against some steps. For example, for the steps "spend the holiday with your beloved, eat KFC, check out the light displays," the top 3 ranked goals are "Celebrate a Japanese Christmas" (correct)[14], "Celebrate a Czech Christmas," and "Celebrate a British Christmas." These examples show that the model trained on our data can retrieve texts based on GOAL-STEP relations, beyond simply semantic relevance.

For Step Ordering, the model can perfectly order some articles with as much as 10 steps. For example, given the goal "Clean a Trumpet," the first 5 predicted, ordered steps are "gather your materials," "disassemble your trumpet," "fill up a bathtub," "place a towel down in the tub," and "set your trumpet parts down to soak." This shows that the model trained on our data can order certain long sequences of events based on STEP-STEP TEMPORAL relations.

At this point, we have a suite of LLMs that can discriminate the GOAL-STEP relation and STEP-STEP TEMPORAL relation, given two events. To achieve this, we construct a dataset of procedural knowledge from the web, and fine-tune LLMs on examples that illustrate the above two relations, which collectively constitute the relation event schema I proposed at the beginning of this Chapter. Next, I will show evidence that these LLMs can effectively perform on various event-centric reasoning tasks.

The work above was published in Zhang et al. (2020c), in which my collaborator Qing Lyu and I

[13] More examples are in Appendix A.2.

[14] KFC and light displays are Japanese Christmas traditions (Kimura and Belk, 2005).

contributed equally in roughly all components. I have obtained approval from all collaborators to exclusively include this work in this thesis.

3.2. Application of Event Relation

Using the new-gained ability of LLMs to predict the edges in a graph of the relational event schema described in the beginning of Section 3.1, we will now apply these two relations to two major NLP tasks, intent detection (a component of dialog systems) and next event prediction (a component of commonsense inference). For both cases, if models equipped with the knowledge of relational event schema outperforms those which are end-to-end, we would have shown that our structured representation of events is practically beneficial.

3.2.1. Intent Detection

Task-oriented dialog systems like Apple's Siri, Amazon Alexa, and Google Assistant have become pervasive in smartphones and smart speakers. To support a wide range of functions, dialog systems must be able to map a user's natural language instruction onto the desired skill or API. Performing this mapping is called intent detection.

Intent detection is usually formulated as a sentence classification task. Given an utterance (e.g. "wake me up at 8"), a system needs to predict its intent (e.g. "Set an Alarm"). Most modern approaches use neural networks to jointly model intent detection and slot filling (Xu and Sarikaya, 2013; Liu and Lane, 2016; Goo et al., 2018; Zhang et al., 2019a). In response to a rapidly growing range of services, more attention has been given to zero-shot intent detection (Ferreira et al., 2015a,b; Yazdani and Henderson, 2015; Chen et al., 2016; Kumar et al., 2017). While most existing research on intent detection proposed novel model architectures, few have attempted data augmentation. One such work (Hu et al., 2009) showed that models can learn much knowledge that is important for intent detection from massive online resources such as Wikipedia.

The core idea of our proposed methodology is to transfer the knowledge of GOAL-STEP event relation to the intent detection task, based on the observation that a goal can approximate an intent, and each step in it can approximate an associated utterance. Hence, we reuse the finetuned models described in Section 3.1.5 with some minor differences. To enable multilingual settings, we finetune a pretrained RoBERTa model for the English datasets and a pretrained XLM-RoBERTa model

(Conneau et al., 2020) for the multilingual datasets. In test time, we cast the instances of the intent detection datasets into a multiple-choice format, where the utterance is the input and the full set of intents are the possible candidates, consistent with our wikiHow pretraining task. For each model, we append a linear classification layer with cross-entropy loss to calculate a likelihood for each candidate, and output the candidate with the maximum likelihood.

For each intent detection dataset in any language, we consider the following settings:
+in-domain (+ID): a model is only trained on the dataset's in-domain training data;
+wikiHow +in-domain (+WH+ID): a model is first trained on our wikiHow data in the corresponding language, and then trained on the dataset's in-domain training data;
+wikiHow zero-shot (+WH 0-shot): a model is trained only on our wikiHow data in the corresponding language, and then applied directly to the dataset's evaluation data.

For non-English languages, the corresponding wikiHow data might suffer from smaller sizes and lower quality. Hence, we additionally consider the following cross-lingual transfer settings for non-English datasets:
+en wikiHow +in-domain (+enWH+ID), a model is trained on wikiHow data in English, before it is trained on the dataset's in-domain training data;
+en wikiHow zero-shot (+enWH 0-shot), a model is trained on wikiHow data in English, before it is directly applied to the dataset's evaluation data.

We consider the 3 following benchmarks:
The Snips dataset (Coucke et al., 2018) is a single-turn English dataset. It is one of the most cited dialog benchmarks in recent years, containing utterances collected from the Snips personal voice assistant. While its full training data has 13,784 examples, we find that our models only need its smaller training split consisting of 2,100 examples to achieve high performance. Since Snips does not provide test sets, we use the validation set for testing and the full training set for validation. Snips involves 7 intents, including *Add to Playlist*, *Rate Book*, *Book Restaurant*, *Get Weather*, *Play Music*, *Search Creative Work*, and *Search Screening Event*. Some example utterances include "Play the newest melody on Last Fm by Eddie Vinson," "Find the movie schedule in the area," etc.

	Training Size	Valid. Size	Test Size	Num. Intents
Snips	2,100	700	N/A	7
SGD	163,197	24,320	42,922	4
FB-en	30,521	4,181	8,621	12
FB-es	3,617	1,983	3,043	12
FB-th	2,156	1,235	1,692	12

Table 3.2: Statistics of the dialog benchmark datasets.

The Schema-Guided Dialogue dataset (SGD) (Rastogi et al., 2020b) is a multi-turn English dataset. It is the largest dialog corpus to date spanning dozens of domains and services, used in the DSTC8 challenge (Rastogi et al., 2020a) with dozens of team submissions. Schemas are provided with at most 4 intents per dialog turn. Examples of these intents include *Buy Movie Tickets for a Particular show*, *Make a Reservation with the Therapist*, *Book an Appointment at a Hair Stylist*, *Browse attractions in a given city*, etc. At each turn, we use the last 3 utterances as input. An example: "That sounds fun. What other attractions do you recommend? There is a famous place of worship called Akshardham."

The Facebook multilingual datasets (FB-en/es/th) (Schuster et al., 2019) is a single-turn multilingual dataset. It is the only multilingual dialog dataset to the best of our knowledge, containing utterances annotated with intents and slots in English (en), Spanish (es), and Thai (th). It involves 12 intents, including *Set Reminder*, *Check Sunrise*, *Show Alarms*, *Check Sunset*, *Cancel Reminder*, *Show Reminders*, *Check Time Left on Alarm*, *Modify Alarm*, *Cancel Alarm*, *Find Weather*, *Set Alarm*, and *Snooze Alarm*. Some example utterances are "Is my alarm set for 10 am today?" "Colocar una alarma para mañana a las 3 am," ฉันมีนาฬิกาปลุกสำหรับตอนเช้ามั๊ย etc.

Statistics of the datasets are shown in Table 3.2.

We compare our models with the previous state-of-the-art results of each dataset:

• Ren and Xue (2020) proposed a Siamese neural network with triplet loss, achieving state-of-the-art results on Snips and FB-en;

• Zhang et al. (2019) used multi-task learning to jointly learn intent detection and slot filling,

	Snips	SGD	FB-en
(Ren and Xue, 2020)	.993	N/A	.993
(Ma et al., 2019)	N/A	.948	N/A
+in-domain (+ID)	.990	.942	.993
(ours) +WH+ID	**.994**	**.951**†	**.995**†
(ours) +WH 0-shot	.713	.787	.445
Chance	.143	.250	.083

Table 3.3: The accuracy of intent detection on English datasets using RoBERTa. State-of-the-art performances are in bold; † indicates statistically significant improvement from the previous state-of-the-art.

	FB-en	FB-es	FB-th
(Ren and Xue, 2020)	.993	N/A	N/A
(Zhang et al., 2019)	N/A	.978	.967
+in-domain (+ID)	.993	.986	.962
(ours) +WH+ID	**.995**	.988	.971
(ours) +enWH+ID	**.995**	**.990**†	**.976**†
(ours) +WH 0-shot	.416	.129	.119
(ours) +enWH 0-shot	.416	.288	.124
Chance	.083	.083	.083

Table 3.4: The accuracy of intent detection on multilingual datasets using XLM-RoBERTa.

achieving state-of-the-art results on FB-es and FB-th;

- Ma et al. (2019) augmented the data via back-translation to and from Chinese, achieving state-of-the-art results on SGD.

The modeling details can be found in Appendix A.3.

The performance of RoBERTa on the English datasets (Snips, SGD, and FB-en) are shown in Table 3.3. We repeat each experiment 20 times, report the mean accuracy, and calculate its p-value against the previous state-of-the-art result, using a one-sample and one-tailed t-test with a significance level of 0.05. Our models achieve state-of-the-art results using the available in-domain training data. Moreover, our wikiHow data enables our models to demonstrate strong performances in zero-shot settings with no in-domain training data, implying our models' strong potential to

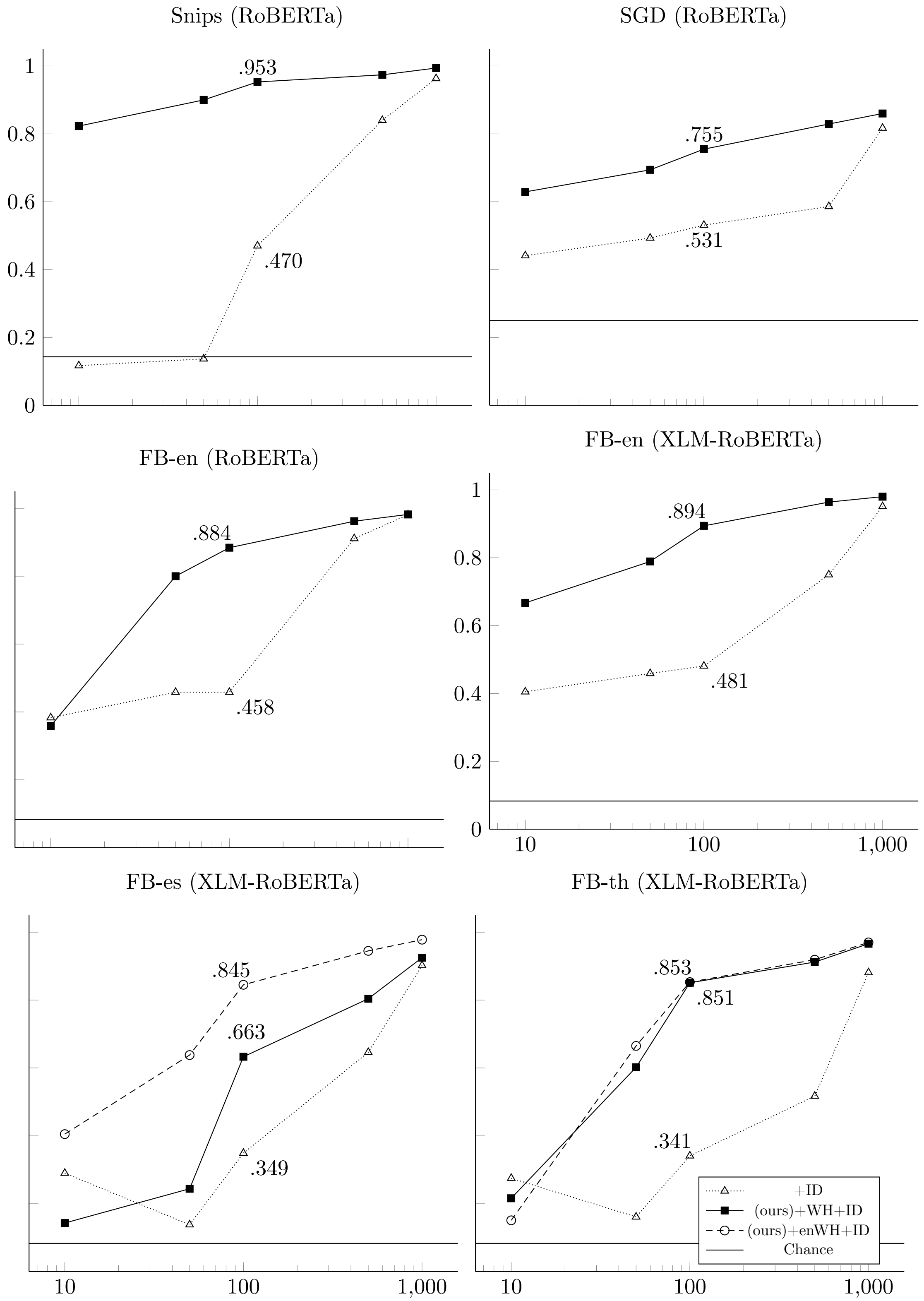


Figure 3.6: Learning curves of models in low-resource settings. The vertical axis is the accuracy of intent detection, while the horizontal axis is the number of in-domain training examples of each task, distorted to log-scale.

adapt to new domains.

The performance of XLM-RoBERTa on the multilingual datasets (FB-en, FB-es, and FB-th) are shown in Table 3.4. Our models achieve state-of-the-art results on all 3 languages. While our wikiHow data in Spanish and Thai does improve models' performances, its effect is less salient than the English wikiHow data.

Our experiments above focus on settings where all available in-domain training data are used. However, modern task-oriented dialog systems must rapidly adapt to burgeoning services (e.g. Alexa Skills) in different languages, where little training data are available. To simulate low-resource settings, we repeat the experiments with exponentially increasing number of training examples up to 1,000. We consider the models trained only on in-domain data (+ID), those first pretrained on our wikiHow data in corresponding languages (+WH+ID), and those first pretrained on our English wikiHow data (+enWH+ID) for FB-es and FB-th.

The learning curves of each dataset are shown in Figure 3.6. Though the vanilla transformers models (+ID) achieve close to state-of-the-art performance with access to the full training data (see Table 3.3 and 3.4), they struggle in the low-resource settings. When given up to 100 in-domain training examples, their accuracies are less than 50% on most datasets. In contrast, our models pretrained on our wikiHow data (+WH+ID) can reach over 75% accuracy given only 100 training examples on all datasets.

As our model performances exceed 99% on Snips and FB-en, the concern arises that these intent detection datasets are "solved". We address this by performing error analysis and providing future outlooks for intent detection.

Our model misclassifies 7 instances in the Snips test set. Among them, 6 utterances include proper nouns on which intent classification is contingent. For example, the utterance "please open Zvooq" assumes the knowledge that Zvooq is a streaming service, and its labelled intent is "Play Music."

Our model misclassifies 43 instances in the FB-en test set. Among them, 10 has incorrect labels: e.g.

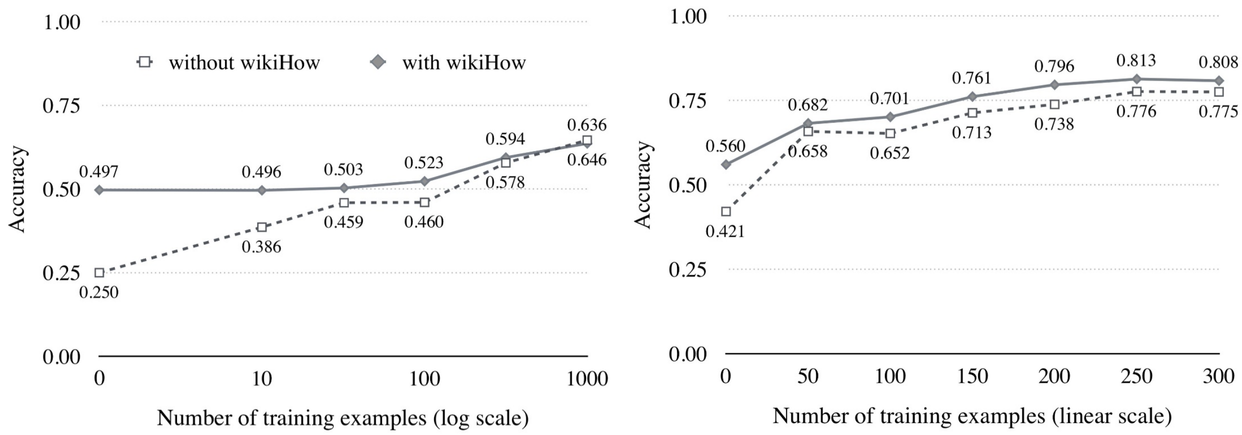
the labelled intent of “have alarm go off at 5 pm” is “Show Alarms,” while our model prediction “Set Alarm” is in fact correct. 28 are ambiguous: e.g. the labelled intent of “repeat alarm every weekday” is “Set Alarm,” whereas that of “add an alarm for 2:45 on every Monday” is “Modify Alarm.” We only find 1 example an interesting edge case: the gold intent of “remind me if there will be a rain forecast tomorrow” is “Find Weather,” while our model incorrectly chooses “Set Reminder.”

By performing manual error analyses on our model predictions, we observe that most misclassified examples involve ambiguous wordings, wrong labels, or obscure proper nouns. Our observations imply that Snips and FB-en might be too easy to effectively evaluate future models.

State-of-the-art models now achieve greater than 99% percent accuracy on standard benchmarks for intent detection. However, intent detection is far from being solved. The standard benchmarks only have a dozen intents, but future dialog systems will need to support many more functions with intents from a wide range of domains. To demonstrate that our pretrained models can adapt to unseen, open-domain intents, we hold out 5,000 steps (as utterances) with their corresponding goals (as intents) from our wikiHow dataset as a proxy of an intent detection dataset with more than 100,000 possible intents (all goals in wikiHow).

For each step, we sample 100 goals with the highest embedding similarity to the correct goal, as most other goals are irrelevant. We then rank them for the likelihood that the step helps achieve them. Our RoBERTa model achieves a mean reciprocal rank of 0.462 and a 36% accuracy of ranking the correct goal first. As a qualitative example, given the step “find the order that you want to cancel,” the top 3 ranked steps are “Cancel an Order on eBay”, “Cancel an Online Order”, “Cancel an Order on Amazon.” This hints that our pretrained models’ can work with a much wider range of intents than those in current benchmarks, and suggests that future intent detection research should focus on open domains, especially those with little data.

In conclusion, by pretraining language models on wikiHow, we attain state-of-the-art results in 5 major intent detection datasets spanning 3 languages. The wide-ranging domains and languages of our pretraining resource enable our models to excel with few labelled examples in multilingual



(a) SWAG

(c) Story Cloze Test

Figure 3.7: Accuracy of RoBERTa on SWAG and Story Cloze Test with different training set sizes, with and without being previously fine-tuned on our data of STEP-STEP TEMPORAL relation.

settings, and suggest open-domain intent detection is now feasible.

The work above was published in Zhang et al. (2020b), in which I primarily contributed to all components. I have obtained approval from all collaborators to exclusively include this work in this thesis.

3.2.2. Next-Event Prediction

In the previous Section, I have demonstrated that the GOAL-STEP event relation can effectively transfer to the task of intent detection. This provides evidence that injecting structured knowledge in LLMs leads to performance gain compared to end-to-end usage. Next, we will see how the STEP-STEP TEMPORAL event relation can transfer to next-event prediction tasks.

We consider two datasets in different domains. **SWAG** (Zellers et al., 2018) is a *commonsense inference* dataset constructed from video captions. Given a context, a system chooses one event most likely to happen from four candidates. For transfer learning, we use up to 1,000 examples for training and the standard validation set. We use the model trained on our Step Inference task to transfer to this task.

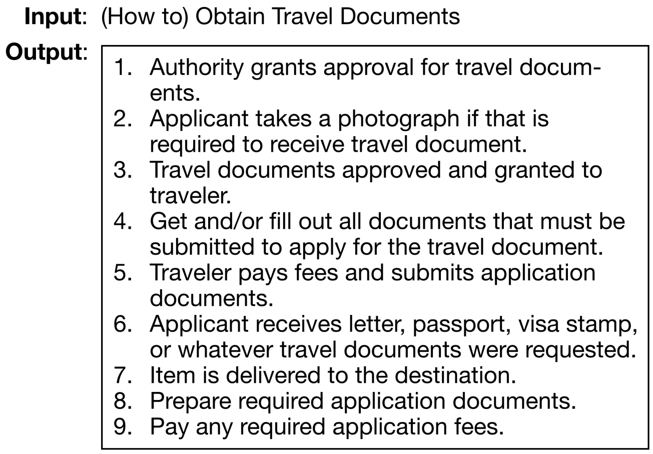
Story Cloze Test (Mostafazadeh et al., 2016a) is a *story understanding* dataset in the fiction domain, where a system chooses an ending to a 4-sentence-story from 2 candidates. We use up to

314 examples for training and 1,571 examples for validation, from the 2016 and 2018 data releases after removing duplicates. We use the model trained on our Step Ordering task to transfer to this task. To mimic the "next sentence prediction" format, we convert each example in our task to a "next step prediction" question with 4 prompt steps and 2 candidate steps, exactly one of which happens after the prompt.

Both datasets come with a sizeable training set, which may easily lead to overfitting. Hence, we only use a subset of their training data to simulate a low-resource scenario. Therefore, we are not comparing to the state-of-the-art performances involving the entire in-domain training sets. For each target task, we finetune a vanilla RoBERTa model and one pretrained on our data described in Section 3.1.5 on increasingly larger portions of the target training set, and observe accuracy on the validation set, as the test set labels are not publicly available.

Figure 3.7 shows the learning curves of the downstream tasks with an increasing number of their training samples, demonstrating a clear advantage of using our training data in low-resource settings. For SWAG, the model trained on our data has a zero-shot performance 24% over chance, outperforming the vanilla model when up to 1,000 training examples are given. For Snips, the model trained on our data boasts an impressive 78% zero-shot performance, approaching perfect accuracy rapidly after some in-domain training. For the Story Cloze Test which has the largest domain-mismatch with our data, the model still benefits from the knowledge learned from it consistently, given any portion of in-domain training data up to the full size in our experiment. These results show that the model learns real-world procedural knowledge from our wikiHow-based tasks, which can be readily applied to various domains and writing styles.

In this section, I have respectively demonstrated the utility of the GOAL-STEP relation and the STEP-STEP TEMPORAL relation on the intent detection task and the next-event prediction task. Both applications illustrate the idea of *structured event reasoning*, as opposed to end-to-end, structure-agnostic LLMs. For example, for intent detection, the key insight to success is that the utterance and intent constitute a portion of the relational event schema. Therefore, models equipped with that structured knowledge are on track to perform better.



Input: (How to) Obtain Travel Documents

Output:

1. Authority grants approval for travel documents.
2. Applicant takes a photograph if that is required to receive travel document.
3. Travel documents approved and granted to traveler.
4. Get and/or fill out all documents that must be submitted to apply for the travel document.
5. Traveler pays fees and submits application documents.
6. Applicant receives letter, passport, visa stamp, or whatever travel documents were requested.
7. Item is delivered to the destination.
8. Prepare required application documents.
9. Pay any required application fees.

Figure 3.8: An example script constructed by our Step-Inference-Ordering pipeline in a zero-shot manner. The input is a *goal*, and the output is an ordered list of steps.

3.3. Application of Event Schema

In the two previous sections, I have shown that LLMs finetuned on the two event relations, collectively constituting the event schema, can outperform end-to-end usages in multiple downstream tasks and datasets. In this section, we will attempt to combine both abilities to tackle the challenging task of procedural **script generation**, generating a sequence of coherent steps given a goal. This process is also equivalent to generating an event schema tree of depth of 1, in which the nodes are the sub-steps or events and the edges are the two relations between each pair of the nodes.

3.3.1. What is Script Learning?

In both lines, it still remains an open problem what kind of automatic task most accurately evaluates a system's understanding of scripts. Most prior work has designed tasks focusing on various fragmented pieces of such understanding. For example, Narrative Cloze assesses a model's knowledge for completing a close-to-finished script. A related concept, Event Sequence Descriptions (ESD), on the other hand, evaluates script learning systems with the aforementioned variety of tasks, each touching upon a specific piece of script knowledge nonetheless (see Section 2.1). Recent work has also brought forth generation-based tasks, but mostly within an open-ended/specialized domain like story or recipe generation (Fan et al., 2018; Xu et al., 2020).

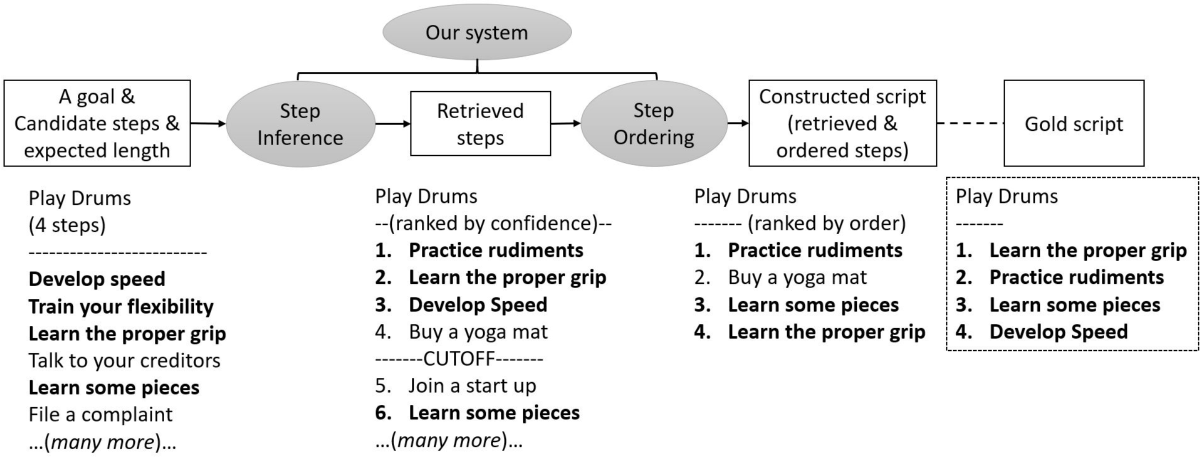


Figure 3.9: Our Step-Inference-Ordering pipeline for the GOSC Retrieval task. An example *ordered* script is shown with example steps in the input and output. Those that appear in the ground-truth script is in bold.

3.3.2. Goal-Oriented Script Construction

We propose the task of *Goal-Oriented Script Construction* (GOSC) to *holistically* evaluate a model's understanding of scripts.Given a *goal* (or the name of a script), we ask the model to construct the sequence of *steps* (or events in a script) to achieve the goal. This task targets a model's ability to narrate an entire script, subsuming most existing evaluation tasks. Our rationale is that a model that *understands* some scripts (e.g. how to "*travel abroad*" and "*go to college*") should be able to produce *new* ones (e.g. how to "*study abroad*") using the absorbed knowledge, close to how humans learn.

Concretely, given a *goal* g, a system constructs a complete script as an ordered list of *steps* S, with a ground-truth reference T. As a hint of the desired level of granularity, we also provide an expected number of steps (or length of the script), l, as input. Depending on whether the set of possible candidate steps are given in advance, GOSC can happen in two settings: Generation or Retrieval.

While almost all prior script learning work has focused on English, we leverage our wikiHow corpus to enable multilingual settings, just as Section 3.2.1.

3.3.3. Models

We develop two systems based on state-of-the-art Transformers for the GOSC task.[15]

Generation Approach For the Generation setting, we finetune mT5 (Xue et al., 2020), a pretrained generation model that is not only state-of-the-art on many tasks but also the only available massively multilingual one to date.

During fine-tuning, we provide the goal of each article in the training set as a prompt, and train the model to generate the sequence of all the steps conditioned on the goal. Therefore, the model's behavior is similar to completing the task of inferring relevant steps and sorting them at once. At inference time, the model generates a list of steps given a goal in the test set.

Retrieval Approach We then implement a *Step-Inference-Ordering pipeline* for the Retrieval setting. Our pipeline contains a Step Inference model to first gather the set of desired steps, and a Step Ordering model to order the steps in the set. These models are based on our previous work described in Section 3.1.5. Under the hood, the models are pretrained XLM-RoBERTa (Conneau et al., 2020) or mBERT (Devlin et al., 2019) for binary classification, both state-of-the-art multilingual representations.

Our Step Inference model takes a goal and a candidate step as input, and outputs whether the candidate is indeed a step toward the goal with a confidence score. During training, for every script, its goal forms a positive example along with each of its steps. We then randomly sample 50 steps from other scripts within the same wikiHow category and pair them with the goal as negative examples. The model predicts a label for each goal-step pair with a cross-entropy loss. During evaluation, for each script in the test set, every candidate step is paired with the given goal as the model input. We then rank all candidate steps based on the model confidence scores decreasingly. Finally, the top l steps are retained, where l is the required length.

Our Step Ordering model takes a goal and two steps as input, and outputs which step happens first. During training, we sample every pair of steps in each ordered script as input to the model with a

[15] Reproducibility details can be found in Appendix A.4.

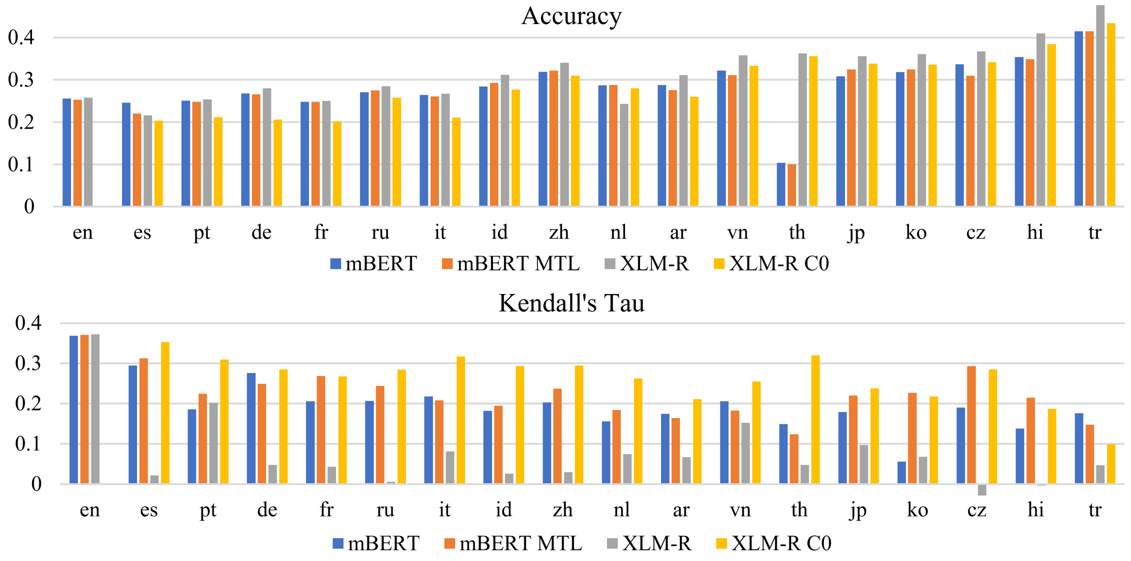


Figure 3.10: Detailed performance on each language from Table 3.6.

cross-entropy loss. During evaluation, we give every pair of retrieved steps as input, and count the total number of times that a step is ranked before others. We then sort all steps by this count to approximate their complete ordering.

An illustration of our Step-Inference-Ordering pipeline is shown in Figure 3.9. We also consider two additional variations.

Multitask Learning (MTL): The Step Inference and the Step Ordering models share the encoder layer, but have separate classifier layers. During training, the MTL system is then presented with a batch of examples from each task in an alternating fashion. During evaluation, the corresponding classifier is used.

Cross-Lingual Zero-Shot Transfer (C0): While there are abundant English training scripts, data in some other languages are scarce. Hence, we also attempt to directly evaluate the English-trained models on non-English data.

3.3.4. In-Domain Evaluation

To demonstrate the performance of models on the GOSC task, we evaluate them on our multilingual wikiHow dataset using both automatic metrics and human judgments. The ultimate utility for this task is the extent to which a human can follow the constructed steps to accomplish the given goal.

Lang.	en	es	pt	de	fr	ru
Perp.	17	11	24	97	46	79
Bert.	.823	.702	.682	.677	.718	.682
Lang.	it	id	zh	nl	ar	vn
Perp.	116	269	13,249	955	746	97
Bert.	.653	.692	.667	.690	.701	.695
Lang.	th	jp	ko	cz	hi	tr
Perp.	29,538	73,952	2,357	1,823	2,033	36,848
Bert.	.701	.679	.692	.682	.704	.665

Table 3.5: Auto evaluation results for the Generation setting (Perplexity and BERTScore F1 measure). The performance of multilingual T5 is reported.

Model	English only		Avg. all lang.s	
	Acc.	Kendall's τ	Acc.	Kendall's τ
mBERT	.256	.369	.286	.198
mBERT MTL	.253	.371	.283	.226
XLM-R	.258	.372	.317	.075
XLM-R C0	-	-	.291	.264

Table 3.6: Auto evaluation results for the Retrieval setting (Accuracy and Kendall's Tau). The performance of mBERT and XLM-RoBERTa, along with their multitask (MTL) and crosslingual zero-shot transfer (C0) variations, are reported. Multitask XLM-R and cross-lingual zero-shot mBERT are found to perform a lot worse and thus omitted.

As direct user studies might be costly and hard to standardize, we carefully choose measures that adhere to this utility. By default, all models are trained and evaluated on the same language.

Auto Evaluation for Generation Setting To automatically evaluate models in the Generation Setting, we report **perplexity** and **BERTScore** (Zhang et al., 2019b), as two frequently used metrics for evaluating text generation.

The mean perplexity of mT5 on the test set of each language is shown in Table 3.5. The results show a large range of variation. To see if perplexity correlates with the data size, we conduct a Spearman's rank correlation two-tailed test. We find a Spearman's ρ of -0.856 and a p-value of $1e-5$ between the perplexity and the number of articles in each language in our dataset; we find a Spearman's ρ of -0.669 and a p-value of $2e-4$ between the perplexity and the number of tokens in each language in the mC4 corpus where mT5 is pretrained on. These statistics suggest a significant correlation between perplexity and data size, while other typological factors are open to investigation.

Table 3.5 also shows the BERTScore F1 measure of the generated scripts compared against the gold scripts. Except for English (.82), the performance across different languages varies within a relatively small margin (.65 - .72). However, we notice that as a metric based on the token-level pairwise similarity, BERTScore may not be the most suitable metric to evaluate scripts. It is best designed for *aligned* texts (e.g. a machine-translated sentence and a human-translated one), whereas in scripts, certain candidate steps might not have aligned reference steps. Moreover, BERTScore does not measure whether the ordering among steps is correct. To address these flaws, we further perform human evaluation later.

Auto Evaluation for Retrieval Setting To automatically evaluate models in the Retrieval Setting, we first calculate accuracy, i.e. the percentage of predicted steps that exist in the ground-truth steps. To account for the ordering of steps, we also compute Kendall's τ between the overlapping steps in the prediction and the ground-truth.

The performance of our Step Inference-Ordering pipeline using mBERT and XLM-RoBERTa[16] on all 18 languages are shown in Figure 3.10. Across languages, the results are generally similar with a large room for improvement. On average, our best system constructs scripts with around 30% accuracy and around 0.2 Kendall's τ compared to the ground-truth. Compared to the baseline, our multitask and cross-lingual zero-shot variations demonstrate significant improvement on ordering. This is especially notable in low-resource languages. For example, MTL on Korean and C0 on Thai both outperform their baseline by 0.17 on Kendall's τ.

Human Evaluation To complement automatic evaluation, we ask 6 annotators[17] to each *edit* 30 output scripts by the Step-Inference-Ordering pipeline and mT5 in English, French, Chinese, Japanese, Korean and Hindi, respectively. The *edit* process consists of a sequence of two possible

[16] XLM-RoBERTa is not able to converge on the training data for Step Ordering for all but 3 languages using a large set of hyperparameter combinations.

[17] The annotators are graduate students and native or proficient speakers of the language assigned.

Retrieval: Step-Inference-Ordering pipeline						
Language	en	fr	zh	jp	ko	hi
Correctness	.70	.39	.50	.49	.45	.82
Completeness	.70	.39	.50	.49	.45	.82
Orderliness	.45	.38	.16	.12	.10	.75
Generation: mT5						
Language	en	fr	zh	jp	ko	hi
Correctness	.39	.51	.46	.40	.37	.49
Completeness	.35	.40	.46	.30	.36	.41
Orderliness	.82	.46	.60	.81	.69	.88

Table 3.7: Human judgments of correctness, completeness and orderliness of the output of the Step-Inference-Order pipeline and the mT5 model for the same set of 30 gold scripts, in six languages.

actions: either 1) delete a generated step entirely if it is irrelevant, nonsensical or not a reasonable step of the given goal, or 2) move a step somewhere else, if the order is incorrect. Then, the generated script is evaluated against the edited script in 3 aspects:

Correctness, approximated by the length (number of steps) of the edited script over that of the originally constructed script (c.f. precision);

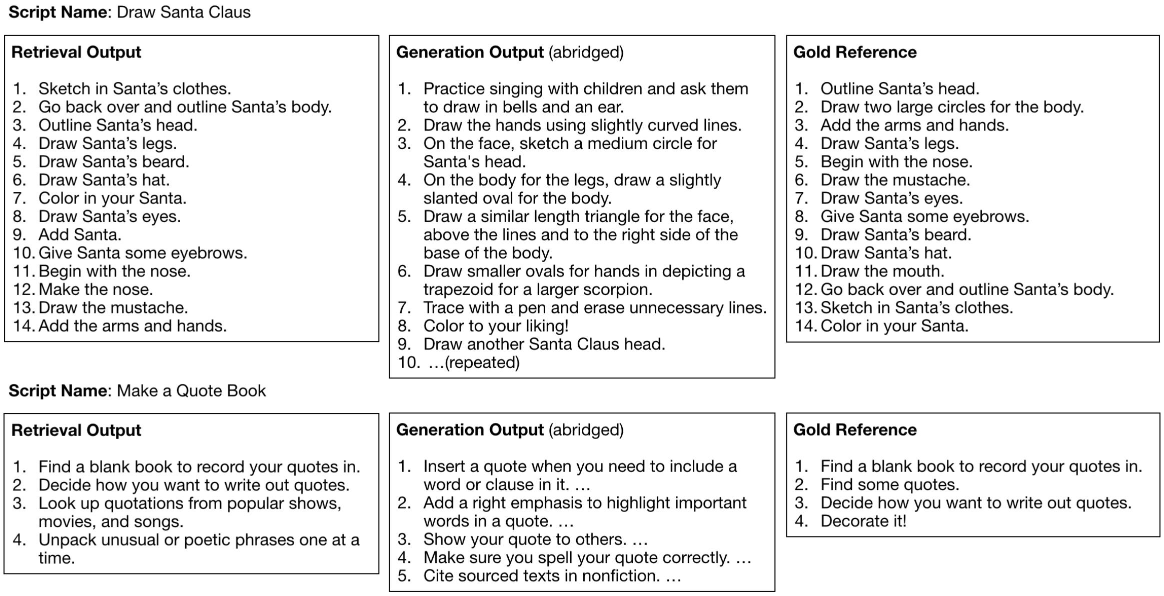
Completeness, approximated by the length of the edited script over that of the ground-truth script (c.f. recall);

Orderliness, approximated by Kendall's τ between overlapping steps in the edited script and the generated script.[18]

The results are shown in Table 3.7. While the constructed scripts in the Retrieval setting contain more correct steps, their ordering is significantly worse than those in the Generation setting. This suggests that the generation model is better at producing *fluent* texts, but can easily suffer from hallucination.

Qualitative Examples To understand models' behavior, we present two representative scripts produced by the mBERT Retrieval model and the mT5 Generation model side by side, accompanied by the ground-truth script, shown in Figure 3.11.

[18] In this formulation, the correctness and completeness of a retrieval-based model are equal, since the length of its constructed script is equal to that of the ground truth script by definition.



Script Name: Draw Santa Claus

Retrieval Output

1. Sketch in Santa's clothes.
2. Go back over and outline Santa's body.
3. Outline Santa's head.
4. Draw Santa's legs.
5. Draw Santa's beard.
6. Draw Santa's hat.
7. Color in your Santa.
8. Draw Santa's eyes.
9. Add Santa.
10. Give Santa some eyebrows.
11. Begin with the nose.
12. Make the nose.
13. Draw the mustache.
14. Add the arms and hands.

Generation Output (abridged)

1. Practice singing with children and ask them to draw in bells and an ear.
2. Draw the hands using slightly curved lines.
3. On the face, sketch a medium circle for Santa's head.
4. On the body for the legs, draw a slightly slanted oval for the body.
5. Draw a similar length triangle for the face, above the lines and to the right side of the base of the body.
6. Draw smaller ovals for hands in depicting a trapezoid for a larger scorpion.
7. Trace with a pen and erase unnecessary lines.
8. Color to your liking!
9. Draw another Santa Claus head.
10. …(repeated)

Gold Reference

1. Outline Santa's head.
2. Draw two large circles for the body.
3. Add the arms and hands.
4. Draw Santa's legs.
5. Begin with the nose.
6. Draw the mustache.
7. Draw Santa's eyes.
8. Give Santa some eyebrows.
9. Draw Santa's beard.
10. Draw Santa's hat.
11. Draw the mouth.
12. Go back over and outline Santa's body.
13. Sketch in Santa's clothes.
14. Color in your Santa.

Script Name: Make a Quote Book

Retrieval Output

1. Find a blank book to record your quotes in.
2. Decide how you want to write out quotes.
3. Look up quotations from popular shows, movies, and songs.
4. Unpack unusual or poetic phrases one at a time.

Generation Output (abridged)

1. Insert a quote when you need to include a word or clause in it. …
2. Add a right emphasis to highlight important words in a quote. …
3. Show your quote to others. …
4. Make sure you spell your quote correctly. …
5. Cite sourced texts in nonfiction. …

Gold Reference

1. Find a blank book to record your quotes in.
2. Find some quotes.
3. Decide how you want to write out quotes.
4. Decorate it!

Figure 3.11: Two example scripts constructed by our Retrieval and Generation approaches.

The retrieved "Draw Santa Claus" script has a high step accuracy (85%), with a reasonable ordering of drawing first the outline and then details. The generation output is more off-track, hallucinating irrelevant details like "singing" and "scorpion", despite being on the general topic of drawing. It also generates more repetitive steps (e.g. the head is drawn twice), most of which are abridged.

As for "Make a Quotebook", the retrieved script has a 50% step accuracy. The third step, though not in the gold reference, is similar enough to "find some quotes", suggesting that our exact match evaluation isn't perfect. In the generated script, all steps are also generally plausible, but some essential steps are missing (e.g. find a book, find quotes). This suggests that the generation model dwells too much on the details, ignoring the big picture.

These patterns in the two scripts are common in the model outputs, a larger sample of which is included in the Supplementary Materials.

3.3.5. Out-Domain Evaluation

To show the potential of our model for transfer learning, we use the retrieval-based Step-Inference-Ordering pipeline finetuned on wikiHow to construct scripts for other datasets and domains. We quantitatively evaluate our model on 4 other script learning corpora, and qualitatively analyze some

Corpus	Corpus Stats.		Results	
	Scenarios	ESDs	Acc.	Kendall's τ
SMILE	22	386	.435	.391
OMICS	175	9044	.346	.443
DeScript	40	4000	.414	.418
KAIROS	28	28	.589	.381

Table 3.8: The zero-shot GOSC Retrieval performance of XLM-RoBERTa finetuned on wikiHow on 4 target corpora.

constructed scripts in a case study.

Quantitative Evaluation Since no multilingual script data are available yet, we perform transfer learning experiments on 4 other English script corpora, OMICS (Singh et al., 2002), SMILE (Regneri et al., 2010), DeScript (Wanzare et al., 2016)[19], and the KAIROS Schema Learning Corpus (LDC2020E25). The first 3 pertain to human activities, while the last is in the military and political domain. They are all in the format of different *scenarios* (e.g. "eat in a restaurant", similar to our *goal*) each with a number of *event sequence descriptions* (ESDs, similar to our *steps*). Statistics for each corpus are in Table 3.8.

For each dataset, we select the ESD with the most steps for every scenario as a representative script to avoid duplication, thus converting the dataset to a GOSC evaluation set under the Retrieval setting. We then use the XLM-RoBERTa-based Step-Inference-Ordering pipeline trained on our English wikiHow dataset to directly construct scripts on each target set, and report its zero-shot performance in Table 3.8. We see that $30\% - 60\%$ steps are accurately retrieved, and around 40% are correctly ordered. This is close to or even better than the in-domain results on our English test set. As a comparison, a random baseline would have only 0.013 Accuracy and 0.004 τ on average. Both facts indicate that the script knowledge learned from our dataset is clearly non-trivial.

KAIROS Case Study: The *Bombing Attack* Scripts To explore if the knowledge about *procedural* scripts learned from our data can also facilitate the zero-shot learning of *narrative* scripts, we present a case study in the context of the DARPA KAIROS program[20]. One objective of KAIROS is to automatically induce scripts from large-scale narrative texts, especially in the military and

[19] The above 3 corpora are all obtained from http://www.coli.uni-saarland.de/projects/smile/

[20] `www.darpa.mil/program/knowledge- directed-artificial-intelligence-reasoning -over-schemas`

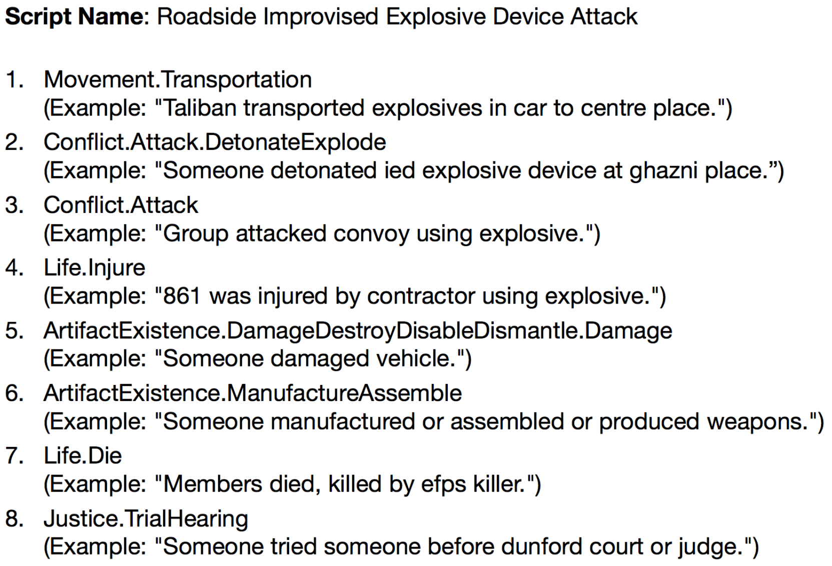
political domain. We show that models trained on our data of commonplace events can effectively transfer to vastly different domains.

With the retrieval-based script construction model finetuned on wikiHow, we construct five scripts with different granularity levels under the *Improvised Explosive Device (IED) attack* scenario: "Roadside IED attack", "Backpack IED attack", "Drone-brone IED attack", "Car bombing IED attack", "IED attack". We take the name of each script as the input *goal*, and a collection of related documents retrieved from Wikipedia and Voice of America news as data sources for extracting step candidates.

Our script construction approach has two components. First, we extract all events according to the KAIROS Event Ontology from the documents using OneIE (Lin et al., 2020b). The ontology defines 68 event primitives, each represented by an *event type* and multiple *argument types*, e.g. a Damage-type event has arguments including Damager, Artifact, Place, etc. OneIE extracts all event instances of the predefined primitives from our source documents. Each event instance contains a *trigger* and several *arguments* (e.g. Trigger: "destroy", Damager: "a bomber", Artifact: "the building", ...). All event instances form the candidate pool of steps for our target script.

Since the events are represented as trigger-arguments tuples, a conversion to the raw textual form is needed before inputting them into our model. This is done by automatically instantiating the corresponding event type template in the ontology with the extracted arguments. If an argument is present in the extracted instance, we directly fill it in the template; else, we fill in a placeholder word (e.g."some", "someone", depending on the argument type). For example, the template of Damage-type events is "$\langle arg1 \rangle$ damaged $\langle arg2 \rangle$ using $\langle arg3 \rangle$ instrument", which can be instantiated as "A bomber damaged the building using some instrument"). Next, we run the Step Inference-Ordering Pipeline described before on the candidate pool given the "goal". The only modification is that since we don't have a gold reference script length in this case, all retrieved steps with a confidence score higher than a threshold (default=0.95) are retained in the final script.

We manually evaluate the constructed scripts with the metrics defined in Section 3.3.4, except *Com-*



Script Name: Roadside Improvised Explosive Device Attack

1. Movement.Transportation
   (Example: "Taliban transported explosives in car to centre place.")
2. Conflict.Attack.DetonateExplode
   (Example: "Someone detonated ied explosive device at ghazni place.”)
3. Conflict.Attack
   (Example: "Group attacked convoy using explosive.")
4. Life.Injure
   (Example: "861 was injured by contractor using explosive.")
5. ArtifactExistence.DamageDestroyDisableDismantle.Damage
   (Example: "Someone damaged vehicle.")
6. ArtifactExistence.ManufactureAssemble
   (Example: "Someone manufactured or assembled or produced weapons.")
7. Life.Die
   (Example: "Members died, killed by efps killer.")
8. Justice.TrialHearing
   (Example: "Someone tried someone before dunford court or judge.")

Figure 3.12: An example narrative script produced by our retrieval-based pipeline trained on wikiHow. Each event is represented by its Event Type and an example sentence.

pleteness as we don't have gold references. The 5 constructed scripts have an average *Correctness* of 0.735 and *Orderliness* of 0.404. Despite the drastic domain shift from wikiHow to KAIROS, our model can still exploit its script knowledge to construct scripts decently. An example script, "Roadside IED attack", is shown in Figure 3.12. All the steps retrieved are sensible, and most are ordered with a few exceptions (e.g. the ManufactureAssemble event should precede all others).[21]

The work above was published in Lyu et al. (2021), in which my collaborator Qing Lyu and I contributed equally in roughly all components. I have obtained approval from all collaborators to exclusively include this work in this thesis.

3.4. Summary

Until this point, I have defined a relational event schema using two event relations. Recall that in the very beginning of this thesis, I outlined the two desirable metrics of any method: performance of trustworthiness. By fine-tuning pre-trained LLMs on a natural language representation of these relations, I have shown an increase of performance over end-to-end usage. For end-users, this methodology also brings about an increased sense of trustworthiness. For example, to tackle an

[21] More details on the format of the script, all five constructed scripts, the event ontology, and a list of news documents used can be found in the Supplementary Materials.

intent detection task, a user using an end-to-end LLM would not know why or how the model succeeds or fails, whereas a user using an LLM finetuned on the GOAL-STEP event relation at least knows that the model is equipped with a particular piece of knowledge helpful to tackle the current task. For the script generation task, our proposed pipeline is much more transparent (c.f., neural module networks described in Section 2.3.1), as the model outputs of the two stages (step retrieval and step ordering) are mechanically combined as the final output. Such is a case of structured reasoning argued in this thesis. For a variety of event-centric tasks, one may represent an event as our proposed schema, and use either its components of itself as a whole to solve various problems.

However, there are still two fundamental issues. First, the atomic unit of the current event schema is still events (sub-events). As a result, reasoning tasks that involve more fine-grained information cannot be tackled. This calls for a more general representation of events. Second, our approach of fine-tuning LLMs is reliant on a sizeable amount of training data. Moreover, it is challenging for end-users to understand how an LLM generates an output based on a large amount of training data. Both issues lead to a lack of user-control, or the ability to improve and trust the model. In the next chapter, I attempt to tackle both issues.

CHAPTER 4

EVENT-ENTITY SCHEMA: A SEMI-SYMBOLIC REPRESENTATION

Imagine teaching a kid or a robot how to cook. Midway through the process, you are asked the following question.

Is it safe to touch (the center of) the pan?

Essentially, this question calls for reasoning about the event "*touch the pan*" given certain circumstances. As discussed before, one may naively pose this question directly to an end-to-end LLM, a baseline approach that I have shown to have many shortcomings. As before, a reasonable alternative may be using a structured event representation. Unfortunately, the relational event schema described in the last chapter would not work here, because the event "*touch the pan*" is neither a goal nor a step. Fundamentally, the answer to this question does not depend on the relation among its sub-events or any other events. To answer this question, most humans would realize a connection to an **entity state**:

Is it safe to touch the pan only if the pan is cool.

The above statement describes a logic statement (i.e., inference) that can be written semi-formally as:

$$\texttt{entity\_has\_attribute}(pan, cool) \Rightarrow \texttt{event\_can\_happen}(safe\ to\ touch\ the\ pan) \tag{4.1}$$

The above formula indicates two items that must be inferred: 1. whether the entity *pan* is *hot*, and 2. the causality between the state of an entity and the plausibility of an event. In this particular instance, the former can be inferred from the context, while the latter is commonsense. In real life involving complex scenarios, inferring either of the two items is highly challenging, and therefore LLMs are reasonably good tools to do so. Instead of using LLMs to directly answer the question, we have now decomposed the task and use LLMs in a modular fashion.

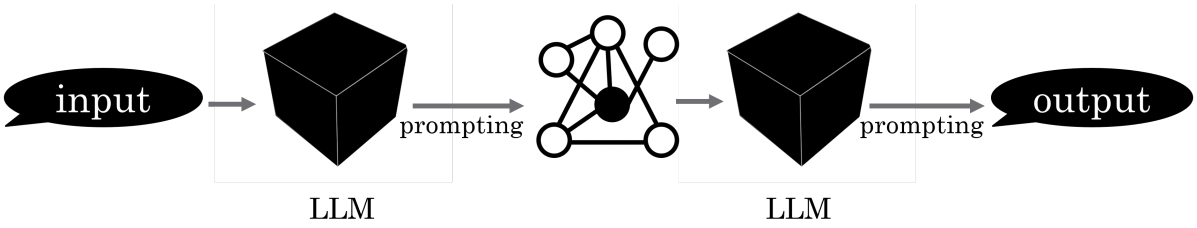
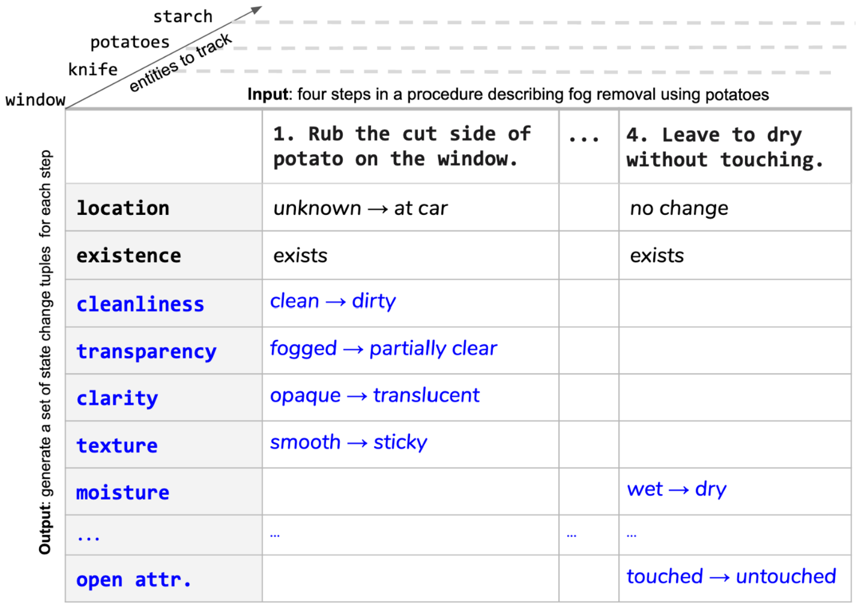


Figure 4.1: An illustration of my proposed pipeline leveraging a semi-symbolic representation of entities. The LLMs are interacted via few-shot prompting.

The thought process above demonstrates the neurosymbolic methodology. The neural part naturally refers to the LLMs whose input is free-formed natural language. The symbolic part refers to how we decompose the task, grounding the concepts of *pan*, *cool*, `entity_has_attribute`, and even the inference operation ⇒ into symbols. Therefore, answering the question "*is it safe to touch the pan?*" becomes a two -hop reasoning problem, where each sub-problem can either be answered formally with symbols, or using LLMs with natural language. For example, one might use LLMs to figure out that the *pan* is *cool*, a symbolic expression, based on some textual description. At this point, if the causality in Equation 4.1 is conveniently provided or otherwise inferred, we can deterministically deduce the final answer "*it is safe to touch the pan*" (more in Chapter 5). Otherwise, we may also express "*the pan is cool*" as a half-language, half-symbolic expression and feed it to LLMs to help them arrive at the answer. In either approach, we leverage the strong-suits of both neural and symbolic methods, whereas in this Chapter, I will focus on a semi-symbolic representation that is predicted by one LLM and fed into another to answer questions (see Figure 4.1).

I propose a semi-symbolic representation based on entities called an **entity schema** (Figure 4.2) that builds upon the previously discussed event-relation schema which models an event as a procedure including a goal event (e.g., *defog a window using potatoes*) and a sequence of ordered step events (e.g., *rub the cut side of potato on the window*). For each step, the schema models an array of 4-tuples describing an entity state change. Each 4-tuple contains an entity, an attribute, a state before the step, and a state after the step (e.g., *the window*'s *texture* was *smooth* before and *sticky* after). Essentially, the entity schema is a matrix where the axes are step, entity, and attribute, while the value is the before and after states. Unlike the previously discussed relational event schema where


starch
potatoes
knife
window
entities to track
Input: four steps in a procedure describing fog removal using potatoes
Output: generate a set of state change tuples for each step

	1. Rub the cut side of potato on the window.	...	4. Leave to dry without touching.
location	unknown → at car		no change
existence	exists		exists
cleanliness	clean → dirty		
transparency	fogged → partially clear		
clarity	opaque → translucent		
texture	smooth → sticky		
moisture			wet → dry
...	...	...	...
open attr.			touched → untouched

Figure 4.2: An example from the OpenPI dataset (Tandon et al., 2020).

events are expressed as natural language, here, the entities are both textual and symbolic (i.e., semi-symbolic), for they can not only be consumed by LLMs (this Chapter), but also possibly be consumed by symbolic algorithms or grounded to some environment (Chapter 5). To learn the entity schema, I first enhance an existing dataset (Tandon et al., 2020) by canonicalizing the entities, so that different mentions of *window*, *glass*, *pane* all fall under a unified symbol (Section 4.1). Using this enhanced dataset, I tune LLMs that can predict an entity schema given a procedure. I demonstrate their utility via a downstream task to causally reason about entities and events. There, I show that modern LLMs trained on a mixture of code and text can effectively leverage the event schema in the form of a Python-code (Section 4.2). Finally, I show that such a code-like form does not perform equally well on a variety of other NLP tasks (Section 4.3).

In the Chapter 3, I have been using the pre-train then fine-tune paradigm with the BERT-family LLMs. In contrast, the discussions in this Chapter will revolve around the in-context learning paradigm made possible by the state-of-the-art GPT-family LLMs (see categorization described in Section 2.2.1), where the LLMs are interfaced via few-shot prompting. This paradigm shift is preferred as it eliminates the need of a sizeable fine-tuning dataset, offering more flexibility.

4.1. Learning Entity States

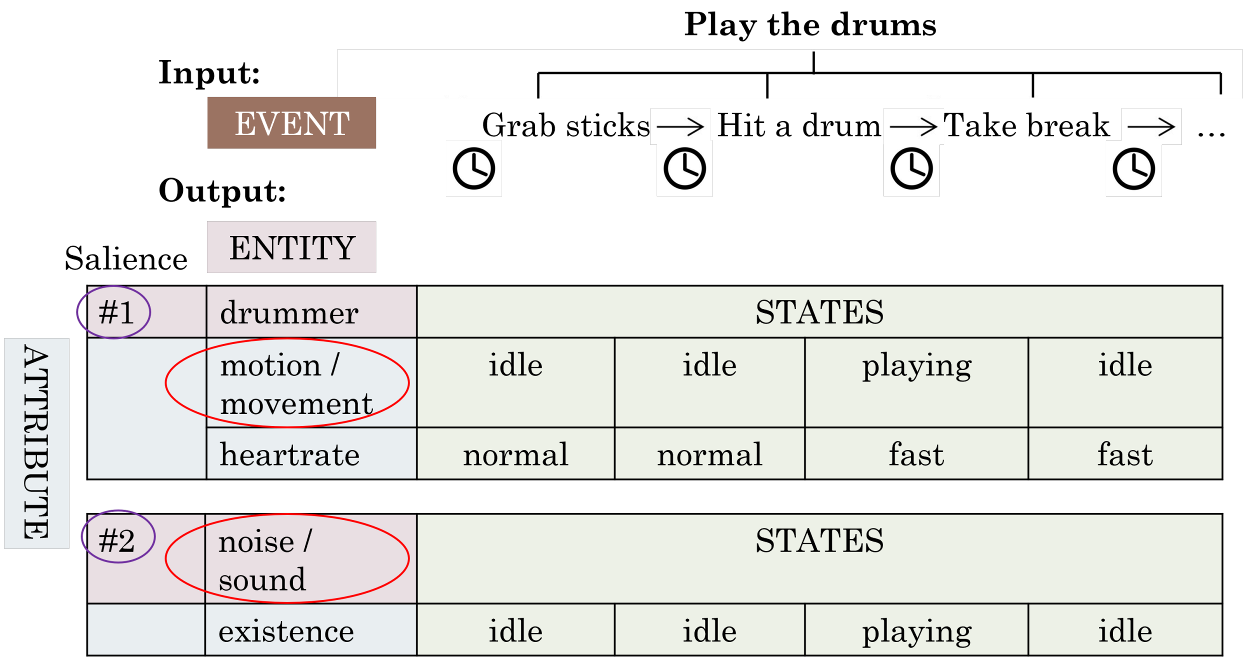
My overarching goal is to train and evaluate LLMs to predict an entity schema given a procedure, so that this information can be used for reasoning in later sections. To do that, my collaborators and I start with the OpenPI dataset (Tandon et al., 2020) with crowdsourced annotations of the entity states in procedural texts (Figure 4.2), identify its shortcomings, and and propose an improved OpenPI2.0 dataset.

4.1.1. Background

Tracking entity states in procedural texts is closely related to many NLP reasoning tasks. To name a few, question answering (QA) about events (e.g., *why use gloves when retrieving the tray from the oven*) often require knowledge of entity states (e.g., *the tray becomes very hot while in the oven*) (Tandon et al., 2019; Spiliopoulou et al., 2022); planning (Wang et al., 2022; Brohan et al., 2023) largely involves actions upon entities resulting in state changes. Procedural entity tracking is challenging in itself, requiring much understanding of an implicit environment as well as external knowledge of what events affect which entities, and how.

Prior work on entity state tracking spans various disciplines of AI. For instance, object tracking, a sub-task of entity state tracking, has led to much work in both robotics (Wang et al., 2007) and computer vision (Comaniciu et al., 2003). In NLP, early efforts focus on synthetic, closed-domain data (Weston et al., 2015; Long et al., 2016) and more recent ones shift attention to real-world procedures (Bosselut et al., 2017; Dalvi et al., 2018; Gupta and Durrett, 2019; Du et al., 2019; Mysore et al., 2019) with a closed set of entities and attributes. The only open-ended dataset to our knowledge is still OpenPI (Tandon et al., 2020) which we build on.

A small body of work on entity salience has focused on annotating entity salience in news articles and web pages for better information retrieval, recommendation, and linking (Gamon et al., 2013; Dunietz and Gillick, 2014; Dojchinovski et al., 2016; Trani et al., 2018; Wu et al., 2020). In contrast, we focus on entities in procedural texts, situating our work in script learning, robotic execution, automatic planning and reasoning, etc. Due to this mismatch of purpose, the definition,



		STATES			
#1	drummer				
	motion / movement	idle	idle	playing	idle
	heartrate	normal	normal	fast	fast

		STATES			
#2	noise / sound				
	existence	idle	idle	playing	idle

Figure 4.3: For each step in a procedure, OpenPI annotates the state change of attributes of entities. Our OpenPI2.0 additionally canonicalizes the entities and attributes (red circles) and includes their salience scores (purple circles).

annotation process, and downstream applications of our entity salience and theirs are all fundamentally different.

We propose the OpenPI2.0 dataset which builds on OpenPI (Tandon et al., 2020) (Open Procedural Inference), a large-scale dataset for tracking entity states in procedural texts from wikiHow.com. It contains annotations of entities, attributes, and state changes for each step (e.g., after the step "set the pan in a heated oven", the *pan*'s *temperature* was *cool* before and *hot* afterwards). OpenPI2.0 features two critical improvements (see Figure 4.3 for a demonstration of key features of OpenPI and OpenPI2.0):

1. **Canonicalization**. Originally, different mentions of the same entity or attribute render evaluation difficult. Here, we prompt LMs to effectively cluster the entities and attributes.
2. **Entity Salience**. Originally, all entities that undergo changes are listed in parallel. Here, we provide both human and model-predicted annotations of their salience.

Regarding canonicalization, clustering paraphrases evidently allows for fairer evaluation. Moreover, as our task of predicting entities, attributes, and states is a generation task with imperfect and incomplete ground-truth references, we show that expanding each entity or attribute cluster with possible paraphrases (thus providing more references) is effective for reducing the false-negative rate. We then comprehensively report various state-of-the-art LMs' performance of entity tracking on OpenPI2.0.

Regarding entity salience, we provide both manually annotated and automatically calculated labels. We evaluate them based on correlation with ground-truth data, and show that LMs can reliably predict entity salience with a close-to-human performance. We argue that salient entities acts as a means of *compression* of the most critical information in procedural texts, similar to saliency maps in computer vision (Simonyan et al., 2013). We proceed to qualitatively and quantitatively show that salient entities, as chain-of-though of LM prompting, benefit downstream tasks such as QA and classical planning, while reducing cost by excluding less important entities in the prompt.

4.1.2. Canonicalization

In the original OpenPI dataset, the entities and attributes that undergo change were written by crowd workers. Consequently, the dataset contains different ways of expressing the same entity (e.g., *coffee maker*, *coffee machine*, *espresso machine* in a coffee-making procedure) or attribute (e.g., *texture*, *smoothness*, *sheen* of a paint). Canonicalization by clustering the entities and procedures is thus important for two reasons: 1) it facilitates evaluation especially in a generation setting, where a model might be wrongly penalized for predicting a paraphrase of some correct entity or attribute; 2) it facilitates further annotation of features such as salience (§4.1.3) of the entities and attributes.

Clustering Entities and Attributes While canonicalization seems straightforward, it is non-trivial in OpenPI2.0 because clustering is highly context-dependent. For example, the entity *torso* and *paper chunk* usually have nothing to do with each other, but in fact refer to the same thing in a procedure of making a paper bird.

Role	Content
User	I am trying to make coffee. First, I put some coffee beans and tap water into the corresponding compartment of the espresso machine. Then, I select the desired type of coffee to make produced. Then I put a mug under the espresso machine and press start. Do you get it?
Assistant	Yes.
User	We have the following objects: *water*, *coffee maker*, *coffee machine*, *mug*, *espresso machine*. Group those that refer to the same thing. You must include all the provided entities. Do not add any entity that is not provided in the list.
Assistant	<start of generation> The grouped objects are: - ['water'] - ['coffee maker', 'coffee machine', 'espresso machine'] - ['mug']

Table 4.1: Our chosen prompt for entity and attribute clustering.

	Entity	Attribute
Cluster Recall	.425	.881
Cluster Precision	.593	.906
Cluster F1	.495	.893

Table 4.2: Evaluation of entity and attribute clustering.

Clustering Due to the contextual nature of the task, we prompt one of the SoTA LMs `gpt-3.5-turbo` (a.k.a. ChatGPT)[22] as shown in Table 4.1. We use 3-shot prompting, meaning that the complete prompt includes three handwritten examples and the prompt header of the example to be inferred, only containing the "User" role. The temperature is as default (0.7) and so are other hyperparameters. We aggregate output from five runs of ChatGPT as the final entity cluster and three runs for attribute cluster, as doing so is found to be empirically superior than single-pass generation[23].

To see if our model can cluster entities and attributes effectively, we evaluate the results using precision, recall, and F1 scores with exact match against a set of manually-labeled clusters from 20 procedures in the development set.

We see that ChatGPT scores better in clustering attributes compared to entities. Error analysis shows that two factors contribute to this performance discrepancy. First, most attributes describe the physical properties of an entity. Therefore, attribute clusters are less context-dependent com-

[22] platform.openai.com/docs/models/gpt-3-5

[23] With results from the 5 runs, individual Entity clusters are added to the final cluster based on their number of occurrences. For instance, if `(pan, pot)` occurred 5 times, then it will be added to the final cluster first.

	schemata (global)				schemata (local)				states	
	F1	F1+exp	BS	BS+exp	F1	F1+exp	BS	BS+exp	acc.	BS
`gpt-3.5-turbo`	.151	.249	.843	.869	.025	.039	.798	.804	.074	.600
`text-davinci-003`	.362	.450	.891	.920	.130	.155	.798	.810	.225	.682
`LLaMA 65B`	.129	.174	.799	.820	.045	.060	.801	.800	.102	.577

Table 4.3: Exact match F1 or accuracy and BERTScore on the schemata and states prediction sub-tasks, with and without cluster expansion. The schemata sub-task is evaluated both globally (per-procedure) and locally (per-step).

	complete			
	F1	F1+exp	BS	BS+exp
`gpt-3.5-turbo`	.016	.016	.772	.790
`text-davinci-003`	.034	.034	.807	.821
`LLaMA 65B`	.117	.117	.429	.440

Table 4.4: Exact match F1 and BERTScore of complete sentences including an entity, an attribute, a pre-state, and a post-state, following the original OpenPI paper. Canonicalization and expansion lead to little help for exact match as it is only done on entity and attribute clusters, while the state names can still be expressed in many ways, causing false negatives.

pared to entity clusters. Second, many attributes are shared amongst entities. For instance, out of 1,145 attribute annotations in the development set, 204 of them are *"location"*.

Cluster expansion Though the existing entities and attributes are now clustered in OpenPI2.0, there may still be other paraphrases that a model might rightfully predict and wrongfully penalized for. Thus, we again prompt ChatGPT to expand the clusters by generating paraphrases given a cluster of entities or attributes (prompt omitted).

To evaluate the quality of entities and attributes generated from the expansion, we manually rate 20 procedures and find that 83.3% of the generated, paraphrased entities and 59.4% attributes are correct. This is largely because entity names are oftentimes self-explanatory and less context-dependent whereas the attribute names and their meanings are highly dependant on the context.

Utility: Evaluation of Entity Tracking Just as the original OpenPI, OpenPI2.0 is meant to be used to benchmark (or train) models on entity tracking – given a step in a procedure, predicting the state changes that entities and their attributes undergo. With the entities and attributes in OpenPI2.0 now fully canonicalized, evaluation can be done more fairly. To start with, we follow

	Correct	No change	Nonsense	Missing
003	585 (82.3%)	106 (15.0%)	14 (2.0%)	383 (20.3%)
3.5	303 (59.4%)	173 (33.9%)	34 (6.7%)	218 (42.7%)

Table 4.5: Error analysis on the schemata prediction task using two SoTA LLMs.

Role	Content
User	Here are some instructions on making coffee. - Buy fresh coffee beans. - Grind the coffee beans. - ... Now, I will provide you with a series of objects, and you will assign scores on a scale of 1-5 to them based on their importance in the instruction. Your answer should strictly be a numerical score followed by a one-sentence explanation.
Assistant	Sure, I can help you with that. Please provide the objects.
User	Coffee bean
Assistant	<generation> 5 - the coffee beans are the most important ingredient in making coffee.

Table 4.6: Our chosen prompt for predicting global or procedure-wide entity salience. For local salience, the wording is similar with only one step provided.

Tandon et al. (2020) and predict one **complete** sentence: "*attribute* of *entity* is *pre-state* before and *post-state* afterwards", which is then compared to such sentences in the ground-truth data (Table 4.4). We further make the evaluation more fine-grained by formulating two sub-tasks: i. predicting **schemata**, namely the entities and their corresponding attributes given a step (e.g., given "turn on the oven", the "temperature" of the "rack" undergo state changes), and ii. predicting the change of **states** given a step, an entity and an attribute (e.g., given the previous information, the state change is from "cool" to "hot"). This evaluation of first predicting a skeleton tensor of entities and attributes is highly practical, and stands in stark contrast with most previous work (§4.1.1) in closed-domain entity tracking, where states are predicted using *given* entities and attributes.

On the development set, we run three SoTA LMs: `gpt-3.5-turbo`, `text-davinci-003`[24] (Brown et al., 2020), and the open-source LLaMA 65B (Touvron et al., 2023). For each model, we start by separately tackling each of the two sub-tasks[25]; namely, a model first predicts attributes of entities (schemata) given a step, and then predicts a pre-state and a post state (states) given the gold entity-attribute pair. All experiments are via 1-shot prompting.

[24] platform.openai.com/docs/models/gpt-3-5

[25] To avoid error propagation, for states prediction, the ground-truth entities and attributes are provided.

For all settings, we consider both exact match (F1 for schemata and complete sentence prediction and accuracy for states prediction) and BERTScore (Zhang et al., 2019b) based on `deberta-xlarge-mnli` (He et al., 2021).

For the schemata prediction sub-task (Table 4.3), the atomic unit to be evaluated is an entity-attribute pair. We consider both a *global* evaluation, where predictions are made per-procedure (e.g., what attributes of entities undergo state changes in the procedure), and a *local* evaluation, where predictions are made per-step. This categorization will reappear in §4.1.3. Schemata prediction is naturally influenced by our entity and attribute clusters. Hence, for exact match we report F1 scores based on exact matches where any entity-attribute prediction that falls under an cluster, obtained by taking a Cartesian product of an entity cluster and an attribute cluster, is considered a true positive. For BERTScore, we calculate the maximum score of a prediction against all entity-attribute strings within all ground-truth clusters. Then, we report the mean score among all predictions as a macro average.

The states prediction sub-task (Table 4.3) is much more straightforward as the entity-attribute pairs are provided and a model only needs to predict a pre-state and a post-state for each. Thus, we simply report the exact match accuracy and BERTScore for each state.

Discussion and Error Analysis We observe that the predicting attributes of entities that undergo state changes is a highly challenging task even for SoTA LMs. Although evidently, expansion of clusters improves performance (fairly, as we have shown that the generated paraphrases are mostly correct), false-negatives that result in underestimation of models cannot be eliminated entirely. One interesting observation is that `text-davinci-003` greatly outperforms the supposedly more superior `gpt-3.5-turbo`. To gain even more insights into models' behavior, we analyze the model output for the schemata prediction sub-task. For each step, we annotate each entity-attribute prediction based on three labels:

- Correct, where the entity-attribute indeed go through some changes;
- Incorrect, because the entity-attribute actually does not go through any changes;

	Annotations Human (A2)	Predictions LM
Global	.759	.719
Local	.578	.400

Table 4.7: Pearson' r with human annotations (A1).

- Incorrect, because the entity-attribute is non-sensical.

In addition, we add any entity-attribute pairs that should have been predicted as going through some change, to measure models' recall. We randomly sample 20 procedures to perform this error analysis and the results are shown in Table 4.5.

Regarding precision, we find that while the majority of the predicted entities are correct, many of the predicted associated attributes are generic ones that do not undergo any change either locally or globally. For example, for the step "Purchase a blackboard eraser", the attributes predicted by `text-davinci-003` for the entity *eraser* are *location* (correct), *cleanness* (static locally), *shape*, and *size* (static globally). The issue is much more pronounced with `gpt-3.5-turbo`, with predictions such as *location* of *seller*, *name* of *brand*, etc, despite that the prompt clearly explains the desired output with an example. We attribute such performance discrepancy to `gpt-3.5-turbo`'s decreased ability to follow examples and its inability to understand nuanced instructions. Regarding recall, both models fail to predict many attributes that the human annotator deems changing. Upon qualitative inspection, most of these missing attributes are no less salient than the predicted ones, suggesting that this issue cannot be explained away with only missing secondary attributes which may be plenty.

We leave to future work the resolution of these issues, which can be mitigated by re-prompting the models by validating if the predicted attributes indeed undergo changes, or simply have them predict the state changes altogether in the first place.

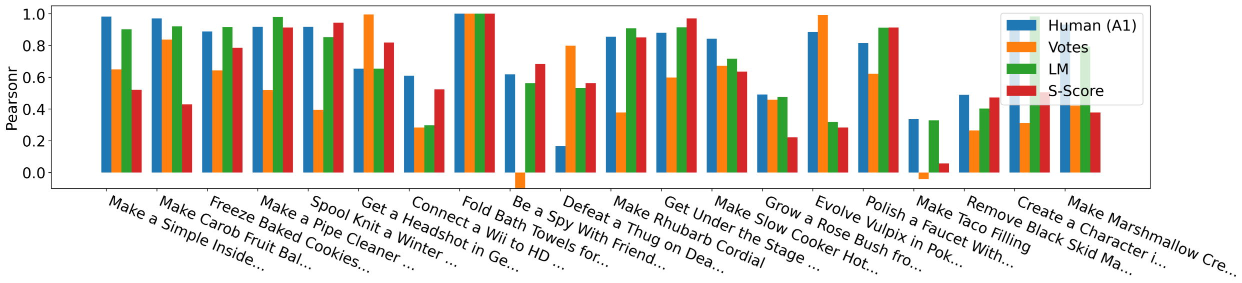


Figure 4.4: Per-procedure correlation of global entity salience between each set of annotations and the ground-truth human annotations.

4.1.3. Salience

The original OpenPI is annotated with many parallel entities in each procedure. Often, they vary greatly by importance in accomplishing the task. For example, in a procedure of cooking a steak, entities *fish*, *oven*, *gloves*, and *spice rack* might all be involved, while some are more indispensable than the rest. In OpenPI2.0, we define two types of entity salience: the global salience refers to the importance of an entity in accomplishing the goal of the procedure, whereas the local salience refers to that in a step.

Human Labeling To first procure ground-truth salience labels, two authors (referred to as A1 and A2) annotated entity salience in the first 20 procedures in the development set as the gold standard of entity salience. We devise and follow these annotation criteria in a Likert scale:

5: without or without mentioning this entity, the procedure or step cannot be done at all (e.g., *lemon* in "Wash faucet with lemon")

4: without this entity, another entity of the same type can be used as a replacement, perhaps with worse outcome or more efforts (e.g., pan in "Sear a salmon" - can also use *grill*)

3: without this entity, the procedure or step can be done in principal, though with slightly worse outcome or more efforts (e.g., *glove* in "Cut off tough branches of a bonsai plant")

2: without this entity, the procedure or step can be done, though with negligibly worse outcome or more efforts (e.g., *vacuum cleaner* in "Drill holes in the wall")

1: the entity appears in the procedure or step rather gratuitously, and the lack thereof makes no difference

0: the entity is irrelevant to the procedure or step

Subjectivity is inevitable even though we strive to minimize subjectivity using this fine-grained scale to capture nuanced situations (e.g., an entity that frequently appears that can be easily replaced versus one that appears only once but is irreplaceable). In later sections, we will see how this scale leads to reasonable inter-annotator agreement and favorable performance on downstream tasks.

LM Prediction We prompt `gpt-3.5-turbo`, as before, to automatically predict salience. Table 4.6 shows an example prompt for predicting global salience. As before, we use the default hyperparameters with a temperature of 0.7. We parse the result by extracting the first digit from the generation as the score, and default to 1 whenever impossible.

Evaluation To first holistically evaluate the modelling of salience, we report pairwise Pearson's correlation coefficients between each set of labels above and the annotations of human A1. In Table 4.7, we report a "macro correlation", namely the mean of correlation of salience scores in each procedure.[26] First, the correlation between the two annotators is high but imperfect, implying subjectivity in the annotation of entity salience. In comparison, the LM predictions come close with especially impressive predictions for global salience.

To understand when and how entity salience can be subjective among humans, in Figure 4.4 we show salience correlation for the first 20 procedures. Some with low inter-human correlation such as "Defeat a Thug" expose a *document-frequency* problem: one human labels the entity "you (actor)" with a salience of 5, believing that without the actor the procedure cannot be undertaken at all, while the other labels 1, believing that the actor occurs in every procedure and thus the mention of it makes no difference. Other procedures such as "Make Taco Filling" expose a *cluster-individual* problem: all individual ingredients are grouped under one cluster, for which annotators are divergent on the salience. While such issues can be mitigated by better defining entity salience in a potential future annotation process, it is much more practical to consider downstream applications.

Using our methods above and adding the model predictions for canonicalization and salience to the

[26] To avoid NaN due to constant input array, a 0 is appended to each array as smoothing.

Role	Content
User	A person walks into a dark room in a **freezing windy night**. In the very beginning before anything is done, how likely is it that she can see what's in the room? Answer on a scale of 1-5, with a 1-sentence explanation.
Assistant	1 - it is highly unlikely for the person to see anything in a dark room without any source of light.
User	Next, she lights up a torch and hangs it near a window. At this point, how likely (same question)...
Assistant	5 - hanging the torch near a window maximizes the distribution of light in the room, greatly enhancing the visibility and increasing the likelihood of seeing what's inside.
^{1}User	Next, she **opens the window**. At this point, how likely (same question)...
^{1}Assistant	**5** - opening the window allows **natural light** to enter the room, combining with the torchlight to provide ample illumination, greatly increasing the likelihood of seeing what's in the room.
^{2}User	Next, she opens the window. The most salient entities for this step are the window, the wind, and the flames. How does each of these entities change?
^{2}Assistant	When she opens the window, the following changes occur: Window: The **window is now open**, allowing for air and potentially some outside light to enter the room. Wind: The wind becomes more noticeable and **enters the room through the open window**, causing a draft and potentially affecting the flames of the torch. Flames: The flames of the torch may flicker or change direction due to the entry of wind through the open window, **altering their shape and intensity**.
^{2}User	With such information, at this point, how likely is it that she can see what's in the room?
^{2}Assistant	**3** - opening the window allows some outside light to enter, the presence of wind and potential **disruption to the flames** may still impede visibility to some extent.

Table 4.8: `gpt-3.5-turbo`'s performance on a CREPE-like example. The interactions with and without using entity salience are both shown. Critical information is illustrated in bold.

OpenPI dataset, we now fully present OpenPI2.0. As the procedures and entity state annotations have not changed, OpenPI2.0 still has 55 procedures with 5.0 steps on average. These procedures are collected from wikiHow and their topics are everyday activities. OpenPI2.0 also inherits the original entity-attribute-state changes annotated by crowd workers. After canonicalization, there are 356 canon entities each with 7.6 unique mentions and 5.5 expanded mentions on average, 3240 canon attributes, each with 3.0 unique mentions and 3.3 expanded mentions on average, and 1193 before-after states in the development set. The quality of clustering and expansion and be evidenced in §4.1.2. Regarding salience, the global salience of entities has a mean of 3.5 and standard deviation of 1.4; the local salience of entities has a mean of 3.4 and standard deviation of 1.5.

Regarding the training set, OpenPI2.0 is no different than OpenPI, for one may either fine-tune an LLM to predict entity states (i.e., produce the entity schema, my proposed structured representation of a procedure) for LLMs with the pre-train then fine-tune paradigm. However, to take advantage of the powerful LLMs with the in-context learning paradigm, OpenPI2.0 allows for effective *model selection*, which entails not only selecting an LLM and its hyperparameters, but also prompt tuning.

In other words, OpenPI2.0 leads to LLMs that can best predict the entity schema of events. I will continue to discuss how that entity schema can effectively work with LLMs to outperform end-to-end usage in some challenging event reasoning tasks.

The work above was published in Zhang et al. (2023b), in which I primarily contributed to all components except the canonicalization part to which Hainiu Xu primarily contributed. I have obtained approval from all collaborators to exclusively include this work in this thesis.

4.2. Application of Entity Schema

We now have LLMs, evaluated on OpenPI2.0, that can effectively predict entity states. Namely, an entity schema can be derived from procedural texts. We are now ready to tackle the task of causal event reasoning, exemplified in the beginning of this Chapter. We will see how a structured representation of said entity schema contributes to model performance.

Before diving into the application, I will first clarify how exactly the entity schema can interface with or be fed into LLMs. Recall that in Chapter 3, the event-relation schema interfaces with LLMs via fine-tuning. This is possible because all the events are natural language sentences and the relations among events are learned via labeled data. Regarding the entity schema, one can similarly fine-tune LLMs to predict *some part* of the schema, such as the pairwise relation between an event and an entity. However, I try to be more ambitious and attempt to feed the *entire* entity schema as input to LLMs to facilitate their reasoning. In this past, this would not have been possible because LLMs were pre-trained only on natural language, unable to work with a structured representation such as a matrix, graph, or some data structure. Fortunately, at the time that work described in this Chapter took place, latest LLMs are also trained on structured data such as code, in addition to text. Therefore, these LLMs can not only take structured representations as input but also predict them. Along with this advance is the in-context learning paradigm with many advantages over the pre-train then fine-tune paradigm (Section 2.2.1). Most noticeably, it reduces the need for manually labeled data for LLMs to make predictions. In summary, it is now possible for the the cutting-edge LLMs to not only predict by consume an entire entity schema. Even so, how exactly one should represent the schema (i.e., the form; e.g., as a matrix, graph, or Python code) is still a key design choice.

Unlike the form discussed above, the term *structure* (e.g., structured reasoning, event structure, etc.) has a much deeper semantic meaning, referring to the underlying representation of an event. In summary, an unstructured representation of an event is necessarily in the form of natural language, whereas a structured representation can either take the form of natural language or symbolic language.

4.2.1. Motivation

The earlier example of whether "*one can safely touch the pan*" is different from typical QA examples like "*how many states are there in the US*" in that the former question is grounded to a particular environment. In other words, the answer depends on the exact configuration the environment, which in turn is decided by the events that have previous taken place. While the environment can be specified in many possible ways, procedural text, as we have extensively discussed in Chapter 3, is a common descriptor of an environment that changes dynamically through a sequence of steps. Even so, the exact environment configuration is often implicit (e.g., we know that "*we boil the water*", but we are not explicitly told that "*the water is hot.*" Such a QA task is an instance of event reasoning - or, specifically, procedure reasoning - that this thesis focuses on. With these interesting challenges coupled with the added benefit of application to robotics (Brohan et al., 2023) and household smart assistants such as Alexa (Panagopoulou et al., 2022), reasoning about procedures attracts great attention from the NLP community.

Most work on reasoning about procedural texts has focused solely on either predicting the properties of events (e.g., which event is more likely to happen) (Zhang et al., 2020c; Yang et al., 2021; Tandon et al., 2019) or tracking entity states (e.g., what is some property of an entity after some step) (Dalvi et al., 2018; Tandon et al., 2020), while the causal relation between events and entities is largely underexplored – for example, whether "*there is a sizzling sound*" is determined by the state of "*water*" and "*oil.*" Therefore, we claim that many event prediction tasks are multihop reasoning tasks that require the knowledge of intermediate entity states. Causal reasoning about events and entities differs from existing multihop reasoning tasks, such as Yang et al. (2018); Dua et al. (2019)

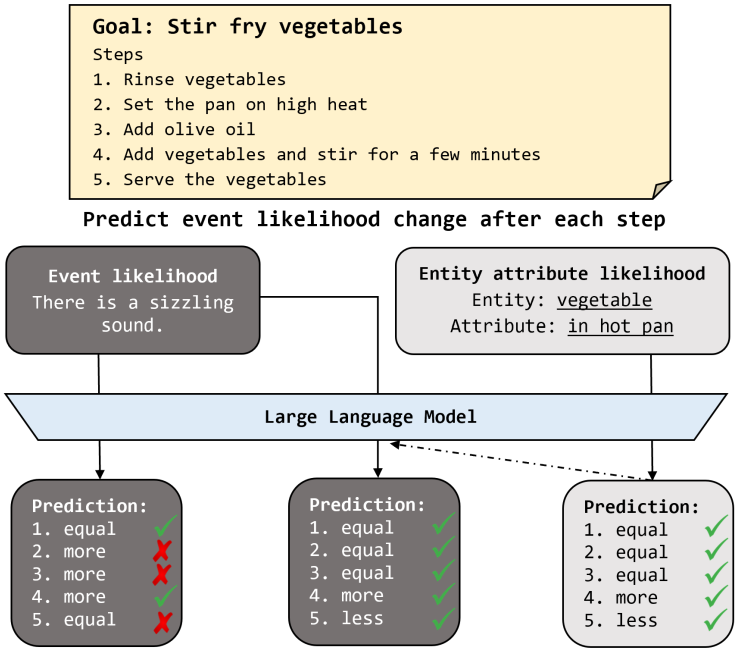


Figure 4.5: Example of our task CREPE. A procedure including a goal and some steps are provided. A model needs to predict the change in the likelihood of an event throughout the procedure. We show that predicting entity states as an intermediate step improves performance.

whose reasoning process is explicitly formulated by a direct question (e.g., *how old is the previous US president*); and Geva et al. (2021) whose supporting evidence is factual and static. In contrast, causal reasoning in procedures requires models to first figure out the relevant entity attributes, then infer their states based on the current context, and finally predict the event.

To this end, we propose the task of **C**ausal **R**easoning of **E**ntities and **E**vents in **P**rocedural Texts (**CREPE**), with an overview in Figure 4.5. Given a procedure consisting of a goal ("*stir fry vegetables*") and some steps ("*rinse vegetable*"...), a model is to predict the likelihood of some unobserved events ("*there is a sizzling sound*") after the execution of each step. This kind of hypothetical, counterfactual event reasoning is a high-level cognitive ability beyond pattern recognition and a manifestation of complex reasoning ability (Pearl and Mackenzie, 2018; Pearl, 2019). Counterfactual reasoning has a long history with formal methods (Forbus, 1984; Lewis, 2013). Less modern work exists in commonsense (Feng et al., 2021), procedural texts (Tandon et al., 2019), and even computer vision (Yue et al., 2021).

4.2.2. Task and Hypothesis

I will first formally describe the task of CREPE. A procedure P of length n consists of a goal G and some steps $s_1 \ldots s_n \in S$, each represented as a short sentence. Each procedure is associated with a set of hypothetical events $e_1 \ldots e_m \in E$ whose likelihood of happening changes throughout the procedure. The task is to predict the change of likelihood of a hypothetical event e_j from step s_{i-1} (the previous step) to step s_i (the current step):

$$\delta_i = p\left(e_j | s_i, \ldots, s_1, G\right) - p\left(e_j | s_{i-1}, \ldots, s_1, G\right)$$

The likelihood change δ_i is positive if the label is "more likely", negative if "less likely", or zero if "equally likely".

In our work, we hypothesize that **the causal relation between entity changes and events** can be leveraged by LLMs to better perform counterfactual reasoning. In other words, any change of the likelihood of a hypothetical event is given rise to by changes of some entity attributes $a_1 \ldots a_m \in A$.

$$\delta_i = p(a_j | s_i, \ldots, s_1, G) - p(a_j | s_{i-1}, \ldots, s_1, G)$$

4.2.3. Dataset

Our CREPE benchmark dataset has two portions. The first is handcrafted and cross-validated by six authors of this paper. The annotation happens in 3 phases: (1) we first write down or acquire a procedure from the web; (2) we then annotate some hypothetical events whose likelihood of happening changes throughout the procedure, and how their likelihood change after each step; (3) for each event, we annotate a tuple of entity, attribute, and change that causes the event likelihood change. To obtain interesting and challenging data, we require annotators to write procedures covering a diverse range of topics and to prioritize events that undergo multiple likelihood changes, and those that involve information implicit from the steps. In our work, we strictly use this portion as the development set to inform all our experimental designs.

Data Statistics			
	Dev	Test	Total
Num. procedures	42	141	183
Num. steps	295	924	1219
Num. event changes	144	180	324
Avg. step per procedure	7.0	6.6	6.7
Avg. token per step	6.8	6.8	6.8
Procedure Topics			
	Dev	Test	Total
Recipe	10	33	43
Household	12	40	52
Craft	4	17	21
Technology	5	19	24
Travel	4	4	8
Sports	2	13	15
Others	5	15	20

Table 4.9: Statistics of the CREPE dataset.

The second portion, designed to be drawn from a different distribution to minimize bias, was annotated by students in an Artificial Intelligence class at the University of Pennsylvania who participated in an extra-credit assignment. The students were given an overview of the project and some guidelines to annotate data with the aforementioned criteria. We carefully validated all resulting annotations by discarding or editing erroneous and inappropriate examples. In our work, we strictly use this portion as the test set to evaluate the generalization ability of our final models. The complete dataset and annotation instructions can be found in our public repository containing no personally identifiable information of any annotator.

The statistics of CREPE are in Table 4.9. In this work, we consciously focus on few-shot and in-context settings because our data annotation inevitably contains bias and limitation, and thus cannot be truly representative of counterfactual reasoning in every scenario. In such cases, we believe having a sizeable training set aggravates such biases and induces spurious artifacts.

The task of CREPE is essentially ternary classification, where the likelihood change of each event after each step is labeled as one of “more likely”, “less likely”, or “equally likely”. Here, all models

```
Goal: Wash sneakers
Context: I remove shoelaces. I rinse.
Question: What is the likelihood that my feet get wet by wearing the sneakers?
Answer: likely
```

Figure 4.6: Our GPT-3 prompt, which is typical for a QA task. Each likelihood label is compared with the previous one to get the label for the change.

have no access to the annotated entity state changes until later sections.

4.2.4. Text Form

To show the challenge CREPE brings to existing models, we first introduce some naive baselines.

- The **chance** baseline assigns random labels.

- The **majority** baseline always assigns the majority label "equally likely".

Next, we consider the following state-of-the-art LLMs as strong baselines, where all models are given exactly three examples in their prompt:

- **T5** (Raffel et al., 2020) is one of the state-of-the-art LLMs. Given the goal, steps, and question formatted by a prompt template, we compare the probability of generating "the answer is no|yes." We use `T0-3B`[27] with 3 billion parameters.

- **T0** (Sanh et al., 2022a) is a variant of T5, fine-tuned on a large set of downstream tasks with natural language prompts. We adopt the same inference process as T5 described above. We use `T0pp`[28] with 11 billion parameters.

- **GPT-3** (Brown et al., 2020) is a series of LLMs that excels at few-shot learning using the prompting mechanism. We consider `text-curie-001` (7B parameters), `text-davinci-002`, `text-davinci-003`, and `ChatGPT` (all 175B parameters). We use default parameters with a temperature of 0 for deterministic predictions. An example of the prompt is shown in Figure 4.6.

[27] https://huggingface.co/t5-3b

[28] https://huggingface.co/bigscience/T0pp

	Naive		**Large Language Models**								**Human**
	Cha.	Maj.	T5	T0	GPT3C	GPT3C+S	GPT3D2	GPT3D3	ChatGPT	Codex (ours)	
Params	-	-	3B	11B	13B	13B	175B	175B	175B	175B	-
Dev	.262	.297	.343	.336	.346	.341	.350	.424	.470	**.585**	.868
Test	.251	.296	.343	.337	.356	.346	.533	.423	.462	**.591**	-

Table 4.10: Macro F1 of baseline models on the CREPE dataset. Human performance is not benchmarked on the test set as we strictly hold out its labels during all experiments. GPT3C represents the `text-curie-001` model. GPT3D2 represents the `text-davinci-002` model with an abnormal performance on the test set that we have confirmed but regrettably cannot explain. GPT3D3 represents the `text-davinci-003` model. GPT3C+S represents the GPT-3 `curie` model finetuned on StrategyQA. All of the above models work with textual prompts. Codex represents the `code-davinci-002` model and works with our proposed code-like prompts.

- **GPT-3 finetuned on StrategyQA** is a GPT-3 `curie` model finetuned with StrategyQA (Geva et al., 2021), a dataset of factual multihop questions and their decomposition. StrategyQA is similar to our task in that estimating the change of event likelihood can also be decomposed into sub-tasks of estimating the change of state of related entities (Section 4.2.6).

Table 4.10 shows that all state-of-the-art LLMs we have attempted achieve close-to-chance performance on CREPE around 0.350 F1, whereas `text-davinci-003` and `ChatGPT` which are known to be stronger at reasoning perform only slightly better. These results showcase that the CREPE task is clearly challenging for even the strongest LLMs when used in an end-to-end manner. Previously, we have established that LLMs can now work with both a natural language form and a symbolic language form of the data. While the above representation is of course an ordinary **text form**, we will next explore a symbolic form of the examples in CREPE that will later be essential for the application of the entity schema.

4.2.5. Code Form

Codex (Chen et al., 2021a) is a variation of GPT-3 that was designed to be prompted with and to generate code, in addition to natural language texts. Shortly before the publication of this work, Madaan et al. (2022) found that prompting Codex with some structured representation such as Python code. Inspired by this observation, we propose novel code representations of procedures and hypothetical events. Among many possibilities we experimented with, the representation with

```python
class Wash_Sneakers:
  # Init
  # Remove shoelaces
  # Rinse
  def __init__(self, event0):
    self.event0 = event0 # My feet get wet by wearing the sneakers.
  def remove_shoelaces(self):
    self.event0.change = "equally likely" # My feet get wet by wearing the sneakers.
  def rinse(self):
    self.event0.change = "more likely" # My feet get wet by wearing the sneakers.
```

Figure 4.7: Our best-performing Python code representation of a procedure and hypothetical events, for Codex.

the best empirical performance is described below, later shown to greatly outperform all baseline models. The representation is exemplified in Figure 4.7.

The procedure is represented as a class where the goal G is the class name, followed by the steps s_i as comments. Then, each step is defined as a member function, in which the hypothetical events e_j are represented as objects with comments. Each event object has an attribute "change" whose value describes the change of the likelihood. During inference, Codex is provided with the prompt including three in-context examples and the current procedure up to the definition of the "init" function and predicts the definition of all step functions. Finally, we extract the assigned value of the "change" attribute as the event likelihood change δ_i.

This prompt design effectively leverages the semantic similarity between procedures with entity states and functions with variables, by representing texts as function identifiers and comments. We use `code-davinci-002`[29] with 175B parameters and default hyperparameters with a temperature of 0.

Results As CREPE is a ternary classification task, we report the macro F1 score across the three classes. As shown in Table 4.10, T5 and T0 perform only slightly better (.343 and .336 F1) than chance (.297 F1). GPT-3, one of the most dominant models across a variety of NLP tasks, is no

[29] While OpenAI announced that `text-davinci-002` is based on `code-davinci-002` (https://platform.openai.com/docs/model-index-for-researchers), we empirically find the former to perform worse with our code prompt and thus only consider the latter with code prompt.

	Dev	**Test**
Codex	**.585**	**.591**
no step comments	.377	.352
no event comments	.576	.555
nested function	.568	.572
flat variables	.338	.341

Table 4.11: Macro F1 of the ablations of our Codex prompt.

better (.336 F1), whereas finetuning it on another multihop reasoning dataset StrategyQA does not bring about any improvement (.341 F1). The latest GPT-3 models, `text-davinci-003` (.424 F1) and `ChatGPT` (.470 F1) which were released contemporarily with this paper, greatly outperform their predecessors.

On the other hand, our code-representation of events as the prompt to Codex greatly outperforms all other models with .585 F1. As Codex is trained on public Github code in addition to the internet texts that GPT-3 is trained on, it is noteworthy that Codex can effectively reason about texts with code-like structures, for a procedure has many analogies to a class in object-oriented programming.

Ablation Studies To understand why the representation in our Codex prompt is effective, we perform an ablation study with various changes of the format to the representation, including:

- Remove steps comments in the beginning
- Remove event comments in step functions
- Use nested functions instead of a class
- Use flat variables to encode goals, steps, and events (no hierarchical class functions)

As seen in Table 4.11, the hierarchical representation of procedures, steps, and events as classes or nested functions is critical. Besides, listing all the steps as comments helps, mimicking a programmer's textual explanation of a class or a function.

4.2.6. Injecting Entities

When a human tries to predict whether the event "*one would get burnt by touching a pan*" is likely, their reasoning process would first focus on some entities in the question (e.g., "the pan"), then attend to some attributes and states of that entity (e.g., the temperature of the pan is hot), and finally draw a logical conclusion (e.g., "the pan being hot means one would get burnt by touching it.") CREPE is constructed precisely with this thought process in mind. An entity-attribute-change tuple is annotated along with each event likelihood change. In this section, we study how to explicitly leverage the intermediate information to assist the prediction of event likelihood prediction.

In CREPE, the task of predicting event likelihood change can be seen as a case of multihop reasoning, where a model first decomposes the question into some open-ended sub-questions, answer these sub-questions, and aggregate them as a final answer. LLMs can be prompted to perform chain-of-thought (CoT) style reasoning (Nye et al., 2021; Wei et al., 2022b). Thus, we ask the question:

> **Q1**. Can LLMs benefit from first **predicting** entity state changes, as a CoT, before predicting event likelihood changes?

Text Form First, we prompt GPT-3 with Wei et al. (2022b)'s CoT paradigm and Press et al. (2023)'s self-ask paradigm, both of which are shown in Figure 4.8. While self-ask relies on search engines for fact retrieval, we use LM generation instead as most of our entity state tracking questions are heavily context-dependent and unanswerable by any search engine. When writing demonstrations for few-shot learning, we impose the following logic progression for the follow-up questions: (1) initial followups shall ask questions on the state of entities that are directly related to the event; (2) followups following the entity state questions shall ask for the logical relationship between the entity states and the original event.

Code Form We modify our Codex prompt in Figure 4.7, so that a sub-event is represented as a string variable whose declaration and value assignments are right before those of the hypothetical event. We refer to this as a *soft representation* of entities (Figure 4.9). During inference, Codex is provided with the code up to the step function header and predicts the entity and event changes for

```
Goal: Wash sneakers
Context: I remove shoelaces. I rinse.
Question: What is the likelihood that my feet get wet by wearing the sneakers?
Answer: To get feet wet by wearing the sneakers, the sneakers must be wet. In the given
context, the sneakers are wet. Therefore, comparing to the previous step, the likelihood
change is ”more likely”.
```

```
Goal: Wash sneakers
Context: I remove shoelaces. I rinse.
Question: What is the likelihood that my feet get wet by wearing the sneakers?
Follow up: Are the sneakers wet?
Intermediate answer: Yes
Follow up: Will my feet get wet by wearing wet sneakers?
Intermediate answer: Yes
Answer: likely
```

Figure 4.8: Our GPT-3 prompt with intermediate questions, mimicking the CoT prompt (top) and the Self-Ask prompt (bottom).

	Naive	**LLMs**		**CoT Large Language Models**				**Human**
	Majority	GPT-3	Codex	GPT-3+CoT	GPT-3+self-ask	Codex soft (ours)	Codex hard (ours)	
Dev	.297	.346	.585	0.359	.342	.624	**.667**	.868
Test	.296	.356	.591	0.379	.345	**.626**	.609	-

Table 4.12: Macro F1 of chain-of-thought models on the CREPE dataset. GPT-3 + CoT|self-ask represents the `text-davinci-002` model prompted with the CoT or self-ask style prompt.

every step function. Our Codex model achieves the new best performance of .624 F1, outperforming the same model without predicted entities as CoT by .039 F1.

```
class Wash_Sneakers():
  # Init
  # Remove shoelaces
  # Rinse
  def init(self, event0, subevent0):
    self.event0 = event0 # My feet get wet by wearing the sneakers.
    self.event0.subevent = subevent0 # The sneakers are wet
  def remove_shoelaces(self):
    self.event0.subevent.change =
      "equally likely" # The sneakers are wet
    self.event0.change = "equally likely" # My feet get wet by wearing the sneakers.
  def rinse(self):
    self.event0.subevent.change =
       "more likely" # The sneakers are wet
    self.event0.change = "more likely" # My feet get wet by wearing the sneakers.
```

Figure 4.9: Our Codex prompt with a soft representation of entity state changes as strings.

The two approaches above both *softly* represent the intermediate entity state changes as texts, either questions or statements. Here, LLMs are not enforced to generate intermediate reasoning steps that contain entities and attributes. To answer Q1 more precisely, we experiment with a *hard entity representation* where the entity-attribute-change tuple is explicitly baked into the Codex prompt as shown in Figure 4.10. Here, each entity is represented as an object with an attribute and assigned value. The hard entity representation leads to a far superior performance of .667 F1 on the development set but generalizes worse on the test set with .609 F1.

To recap, we have shown that LLMs can be prompted to exhibit a CoT that first predicts entity state changes and then event likelihood changes. Hence, our answer to **Q1** raised at the beginning of this subsection is 'yes.'

```
class Wash_Sneakers():
  # Init
  # Remove shoelaces
  # Rinse
  def init(self, event0):
    self.sneakers = Sneakers()
    self.event0 = event0 # My feet get wet by wearing the sneakers.
  def remove_shoelaces(self):
    self.event0.change = "equally likely" # My feet get wet by wearing the sneakers.
  def rinse(self):
    self.sneakers.wet = True
    self.event0.change = "more likely" # My feet get wet by wearing the sneakers.
```

Figure 4.10: Our Codex prompt with a hard representation of entity states as variables, attributes, and values.

	Dev	**Test**
Majority	.297	.296
GPT-3 CoT	.342	.345
w/ gold entity changes	.351	.380
Codex CoT	.667	.609
w/ gold entity changes	**.715**	**.722**
Human	.868	-

Table 4.13: Macro F1 of GPT-3 and Codex with chain-of-thought provided with gold entity state changes.

Annotated Entity States In the above section, we have shown how event likelihood prediction can be improved by first having the LLMs predict entity states as a CoT. These experiments mimic a realistic setting where information about entities is unavailable. However, in some scenarios, the entity states may be provided. For example, an embodied agent or a robot might have a reliable component that tracks entities; some practitioners might care about a small set of procedures in a narrow domain with annotated entity changes; or, some event schemata containing entity information could be used to predict unseen events. Here, we try to answer the following question:

> **Q2**. Can LLMs effectively leverage **annotated** entity state changes to better predict event likelihood changes?

Instead of having LLMs predict entity state changes, we provide the annotated entity state changes in the CREPE dataset to GPT-3 and Codex. Doing so has the additional benefit of verifying that entity state changes indeed causally benefit LLMs in predicting events.

As shown in Table 4.13, our Codex representation with access to gold entity changes leads to improved performance of .715 F1 on the development set. In contrast, GPT-3 does not see any gain. Hence, the answer to **Q2** is 'yes' for the code-trained LLMs but 'no' for standard LLMs.

Predicted Entity States As we will discuss further in Section 4.1.1, entity state tracking is an established task in NLP with existing datasets and models. We have now predicted entity state changes using LLMs in a few-shot learning setting. It is then natural to pose the question:

> **Q3**. Do existing entity state tracking models make predictions that lead to better performance on CREPE?

Our definition of causal reasoning of events is directional since we consider entity state changes as the cause of the change in event likelihoods. To this extent, we incorporate OpenPI (Tandon et al., 2020), the only open-domain entity state tracking dataset in procedural texts, as a part of the pipeline. In OpenPI, the input is a goal, a step, and the output is tuples of an entity, a feature, and two attributes before and after the execution of the step. For example, after "heat the pan [step]", "the temperature [feature] of the pan [entity] is cool [attribute] before and hot [attribute] afterward." While the original paper proposed a GPT2 model (Radford et al., 2019), we opt to finetune the superior GPT-3 Curie model on its data. After the model makes a prediction, we post-process it into the format of CREPE by discarding the feature and producing two entity-attribute-change pairs (e.g., pan-hot-"more likely" and pan-cold-"less likely"). We provide Codex with only the entity changes when the entity is mentioned in the event. Further, to fit our prompt in the context window of Codex, we provide Codex with 5 entity state changes uniformly drawn from a pool of candidate choices at every step. The resulting OpenPI-prompted Codex gives a degraded macro F1 score of 0.553 on the development set and 0.496 on the testing set. Hence, our answer to **Q3** is 'no,' suggesting that existing entity state tracking datasets may be insufficient for our causal reasoning task.

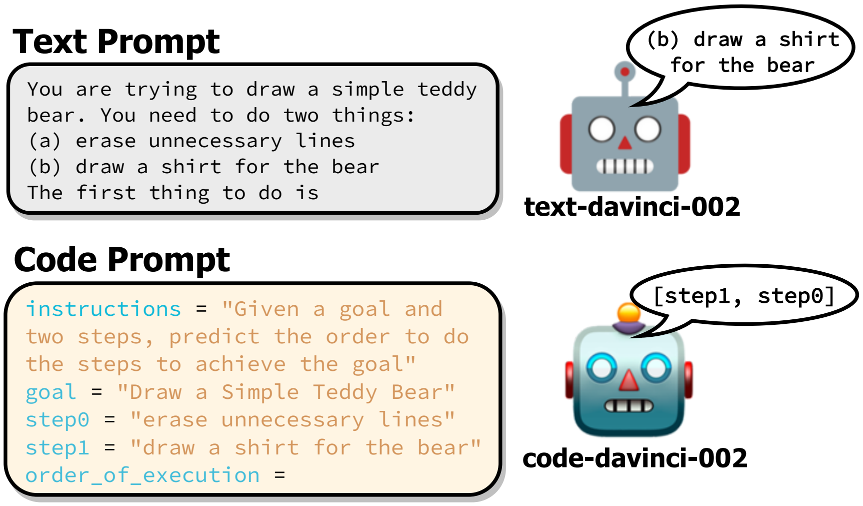
4.2.7. Analysis

In this section, we analyze potential factors that play a role in our Codex model's performance. We investigate three factors: (1) the number of steps in a procedure; (2) explicit mentions of event-related entity-of-interest (EoI) in a given step; and (3) the logical relation (entailment or contradiction) between the event likelihood change and its related entity state change. To study factor (1), we dichotomize procedures from the development set by the average length of the procedure. To investigate factors (2) and (3), we manually labeled the ground truth EoI mentioning and logical relation for the development dataset. Intuitively, estimating event likelihood in lengthy procedures and in steps where EoI is not explicitly mentioned would be difficult. Rather surprisingly, Codex shows no significant performance discrepancy under factors (2) and (3), and only a slight performance difference in factor (1).

Further, the task of CREPE can be divided into two sub-tasks, first to identify whether an event likelihood change occurred at all, and then to classify the change as either more or less likely. We observe that CoT Codex outperforms Codex on both sub-tasks. For the classification task, in particular, CoT Codex obtained a .149 increase in macro F1 score from .805 to .954. This shows not only that CoT Codex is effective, but also that its bottleneck is identifying event likelihood change.

In summary, we present CREPE, a benchmark for causal reasoning about events and entities in procedural texts. After establishing that end-to-end LLMs such as GPT-3 perform close to chance, we discussed two means of improvement, both critical to this thesis. First, we show that a code-like representation of the data can be fed as input to LLMs and greatly improves the performance. Hinging on this exciting finding, Section 4.3 explores whether this symbolic form works across other NLP tasks. Second, we show that the entity schema contributes to the success in CREPE, for the information about entities clearly helps reason about the counterfactual events.

The work above was published in Zhang et al. (2023c), in which I primarily contributed to all components. I have obtained approval from all collaborators to exclusively include this work in this



```
You are trying to draw a simple teddy
bear. You need to do two things:
(a) erase unnecessary lines
(b) draw a shirt for the bear
The first thing to do is
```

```
instructions = "Given a goal and
two steps, predict the order to do
the steps to achieve the goal"
goal = "Draw a Simple Teddy Bear"
step0 = "erase unnecessary lines"
step1 = "draw a shirt for the bear"
order_of_execution =
```

Figure 4.11: For certain tasks, prompting program-trained language models with code-*like* representations works better than prompting with text.

thesis.

4.3. The Versatility of the Code Form

4.3.1. Motivation

Any work that attempts to have LLMs work with a structured representation must decide the *form* that the representation takes (e.g., plain text, matrix, graph, Python code, etc.). Apart from our work above (Zhang et al., 2023c), several concurrent work has found that prompting such LLMs with a code form (e.g., Python, JSON, PDDL) instead of text leads to performance improvements on structured common sense reasoning (Madaan et al., 2022), event argument extraction (Wang et al., 2023), knowledge graph construction (Bi et al., 2023), and story understanding (Dong et al., 2022). Such results naturally lead us to ask whether prompting with the code form is the preferred way of interacting with code-trained LLMs *in general*. While previous work is limited to reasoning tasks, in this work we analyze a broad selection of tasks (e.g., QA, sentiment, summarization) and systematically compare the performance of prompting LLMs with code vs. prompting with text[30]. We find that:

- With the exception of some reasoning tasks, code prompts do not outperform text prompts

[30] The code, prompts, and outputs for our experiments are public at github.com/zharry29/curious_code_prompts

- The style of code prompt has a large effect on performance for some but not all tasks.
- Fine-tuning on text instructions leads to relative improvements when using code prompts.

4.3.2. Experimental Design

Model Selection For our text-based LM we use the original 175 billion parameter `davinci` model introduced by Brown et al. (2020). For our LLM we use the newer `code-davinci-002` model which was explicitly trained on text and code. Neither model underwent any supervised instruction fine-tuning. In addition, we analyze performance on `text-davinci-002`, which is a variant of `code-davinci-002` trained explicitly on human demonstrations using supervised fine-tuning[31]. We include this model to help us determine whether or not fine-tuning LLMs on text instructions affects their ability to interpret code prompts. All three models were queried through the OpenAI API[32] and our experiments cost approximately $2700 in total.

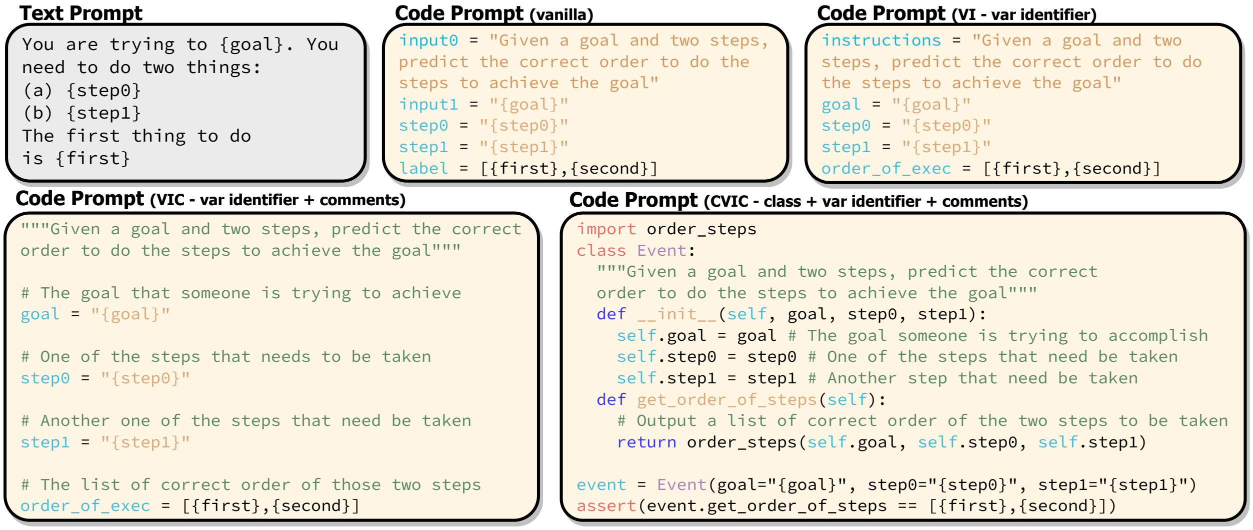
Dataset	Task Category	Num. Eval Examples	Metric	Origin
HellaSwag	Commonsense Reasoning	1000 / 10042	Accuracy	Zellers et al. (2019b)
wikiHow Goal-Step	Commonsense Reasoning	1000 / 1073	Accuracy	Zhang et al. (2020c)
wikiHow Temporal	Commonsense Reasoning	1000 / 3100	Accuracy	Zhang et al. (2020c)
WinoGrande	Commonsense Reasoning	1000 / 1767	Accuracy	Sakaguchi et al. (2021b)
OpenPI	Commonsense Reasoning	111 / 111	ROUGE-F1	Tandon et al. (2020)
ANLI	Natural Language Inference	1000 / 3000	Accuracy	Nie et al. (2020)
Yelp	Sentiment Analysis	1000 / 10000	Pearson's r	Ali (2018)
IMDb	Sentiment Analysis	1000 / 25000	Accuracy	Maas et al. (2011)
HotpotQA	Question Answering	1000 / 7405	Macro-F1	Yang et al. (2018)
SQuAD	Question Answering	1000 / 11873	Macro-F1	Rajpurkar et al. (2018)
CNN/Daily Mail	Summarization	1000 / 13368	ROUGE-2	Nallapati et al. (2016)
XSUM	Summarization	1000 / 11332	ROUGE-2	Narayan et al. (2018)

Table 4.14: The 12 evaluation tasks. Macro F1 is based on Rajpurkar et al. (2016). For each task, we randomly sample a fixed set of 1000 examples from its validation or test set for evaluation. For OpenPI we are limited to 111 examples.

Task Selection Following the methodology of Sanh et al. (2022b) we select tasks in a top-down fashion by first choosing the categories of interest (e.g. Question Answering, Sentiment Analysis, Summarization) and then selecting datasets from within those categories. We pay special attention to common sense and causal reasoning tasks as LLMs prompted with code have been shown to perform well on such tasks. The resulting 12 tasks are listed in Table 4.14 and include Commonsense Reasoning, Natural Language Inference, Sentiment Analysis, Question Answering, and

[31] https://platform.openai.com/docs/model-index-for-researchers

[32] https://openai.com/blog/openai-api



Text Prompt

```
You are trying to {goal}. You
need to do two things:
(a) {step0}
(b) {step1}
The first thing to do
is {first}
```

Code Prompt (vanilla)

```
input0 = "Given a goal and two steps,
predict the correct order to do the
steps to achieve the goal"
input1 = "{goal}"
step0 = "{step0}"
step1 = "{step1}"
label = [{first},{second}]
```

Code Prompt (VI - var identifier)

```
instructions = "Given a goal and two
steps, predict the correct order to do
the steps to achieve the goal"
goal = "{goal}"
step0 = "{step0}"
step1 = "{step1}"
order_of_exec = [{first},{second}]
```

Code Prompt (VIC - var identifier + comments)

```
"""Given a goal and two steps, predict the correct
order to do the steps to achieve the goal"""

# The goal that someone is trying to achieve
goal = "{goal}"

# One of the steps that needs to be taken
step0 = "{step0}"

# Another one of the steps that need be taken
step1 = "{step1}"

# The list of correct order of those two steps
order_of_exec = [{first},{second}]
```

Code Prompt (CVIC - class + var identifier + comments)

```
import order_steps
class Event:
  """Given a goal and two steps, predict the correct
  order to do the steps to achieve the goal"""
  def __init__(self, goal, step0, step1):
    self.goal = goal # The goal someone is trying to accomplish
    self.step0 = step0 # One of the steps that need be taken
    self.step1 = step1 # Another step that need be taken
  def get_order_of_steps(self):
    # Output a list of correct order of the two steps to be taken
    return order_steps(self.goal, self.step0, self.step1)

event = Event(goal="{goal}", step0="{step0}", step1="{step1}")
assert(event.get_order_of_steps == [{first},{second}])
```

Figure 4.12: An example of the four styles of manually written code prompts used in our analysis (`Vanilla`, `VI`, `VIC`, and `CVIC`) for the wikiHow temporal ordering task. At test time, variables in braces are replaced with information from the dataset item (as shown in Figure 4.11). For this task, **`{goal}`**, **`{step0}`**, **`{step1}`** refer to the article title and the steps to order while **`{first}`** and **`{second}`** refer to the true ordering of the steps.

Summarization.

Prompt Formulation We collect text prompts for each task using the PromptSource dataset (Bach et al., 2022), a publicly available collection of crowd-sourced prompt templates. For tasks with many prompts, we randomly select one from those provided in the dataset. For a few tasks absent on PromptSource, we write the prompts ourselves.

For our code prompts, we manually write four custom code prompts per task. The code prompt types are as follows, from least to most Pythonic.

(i). **Vanilla** (`Vanilla`): instructions and inputs are given as variables with generic names;

(ii). **Var Identifier** (`VI`): instructions and inputs are given as variables with meaningful names;

(iii). **Var Identifier + Comments** (`VIC`): instructions and inputs are given as variables with meaningful names along with comments explaining their purpose;

(iv). **Class + Var Identifier + Comments** (`CVIC`): instructions and inputs are given as a task-specific `class`. Functionality is "implemented" as member functions.

Dataset	Performance	σ
Hellaswag	0.65, 0.67, 0.69, 0.67, 0.67	±0.01
wikiHow-GS	0.51, 0.51, 0.51, 0.50, 0.51	±0.00
wikiHow-T	0.62, 0.65, 0.63, 0.63, 0.62	±0.01
Yelp	0.92, 0.92, 0.92, 0.92, 0.92	±0.00
IMDb	0.94, 0.94, 0.94, 0.94, 0.94	±0.00
WinoGrande	0.62, 0.64, 0.61, 0.62, 0.62	±0.01
HotpotQA	0.35, 0.33, 0.35, 0.35, 0.35	±0.01
ANLI	0.59, 0.58, 0.57, 0.60, 0.61	±0.01
OpenPI	36.3, 38.1, 38.3, 37.7, 39.9	±1.16
SQuAD	0.60, 0.62, 0.61, 0.60, 0.63	±0.01
CNN/DM	11.7, 12.0, 12.4, 12.3, 12.0	±0.25
XSUM	14.5, 14.9, 15.5, 15.2, 15.4	±0.36

Table 4.15: Comparison across 5 repeated runs of the `code-davinci-002` model with text prompts using different random seeds for sampling in-context examples. We see minimal standard deviation (σ) between the runs.

Figure 4.12 shows an example of the different styles of code prompts for the wikiHow temporal ordering task. Note that we attempt to write our code prompts such that we match the wording of the text-based PromptSource prompt as closely as possible.

At inference time, for each test example, we randomly sample in-context examples from the training set and add them to the context window until the maximum context length is reached. This process circumvents the bias caused by static in-context examples. We conduct an ablation study where we vary the random seed and show that this process produces consistent results. To see whether the findings in our Results section could be attributed to variance in the random sampling of in-context training examples per test example, we conduct five repeated runs using `code-davinci-002` with different random seeds each time and calculated the standard deviation across the five runs. We report our results in Table 4.15 and find that the choice of in-context examples accounts for very little of the observed variance across prompt type and context length. This finding is surprising as previous work has shown that the selection and ordering of in-context examples has a very large effect on the performance of models Liu et al. (2021). However, it seems that our approach of random sampling in-context examples per test item helps to lessen this inherent variance.

	`Vanilla`	`VI`	`VIC`	`CVIC`
HellaSwag	3	2	1	4
wikiHow Goal-Step	4	2	1	3
wikiHow Temporal	4	3	2	1
Yelp	4	2	1	4
IMDb	1	3	1	4
WinoGrande	4	1	2	3
HotpotQA	4	3	2	1
ANLI	1	2	4	3
OpenPI	1	2	3	4
SQuAD	1	3	4	2
CNN/Daily Mail	4	2	3	1
XSUM	2	4	3	1
Mean	2.75	2.42	2.25	2.58
Standard Deviation	1.36	0.76	1.09	1.26

Table 4.16: Relative performance rank of the four code prompt types from Section 4.3.2 across the 12 tasks. Ranks are calculated based on the results reported in Figure 4.13. We see that the "Variable Identifier + Comments" (`VIC`) style prompt performs the best out of all code prompt types on average.

4.3.3. Results

What is the best type of code prompt? We compare performance across the four code prompt types from Section 4.3.2 on all 12 tasks using `code-davinci-002` and report our results in Figure 4.13. We find that no single type of code prompt performs significantly better than the others across all tasks and that the relative difference in performance between code prompts also varies significantly across tasks. For example, on IMDb and SQuAD all code prompts have roughly even performance while for tasks such as wikiHow-Temporal and WinoGrande we see a near 14% accuracy difference between the worst and best prompt.

In Table 4.16 we report the rank-based statistics of the four code prompt types from Section 4.3.2 on our 12 tasks. Ranks are calculated based on the results reported in Figure 4.13 of the main paper. The numbers in a row reflect the relative standing of each code prompt on the corresponding task. While we note that all code prompts perform within ±0.5 ranks of each other on average, we see that on average the `VIC` prompt performs the best across all tasks and the `Vanilla` prompt performs the worst. Looking to the standard deviation section, we see that the `VI` prompt performs

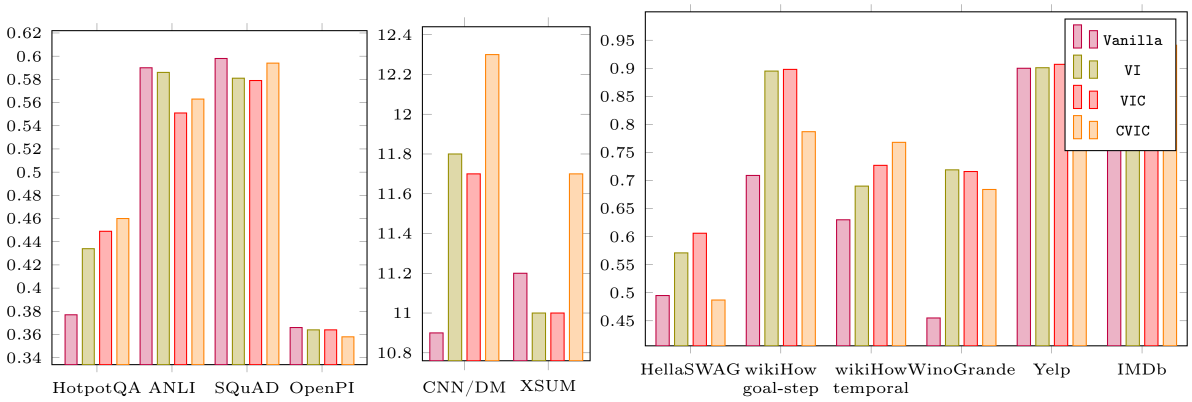


Figure 4.13: Comparison of `code-davinci-002` across the four types of code prompts. Figures are split to allow for different y-axis scales. We see that different prompts do better on different tasks and while some tasks have high variance over prompt types, others do not.

the most consistently across all tasks and that once again the `Vanilla` prompt performs the least consistently.

How many in-context examples should we include in our code prompt? We would like to also investigate how the number of in-context examples in the prompt affects models' ability to perform the task. We therefore conducted an experiment where we filled the context window of `code-davinci-002` with in-context examples up to 2000 tokens, 4000 tokens, and 8000 tokens and plotted the validation accuracy of the model with respect to the number of examples in Figure 4.14.

Contrary to expectations, we find that the number of in-context examples has little effect on model performance for most tasks and actually has a *negative* effect on some tasks. This is especially interesting given that previous work on in-context learning with text prompts finds roughly monotonic improvement from adding more in-context examples (Liu et al., 2022). While further research is necessary, it seems that code prompts may have different scaling behavior than text prompts when used in in-context learning.

Which is better: code or text prompts? In our main experiment we compare the performance of the three GPT models on code prompts (`VIC` style) and text prompts across the 12 datasets. Given the results from Figure 4.14, we fill the context window of all models with in-context examples up to 4000 tokens to serve as a middle ground for comparing code and text prompts. We report the

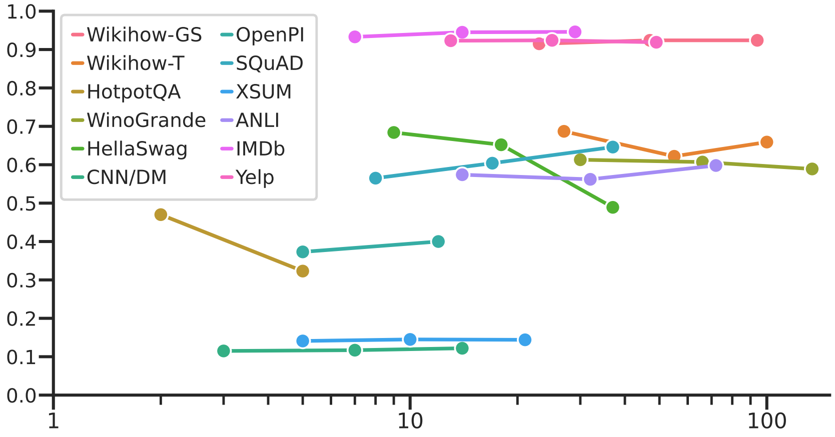


Figure 4.14: Performance score (y-axis) vs number of in-context examples (x-axis, in log scale) using code prompts (`VIC`) with `code-davinci-002`. We see that increasing number of examples does not always increase performance and in some cases makes it worse.

results of our main experiment in Table 4.17 and see several surprising trends.

First, we find that prompting LLMs with code leads to substantial increases in performance for certain few reasoning tasks but that this trend does not hold across all tasks—or even all reasoning tasks. For example, when using code prompts with `code-davinci-002`, we see a 10.5% accuracy increase on wikiHow temporal ordering but a 2.6% accuracy decrease on wikiHow goal-step inference despite both being commonsense reasoning tasks and having identical source material.

Second, we find that supervised instruction fine-tuning on natural language demonstrations does not hurt model performance on code. Rather, we observe that code prompts outperform text prompts on *more* tasks when using `text-davinci-002` than when using `code-davinci-002` despite the fact that `text-davinci-002` received no additional fine-tuning on code instructions.

Finally, we find that LMs not explicitly trained on code can also benefit from code prompting on certain reasoning tasks. In particular, code prompts outperform text prompts on `davinci` for 3 out of our 12 tasks—the same proportion as `code-davinci-002`. The tasks that benefit from code prompts also seem to be largely consistent across the three types of models tested, suggesting some underlying trend as to which tasks systematically benefit from structured input.

Dataset	Metric	`davinci`			`code-002`			`text-002`		
		+Text	+Code	Δ	+Text	+Code	Δ	+Text	+Code	Δ
Hellaswag	Accuracy	0.321	0.307	-0.014	0.652	0.606	-0.046	0.717	0.773	+0.046
wikiHow goal-step	Accuracy	0.347	0.302	-0.045	0.924	0.898	-0.026	0.919	0.915	-0.004
wikiHow temporal	Accuracy	0.495	0.532	+0.037	0.622	0.727	+0.105	0.688	0.761	+0.073
Yelp	Pearson ρ	0.913	0.896	-0.017	0.924	0.907	-0.017	0.919	0.904	-0.015
IMDb	Accuracy	0.872	0.935	+0.063	0.945	0.951	+0.006	0.940	0.952	+0.012
WinoGrande	Accuracy	0.513	0.500	-0.013	0.607	0.716	+0.109	0.628	0.726	+0.098
ANLI	Accuracy	0.333	0.360	+0.027	0.562	0.551	-0.011	0.504	0.557	+0.053
HotpotQA	Macro-F1	-	-	-	0.470	0.449	-0.021	0.490	0.350	-0.140
SQuAD	Macro-F1	0.482	0.466	-0.016	0.604	0.579	-0.025	0.670	0.656	-0.014
OpenPI	ROUGE-F1	-	-	-	37.33	36.36	-0.970	35.60	31.30	-4.300
CNN/Daily Mail	ROUGE-2	9.28	9.13	-0.150	11.74	11.67	-0.070	13.63	13.55	-0.080
XSUM	ROUGE-2	9.38	6.83	-2.550	14.51	11.03	-3.580	14.48	13.26	-1.220

Table 4.17: Performance of the three LMs when using code prompts (+Code) vs. using text prompts (+Text). Blank cells indicate tasks for which single test examples could not fit in the context window. Color indicates how code prompts compare to text prompts. We see that while code prompts outperform text prompts for certain tasks (such as wikiHow temporal and WinoGrande) text prompts are better on average. We also find that instruction fine-tuning (`text-002`) allows for better code prompt utilization.

Our rather anti-climatic findings above suggest that a symbolic form of data does not consistent benefit modern LLMs pre-trained with a mixture of text and code,[33] despite previously reported success of symbolic form on some event reasoning tasks.

The work above was published in Zhang et al. (2023a), in which I formulated the task and primarily contributed to a third of the tasks, with the efforts of data collection, model setup, and evaluation. I also performed the majority of the analysis. I have obtained approval from all collaborators to exclusively include this work in this thesis.

4.4. Summary

In this chapter, I introduced the entity schema, which is a semi-symbolic structured representation of events. I start by motivating the need of involving entities in the process of causal reasoning about events. Then, I propose the OpenPI2.0 that can be used to evaluate LLMs that can predict the entity schema. Using it, I show an improvement in the downstream task of event reasoning, and also drawing attention to the form that a structured representation should take to interface with LLMs. Finally, I show that LLMs can take advantage of pre-training on code and work better with

[33] Our findings directly contradict a contemporaneous work (Mishra et al., 2023), which experimented with a set of much smaller LLMs.

a code form of the entity schema. However, such a code form does not consistently benefit general NLP tasks.

At this point, my quest with “structured reasoning” has become more structured thanks to the semi-symbolic representation. However, the reliance on LLMs to make the final prediction lacks faithfulness. In other words, even were the representation to be completely correct, the LLM that takes it as input might still make a mistake. In the opening example of this chapter, even the knowledge of “the pan is hot” were to be given, the final answer that “one should not touch the pan” might not be reached, due to the non-deterministic nature of LLMs. To address this, in the next Chapter, I propose a fully symbolic representation that is no longer fed to LLMs, but deterministic algorithms.

CHAPTER 5

WORLD MODEL: A SYMBOLIC LANGUAGE REPRESENTATION

In all previous sections, we have focused on defining and predicting a structured event representation that can be leveraged by LLMs in different ways. As LLMs are the only type of models in each pipeline, the performance is limited by their power given any task. Despite success in previously discussed reasoning tasks, there exist tasks that are more challenging. One example is text-based **symbolic planning**, where the conditions and the goal are described using natural language, and a model must pick a sequence of actions to achieve the task. For example, the following are the instructions of a simple block-moving task:

> On a table, I have five blocks stacked on top of each other. From the bottom to the top, we have blue, white, yellow, and brown. Your goal is to rearrange them into red, white, blue, yellow, and brown from top to bottom. You may only move a block with nothing on top of it, and you can put such a block on a table. What should you do?

While even a human child is likely to find the task trivial by some trials, a state-of-the-art LLM like ChatGPT is completely incapacitated[34] (Valmeekam et al., 2023). Note that this is a symbolic reasoning task because even though the input is natural language, it describes a symbolic configuration, operating under a rule-based transition function, and the output space is also discrete and finite. Despite recent LLMs' emergent ability (see Section 2.2.1) to perform some symbolic reasoning tasks to some extent, this failure still stands in contrast with LLMs' success in generating non-symbolic, unstructured plans, perhaps in response to a query like "how to cook eggs."

Let us also examine a second example, a grade-school math question:

> I have 5 apples. You have 4. How many do we have in total?

Solving the problem requires the natural language ability to understand that '*in total*' means addition as well as the symbolic reasoning ability to mathematically perform the addition. While LLMs

[34] Experiment is performed in January 2024 with OpenAI ChatGPT4.

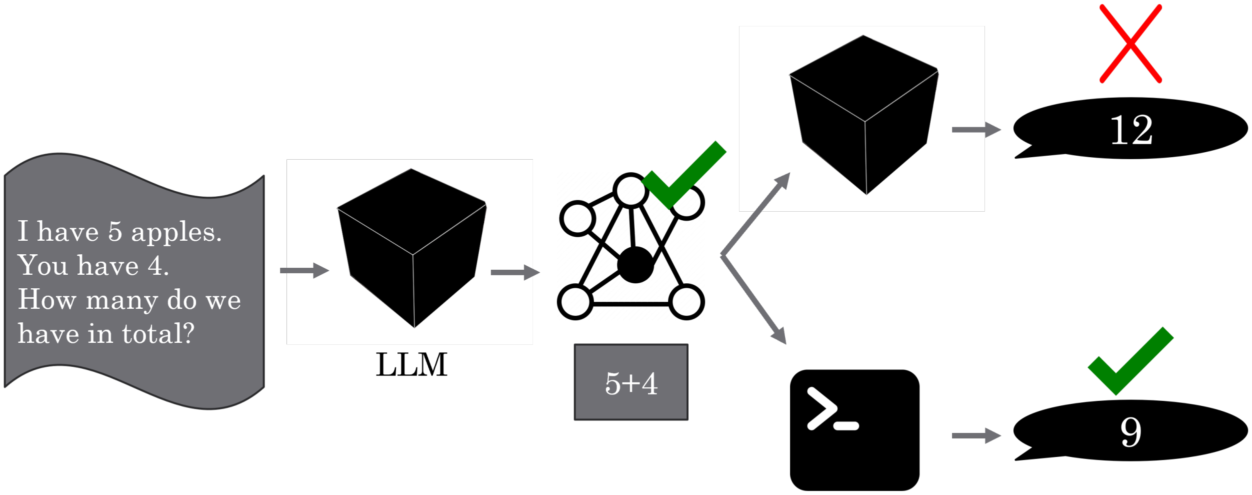


Figure 5.1: An example of a math question involving symbolic reasoning.

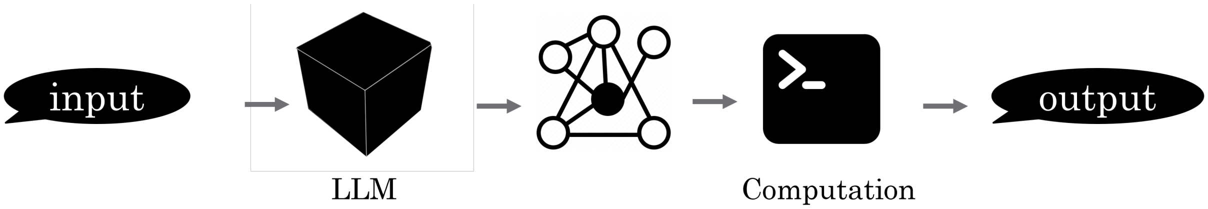


Figure 5.2: An illustration of my proposed pipeline leveraging a symbolic representation of the world pertaining to the problem. The representation is inferred by an LLM and fed to a computational tool such as an algorithm.

can often extract the numerals and even come up with the correct formula, it systematically fails at arithmetic calculation[35] (Lyu et al., 2023). To improve, one may attempt to work on LLMs that can better perform arithmetic calculations, or simply use a calculator to reliably reach the final answer (Figure 5.1), as long as the upstream LLM can provide a well-formed input.

In this thesis, I have demonstrated many ways to use LLMs to solve a problem (Figure 1.2), including an end-to-end usage (baseline), fine-tuning an LLM with structured data (Chapter 3), and prompting an LLM with structured data (Chapter 4). Following the idea above, I will now introduce a **neurosymbolic usage**, where an LLM no longer predict the final answer. Instead, its job ends

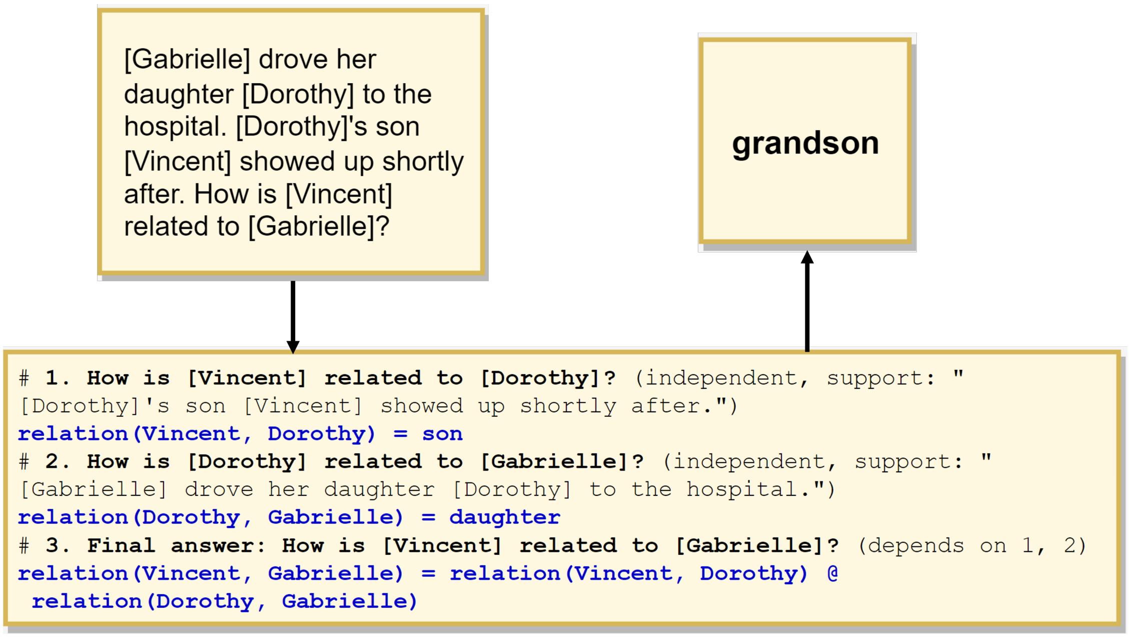
[35] Closed-source, commercial tools like OpenAI ChatGPT in 2024 is likely a pipeline including but not limited to one or more LLMs. It has been able to correct answer this type of math questions by translating it to Python code, similar to the ideas in my own, my collaborators', and other work that I will discuss next. In the paper, we continue to refer to LLMs as the end-to-end models themselves, not pipelines that involve them.

at predicting the structured representation, with which some external computation is responsible for reaching the final answer. This approach is akin to program synthesis (Austin et al., 2021) in the programming language community, using a generative model (e.g., LLM) to produce executable code. However, my work is practically scoped within problem solving and event reasoning, considering only specific representations for each downstream task.

5.1. Preliminary Experiments: Neurosymbolic Usage with LLMs

As introduced in Section 2.2.1, the ability to generate well-formed code is an emergent ability of LLMs (Feng et al., 2020; Chen et al., 2021a; Roziere et al., 2023). However, the neurosymbolic usage of LLMs, namely, using the generated code a means to perform reasoning tasks, is a new concept proposed by Lyu et al. (2023), a work that I secondarily contributed to, along with several contemporaneous work (Chen et al., 2022; Gao et al., 2023). I will fleetingly discuss this work.

In Lyu et al. (2023), I tackle the task of kinship deduction. Concretely, given a problem written in natural language, the answer is as a string-valued variable. I consider the CLUTRR (Sinha et al., 2019) dataset that involves inferring the family relationship (e.g., "grandson") between two people from a short story (e.g., "[Gabrielle] drove her daughter [Dorothy] to the hospital. [Dorothy]'s son [Vincent] showed up shortly after. How is [Vincent] related to [Gabrielle]?", Figure 5.3). This is a symbolic reasoning task where the performance of end-to-end LLMs drastically degrades as the complexity of the problem increases (i.e., more people are involved). In our neurosymbolic usage, we prompt the LM to generate Python code that essentially breaks down the question into sub-questions ("How is [Vincent] related to [Dorothy]" and "How is [Dorothy] related to [Gabrielle]"), as well as provide input extracts as rationales to support the answer ("[Dorothy]'s son [Vincent] showed up shortly after", etc.) as comments. The code for each sub-question is a relational expression representing the relation between the mentioned entities, for example, `relation(Vincent, Dorothy)=son` denotes that Vincent is Dorothy's son. Once we have the code, a fully defined symbolic representation of the problem, the LLM's job is finished. We then use a simple relational inference engine that relies on a set of transitivity rules provided by Zhang et al. (2022) among possible family relationships, e.g., `son@daughter=grandson` (the son of one's daughter is one's grandson). Our solver recursively



[Gabrielle] drove her daughter [Dorothy] to the hospital. [Dorothy]'s son [Vincent] showed up shortly after. How is [Vincent] related to [Gabrielle]?

grandson

```
# 1. How is [Vincent] related to [Dorothy]? (independent, support: "
[Dorothy]'s son [Vincent] showed up shortly after.")
relation(Vincent, Dorothy) = son
# 2. How is [Dorothy] related to [Gabrielle]? (independent, support: "
[Gabrielle] drove her daughter [Dorothy] to the hospital.")
relation(Dorothy, Gabrielle) = daughter
# 3. Final answer: How is [Vincent] related to [Gabrielle]? (depends on 1, 2)
relation(Vincent, Gabrielle) = relation(Vincent, Dorothy) @
 relation(Dorothy, Gabrielle)
```

Figure 5.3: An example from the CLUTRR dataset.

applies these rules on C_{SL} to derive A, and determine that Vincent is Gabrielle's grandson.

On the benchmark, I construct the prompt using 8 exemplars with $K \in \{2, 3\}$, where K is the number of intermediate steps, and test the models on the remaining examples with K up to 10. For `code-davinci-002`, I observe a 45.7% to 71.9% performance leap compared to the end-to-end usage when using the neurosymbolic usage. We can interpret this result from two angles. Regarding performance, the external solver relieves the burden of problem solving from the LLMs, without which the accuracy degrades by 19.4%. Regarding trustworthiness, due to the deterministic nature of the external solver, the answer is correct if and only if the interim representation (pairwise relationships) provided by the LLM is correct. Thus, a verifying agent (either human or automatic) may simply and locally check if each pairwise relationship is correctly extracted from the text. They may also trivially fix any error in the process. Such ability for a human user to interpret, verify, and correct the LLM output is a result of the neurosymbolic usage, not offered by the end-to-end usage.

At this point, I have introduced the neurosymbolic usage of LLMs and shown its preliminary success.

In the next section, I will show that the neurosymbolic usage is an indispensable tool to push the limit of structured reasoning of events, by attempting a historically challenging task: planning.

The work above was published in Lyu et al. (2023), in which I contributed to all experiments on kinship deduction.

Now that I have demonstrated the feasibility and appeal of the neurosymbolic usage of LLMs, I will return to the context of event-centric reasoning. In the next Section, I will explore the task of classical planning which instantiates said methodology. My proposal is to have LLMs generate not the solution, but a symbolic representation of the environment, namely a **world model** that can be executed by some external computation like a symbolic planner. I will next outline this methodology.

5.2. Planning

In this chapter, I will tackle the challenging task of planning that comes in two flavors: formal and informal.

5.2.1. Formal Planning

In the AI community, planning is the task of finding a sequence of actions to achieve a goal in a given environment (Fikes and Nilsson, 1971; LaValle, 2006). Over more than five decades of development, the task of planning has taken on many flavors. I will particular focus on the formulation of classical planning, where an initial and a goal configuration are defined as a collection of entity states. Each action gives rise to an event during which the states change in certain way. Therefore, the aim is to perform certain actions to drive the entity states in the environment to the goal configuration. Classical planning is both symbolic and formal, as all the involved concepts including entities, states, and actions are fully grounded to symbols (instead of natural language that we discussed in previous sections). Given a domain definition (including tuple of initial configuration, goal configuration, and actions), a plan can be found deterministically, if there is one. In other words, it does not require any inference or guesswork to find a plan.

The Planning Domain Definition Language (PDDL) (Aeronautiques et al., 1998) is a programming

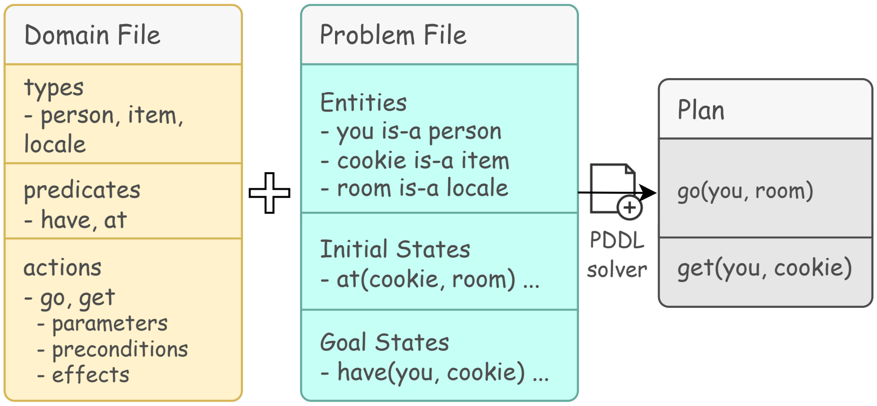


Figure 5.4: A PDDL solver produces a plan based on a minimal domain file and problem file. Previous work assumes the domain file as given, while we predict the action definitions in the domain file.

language designed for classical planning. A PDDL instance contains a domain file $\mathbb{DF}$ and a problem file $\mathbb{PF}$ (Figure 5.4.

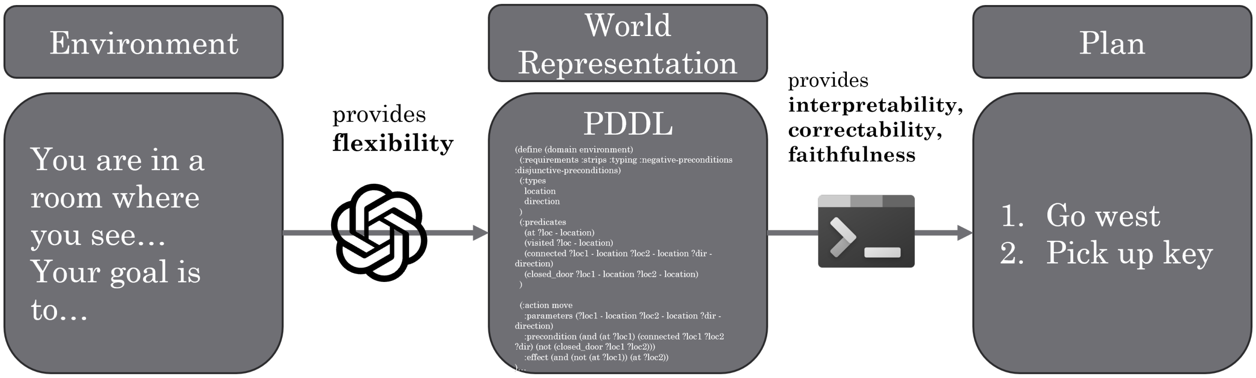
A $\mathbb{DF}$ defines the following elements:

- a header H, which consists of
  - types of entities (e.g., *object*, *location*, *player*)
  - predicates (e.g., if object is *at* a location)
  - names of possible actions (e.g., *boil water*)
- definitions of actions A, which consist of
  - parameters (e.g., water, pot) as a list of typed variables
  - preconditions (e.g., water and pot belongs to player; water is not treated) as a conjunctive normal form of predicates
  - effect (e.g., water is treated) as a conjunctive normal form of predicates

A $\mathbb{PF}$ defines the following elements:

- objects (e.g., rainwater)
- initial states (e.g., bucket is empty)
- goal states (e.g., bucket is filled with rainwater; rainwater is treated)

We say that a $\mathbb{DF}$ can *solve* a $\mathbb{PF}$ if there exists a sequence of actions $A_1, \ldots, A_n$ that results in a transition from the initial state to the goal state.



You are in a room where you see… Your goal is to…

```
(define (domain environment)
  (:requirements :strips :typing :negative-preconditions
:disjunctive-preconditions)
  (:types
    location
    direction
  )
  (:predicates
    (at ?loc - location)
    (visited ?loc - location)
    (connected ?loc1 - location ?loc2 - location ?dir -
direction)
    (closed_door ?loc1 - location ?loc2 - location)
  )

  (:action move
    :parameters (?loc1 - location ?loc2 - location ?dir -
direction)
    :precondition (and (at ?loc1) (connected ?loc1 ?loc2
?dir) (not (closed_door ?loc1 ?loc2)))
    :effect (and (not (at ?loc1)) (at ?loc2))
)...
```

Figure 5.5: The proposed world-building planning methodology that learns a formal representation of an informal description of the environment.

5.2.2. Informal Planning

However, there rarely exist formally defined, fully symbolic environments in the real life. Real-life environments are defined in many modalities; in the scope of NLP, we will focus on those defined using natural language texts. In Section 3.3, the task of script generation can also be thought of as planning, since a user would like to perform the steps and reach the goal. To make an analogy to PDDL, the $\mathbb{DF}$ is equivalent to a textual description or model's parametric knowledge of the environment, and the $\mathbb{PF}$ is equivalent to the goal and the inferred stats-quo. Because natural language is implicit and under-specified, such planning is informal as there are myriad undefined variables with regard to entities, states, and actions. The task of informal planning using natural language requires event reasoning to fill in these blanks. Ultimately, I believe this is also the most practical task to solve in the real world.

5.2.3. Neurosymbolic Method

For the task of planning with natural language, Section 3.3 has already shown the insufficiency of LLMs with both the end-to-usage and fine-tuning with relational structure. I now turn to a neurosymbolic methodology, where the setup of formal planning becomes a means to an end, introducing structure into the methodology. Concretely, my task is to **learn a structured domain definition** (i.e., $\mathbb{DF}$ and $\mathbb{PF}$) from unstructured natural language texts using LLMs. Once this is done correctly, a search-based planner can trivially find a plan, that may be presented to user after being converted to natural language or other modalities (Figure 5.5). In this methodology, an LLM's task is limited to **world-building**, coming up with a 'mental representation' (that is structured and formal) while interacting with a textual environment (that is unstructured and informal). The actual task of finding the plan is delegated to an external solver, much like Section 5.1.

As the output of planning is a sequence of actions, the task planning can be seen as an instance of event-centric reasoning. In my proposed framework of structured reasoning, the structured representation here, expressed by PDDL, is precisely an entity schema much like Section 4.1 and 4.2. Each action event is defined with preconditions and effects, both of which revolve around entities, or more precisely, their state changes. The process of modeling the actions is almost like the reverse of predicting entity states in a given procedure. The aim of the former is to *infer* an entity schema in order to *find* a plan. The aim of the latter is to *reconstruct* an entity schema from an *existing* plan in a post-hoc manner.

From a high level, the approach I will take for planning is a neurosymbolic usage of LLMs that generate a structured representation, or entity schema, or world model, given textual context. I will show that this approach is superior to end-to-end usage of LLMs when it comes to both performance and trustworthiness.

5.2.4. $\mathbb{DF}$ and $\mathbb{PF}$

Following the two-part design of PDDL, the *structured domain definition* or *world model* discussed above consist of two components: $\mathbb{DF}$ and $\mathbb{PF}$. The $\mathbb{PF}$, which specifies the initial and goal entities states, is much easier to learn from text (Lyu et al., 2023; Xie et al., 2023; Liu et al., 2023). For example, the text:

> Block A is on top of Block B, which is on the table.

needs to be translated into the following partial $\mathbb{PF}$:

```
on(A,B)
on_table(B)
```

On the other hand, learning the $\mathbb{DF}$ is much more challenging. The crux of the challenge lies in action modeling, in which much information must be inferred from the text. For example, the text:

> You pick the lock.

may correspond to the following partial $\mathbb{DF}$:

```
(:action pick-lock
    :parameters (?lock ?door ?room1 ?room2)
    :precondition (and
        (not (picked ?lock))
        (locked ?door)
        (not (accessible ?room1 ?room2))
    )
    :effect (and
        (picked ?lock)
        (not (locked ?door))
        (accessible ?room1 ?room2)
    )
)
```

Evidently, the action models in the $\mathbb{DF}$ are challenging to automatically generate, and require extensive manual effort to annotate for every domain. Much previous work tackles action modeling via plan traces (Yang et al., 2007; Cresswell et al., 2013; Lamanna et al., 2021), which is not our focus here. Few work that learns the $\mathbb{DF}$ from text (Yordanova and Kirste, 2016; Lindsay et al., 2017; Hayton et al., 2017, 2020; Jin et al., 2022; Li et al., 2023). These efforts have two issues. First, they involves complex pipeline approaches, many heavily templated and rule-based, and thus can unlikely generalize to diverse domains. Second, they have only targeted a static environment, performing a one-off translation or extraction from text to PDDL, much unlike the real world where an agent can interact with the environment. To overcome the first issue, LLMs with the newly gained ability to generate structured code seem like the tool of choice. However, it is known that LLMs struggle at generating low-resource domain specific languages (Jain et al., 2023) like PDDL.

In Section 5.3, I will discuss efforts to have LLMs generate correct and executable PDDL $\mathbb{DF}$ given complex text descriptions of an environment. To overcome the second issue, in Section 5.4, I target interactive textual environments where an agent not only learns their structured representation, but also continue to grow and refine said representation.

5.3. Planning for Procedural Text

A $\mathbb{DF}$ contains types, predicates, and actions (Figure 5.4). Like previous work, we specifically focus on action modeling of a $\mathbb{DF}$. Given types, predicate, and names of actions of an arbitrary domain and some textual description, we predict the preconditions and effects of each action to complete the domain definition. As a real-life motivating example, a kitchen robot may have access to cookwares and ingredients as well as the actions it can perform like "swinging a knife", but it still needs to predict the precondition that it is only safe to do so in front of a cutting board and the effect that the ingredients will become diced. Once the domain is completely defined, along with the problem definition $\mathbb{PF}$, an off-the-shelf solver can deterministically find a plan given a query.

Next, we provide some first attempts towards the action modeling task. As PDDL for any domain is extremely costly to annotate, there is hardly any data for training a model. Therefore, we demonstrate how our task can be tackled by zero-shot prompting state-of-the-art LMs.

Methodology To predict action definitions in $\mathbb{DF}$ based on the header and some text $\mathbb{T}$, we prompt an LM in a zero-shot manner Brown et al. (2020) by describing the task and providing the input. Note that few-shot prompting is prohibitively costly due to the excessive length of a resulting $\mathbb{DF}$.

We also incorporate the chain-of-thought (CoT) technique Wei et al. (2022b) that explicitly prompts an LM to perform three essential sub-tasks:

- Summarization: describe each action, including the expected preconditions and effects;
- Extraction: list the involved entities and their states before and after;
- Translation: based on the information above, convert $\mathbb{T}$ to PDDL.

We experiment with two large LMs, `gpt-3.5-turbo-16k` and `gpt-4` dated June 2023 with max

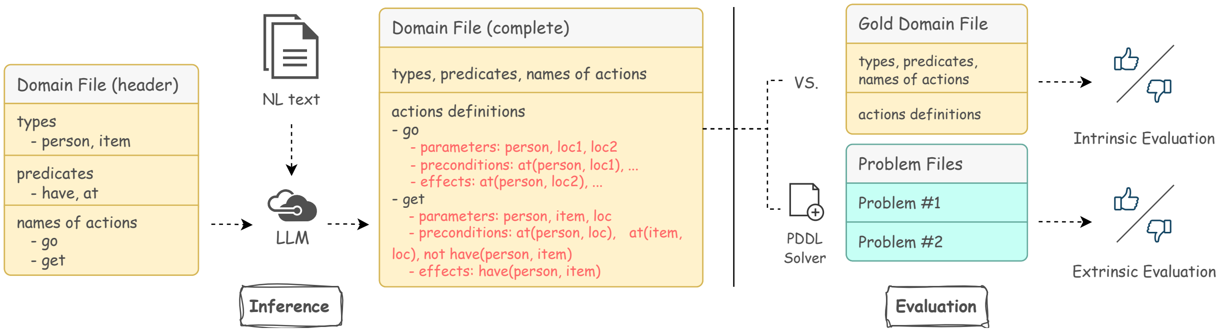

Domain File (header)
types
- person, item
predicates
- have, at
names of actions
- go
- get
NL text
LLM
Inference
Domain File (complete)
types, predicates, names of actions
actions definitions
- go
- parameters: person, loc1, loc2
- preconditions: at(person, loc1), ...
- effects: at(person, loc2), ...
- get
- parameters: person, item, loc
- preconditions: at(person, loc), at(item, loc), not have(person, item)
- effects: have(person, item)
vs.
PDDL Solver
Gold Domain File
types, predicates, names of actions
actions definitions
Problem Files
Problem #1
Problem #2
Evaluation
Intrinsic Evaluation
Extrinsic Evaluation

Figure 5.6: Our formulation of the $\mathbb{DF}$-action prediction task. Given the domain file header that specifies the ontology, a model predicts action definitions including parameters, preconditions, and effects based on textual descriptions. During evaluation, the predicted $\mathbb{DF}$ is both compared to a reference and used to solve corresponding $\mathbb{PF}$s.

tokens of 8192, temperature of 0.5, and default hyperparameters otherwise.

Dataset We introduce the PROC2PDDL dataset of 27 different $\mathbb{T}$-$\mathbb{DF}$-$\mathbb{PF}$s tuples, drawing procedural texts from wikiHow articles of various topics. A class of graduate students in a U.S. university with prior knowledge of PDDL are each given a wikiHow article and annotate a $\mathbb{DF}$ and multiple corresponding $\mathbb{PF}$s from the article, each with a gold plan to solve it. On average, there are 13.33 defined actions in a $\mathbb{DF}$ and 8.07 instantiated actions in a gold plan. During prediction, we treat $\mathbb{T}$ as an annotated one-line summary of all annotated actions.

On average, it takes several hours to train each human annotator, and another several hours to produce a $\mathbb{T}$-$\mathbb{DF}$-$\mathbb{PF}$s tuple. Upon post-inspection, we notice that about 15% are problematic and have to discard or fix them. This finding re-emphasize the great difficulty of annotating domain definitions and motivates our automatic prediction of $\mathbb{DF}$.

We partition the 27 examples into a 5:6:16 train-development-test splits. In this work, the train split is unused as all our methods are zero-shot; only the development set is used for error analysis; the test set is strictly held out for evaluation.

Evaluation Now that a model generates the parameters, preconditions, and effects for each action, we have a complete $\mathbb{DF}$. We evaluate it in two ways (Figure 5.6). *Intrinsically*, we semantically compare the predicted A with the ground-truth provided by our OPENPI2.0 and report an action-

	Intrinsic	Extrinsic	
Model %	action acc.	ℙ𝔽 solve	exact plan
`gpt-3.5`	0.2	1.0	1.0
`gpt-4`	15.9	33.7	4.2
`gpt-4` + CoT	**18.1**	**35.8**	**6.3**
gold	100.0	100.0	100.0

Table 5.1: Performance of the 𝔻𝔽-action prediction on the concatenation of the development and test set of OpenPI2.0. Metrics include action-wide accuracy, average edit distance of action definitions, the proportion of ℙ𝔽s that can be solved, and the proportion of generated plans that exactly match the gold plans.

wide accuracy. Equivalence of two action definitions does not depend on the naming of variables nor on the order within conjunctions. *Extrinsically*, to measure actions' coherence, we use a BFS-based PDDL solver[36] to attempt to solve ground-truth ℙ𝔽s with the predicted 𝔻𝔽 and report a success rate. An unsolved ℙ𝔽 is caused by (1.) no plan can be found, or (2.) the solver runs for more than 30 seconds, or (3.) the solver returns an error (usually a syntax error in generated PDDL).

The intrinsic and extrinsic results are shown in Table 5.1. `gpt-3.5-turbo`, also known as ChatGPT which achieves impressive performance on many tasks, has a close-to-zero performance. In contrast, its predecessor `gpt-4` performs significantly better with 18% action prediction accuracy and 36% solve rate of ℙ𝔽s based on the predicted 𝔻𝔽. Still, the performance far worse than ideal, showing that even a simplified open-domain planning formulation is challenging to state-of-the-art LMs.

CoT is helpful overall since it explicitly spells out many implicit entities and state changes in the extraction stage which are critical to predicting preconditions and effects. In most situations, the model summarizes the action and extracts the entity states correctly, though sometimes missing a few implicit entities. However, CoT's bottleneck lies in the translation stage, during which there are mainly three types of errors.

(i). mismatched predicates: the model uses `(at ?loc ?item)` instead of `(inventory ?item)`;

(ii). hallucinating predicates: the model creates a new predicate `(soaked ?item)` while neglecting the existing `(submerged ?item)`;

[36] https://github.com/pucrs-automated-planning/pddl-parser

Model %	Parameter	Precondition	Effect
`gpt-4`	36.7	31.1	53.0
`gpt-4` + CoT	42.2	29.7	48.1

Table 5.2: Marginalized intrinsic performance of parameter, precondition, and effect.

	Unsolved			Solved	
	Syntax Error	Bad Action	Good Action	Bad Plan	Good Plan
`gpt-4`	3	7	2	0	3

Table 5.3: Statistics of error types on the development set.

(iii). complicating predicates: the model adds unnecessary predicates `(inventory ?submerged_item - item)` when already has `(inventory ?item)`.

To address these, we leave to future work to demonstrate and standardize the translation process by clearly describing when an entity change or stage change is and is not needed, while also encouraging the model to strictly match the given predicates.

Finer-grained evaluation results are shown in Table 5.2 to tease out the performance regarding such component within an action. It is clear that the LM is worse at predicting preconditions than at predicting effects. This is understandable as procedural texts like wikiHow tend to be less explicit about predictions than about effects (e.g., from *bake for 10 minutes* it is obvious that the food will be baked, but it is unclear what state it had been in).

To provide deeper insights into model performance, we manually inspect the model output of `gpt-4` on all 6 examples (15 $\mathbb{PF}$s) in the development set. We consider the following scenarios.

Syntax Error: Model output may contain illegal expressions that cannot be parsed. For example, `(inventory ?player (clean ?strips))` is unacceptable because the arguments to a predicate must be atomic types, not another predicate.

Unsolved: Whenever the predicted $\mathbb{DF}$ cannot solve a $\mathbb{PF}$, we identify the first problematic action that differs with the ground-truth. For example, if the action `cut_plant` misses a critical effect of `(inventory ?player ?stalk)`, then other actions such as `graft_stalk` requiring it cannot be executed. At times, there could be false negatives where the predicted action definitions are in fact reasonable but still cannot lead to a solution.

Solved: The predicted $\mathbb{DF}$ may solve a $\mathbb{PF}$, but the plan may be different from the gold plan. It is naturally possible that the predicted plan is a fluke made possible by under-specified preconditions or over-exaggerated effects, as well as loopholes in the $\mathbb{PF}$ leading to unreasonable shortcuts. For the example in Figure 5.4, a model could *cheat* by defining the action `get` by not requiring the person and object to be in the same location; thus, the predicted plan would unreasonably omit the action `go`. However, at times, the predicted plan could also be a reasonable alternative.

The statistics of these errors on the development set is shown in Table 5.3. When no solution can be found, true negative is highly likely as the model indeed makes aforementioned mistakes during action prediction. When some solution is found, false positive is still possible as the predicted plan may be unreasonable. See attached materials for a complete error analysis of these examples. Our aforementioned future pipeline that separates summarization and translation would likely mitigate these errors.

Through our experiments, it is clear that LLMs are capable of generating PDDL, specifically $\mathbb{DF}$, and specifically action models, given noisy procedural texts. This is already a leap from previous work devising specialized tools for each domain, mostly narrative texts. However, the problem remains that the current text-to-PDDL world building process is one-off and will not work if the environment dynamically develops. In the next section, I will focus on interactive environments where world building happens during exploration.

The work above was published in Zhang et al. (2024b), in which I formulated the task and primarily contributed to the code to clean the dataset, run a PDDL planner, and perform evaluation. I also performed the majority of the analysis. I have obtained approval from all collaborators to exclusively include this work in this thesis.

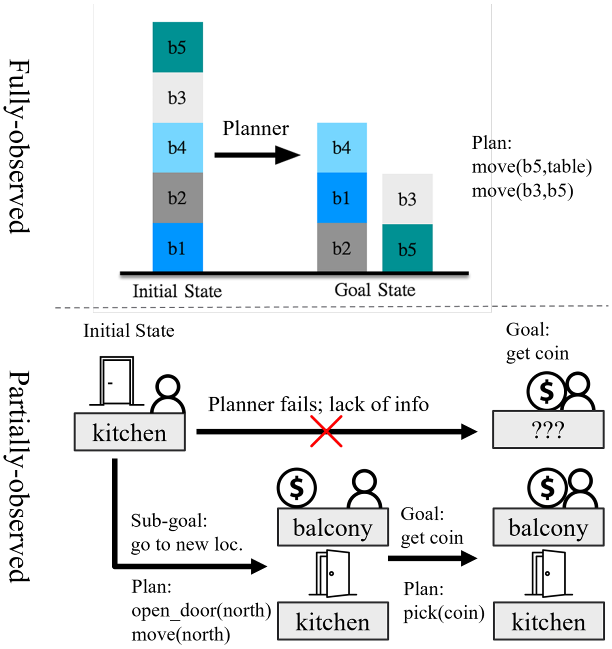


Figure 5.7: A fully-observed environment like BlocksWorld (upper, to rearrange objects from and to a given configuration) can be tackled by generating a PDDL problem file, while a partially observed one like Coin Collector (lower, to look for an object in an unknown location) cannot until sufficient exploration.

5.4. Planning for Interactive Environments

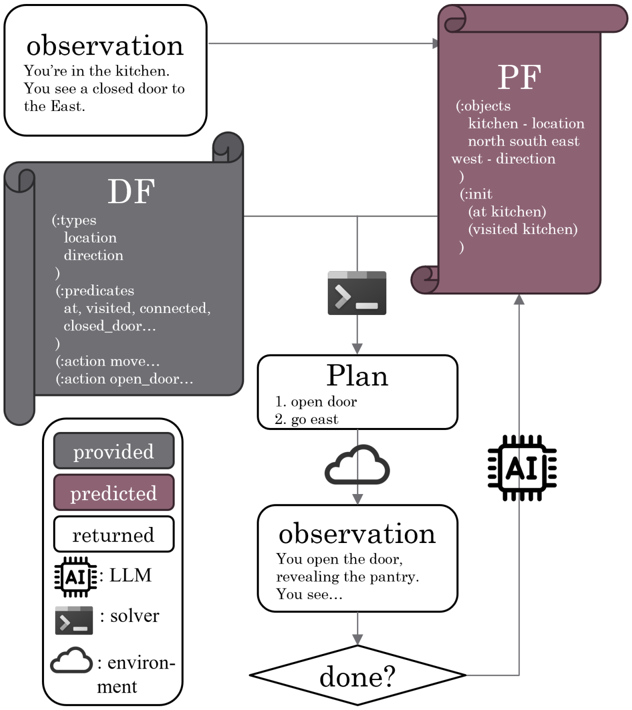
Motivation In Section 4.1 and Section 4.2, I have proposed and discussed event reasoning tasks that deal with a dynamically changing environment. I have shown that these tasks put forward unique challenges, unlike other tasks we have seen, for event state-of-the-art LLMs. I will now explore the same idea for planning, where it is not possible to derive a plan once-and-for-all. Instead, an agent must iteratively interact with the environment, gain new sights, make plans, adjust them, and eventually reach the goal.

All previous work on LLM generating PDDL has only experimented on **fully-observed** environments where all entity states are initially known. Take BlocksWorld as an example (Figure 5.7, upper), both the initial and goal positions of the blocks are spelled out, in which case existing work use LLMs to translate or parse the NL description of the world state into a PDDL $\mathbb{PF}$ (Lyu et al., 2023; Xie et al., 2023; Liu et al., 2023). With such a complete problem file, a full plan can be found and executed to reach the goal. In contrast, most real-world environments are **partially-observed** (Figure 5.7, lower), where the entity states dynamically get uncovered during exploration. Moreover, since the necessary initial and goal states might also be unknown (e.g., looking for an item without knowing where it is), the previous approach falls apart due to the impossibility to specify a complete problem file. This causes a chicken-and-egg problem where a plan is required for exploration, while exploration is required to build PDDL that results in a plan. Given this challenge, past work on partially-observed environments has only used LLMs to directly generate plans Shinn et al. (2023); Majumder et al. (2023) but not PDDL.

We propose PDDLEGO, a methodology to incrementally grow the PDDL by using LLMs to translate the NL observations from the environment into entity states expressed as a PDDL problem file. PDDLEGO solves the above stalemate of under-defined goal states by recursively falling back to a well-defined sub-goal. This way, a plan can be found to reach the sub-goal, leading to new observations obtained by exploring the environment. In this process, the problem file is incrementally built until a plan can be found for the final goal.

Methodology We operate in a partially-observed simulated environment which functions as a multi-turn interaction between the environment and the agent (e.g., *a game to find an item*). Specifically, the environment provides an NL observation (*objects in a room*) along with a list of permitted actions (*move, pick up*), then the agent selects on of these actions, and repeats. The environment can be seen as a finite state machine where each state consists of the conjunction of all entity states and determines the permitted actions. The agent succeeds when a goal state is reached (*the sought item is in hand*); it fails when it cannot possibly reach goal state.

Like prior work in LLM generating PDDL, we assume that a domain file that defines the available actions is provided; this domain file can solve a problem file that defines the initial and goal entity states (*where the agent is, where the item is, how are these two locations connected*) when possible to result in a plan (*go west, pick up item*). Unlike prior work, it is impossible to initially construct a complete problem file that is necessary for reaching the goal until the agent has explored enough to have uncovered all necessary entity states (*we don't know where the item is until we find it*).



```
(:objects
  kitchen - location
  north south east
west - direction
 )
 (:init
  (at kitchen)
  (visited kitchen)
 )
```

```
(:types
  location
  direction
)
(:predicates
  at, visited, connected,
  closed_door...
)
(:action move...
(:action open_door...
```

Figure 5.8: The pipeline of PDDLEGO. A PDDL problem file (PF) is incrementally built during exploration.

Therefore, our PDDLEGO assumes a plannable sub-goal for which partial plans can be found to make progress and iteratively build a problem file based on the new observations. Eventually, the problem is sufficiently defined and the final-goal can be planned for (Figure 5.8).

Formally, the agent is initially in state s_1 and presented with the first observation o_1. The agent then construct an initial problem file PF_1 to plan for the final-goal G.

$$PF_1 = \{LLM(o_1), G\} \tag{5.1}$$

If this problem file can be solved by the provided domain file with a solver, a plan containing one or more actions is found.

$$Plan_1 := (a_1^1, a_1^2, \dots) = solver(DF, PF_1) \tag{5.2}$$

If a plan cannot be found due to a lack of information in the problem file, the goal G is swapped out by an immediate sub-goal G', and the solver retries. The actions in the plan are then sequentially executed, resulting in a list of new observations.

$$(o_2^1, o_2^2, \dots) = exec(s_1, a_1^1, a_1^2, \dots) \tag{5.3}$$

Thus begins the second iteration. Using the new observations, the previous problem file is regenerated (referred to as **PDDL-gen**).

$$PF_2 = \{LLM(PF_1, o_2), G\} \tag{5.4}$$

The process goes on until one observation fulfills the termination condition.

Unlike prior work that generates the problem file once, PDDLEGO's having LLMs iteratively generating the problem file often result in inconsistencies and errors (e.g., missing a connectivity relation between two rooms, using the name a room in a relation without declaring the room, missing a parenthesis, etc.). To tackle this, we have the LLMs only predict the change in the problem file (i.e., the change of entity states), which we deterministically applied to the previous problem file (referred to as **PDDL-edit**).

$$\Delta_2 = LLM(PF_1, o_2),\ PF_2 = PF_1 + \Delta_2 \tag{4'}$$

We will compare our two approaches above with the baseline where LLMs directly generate an action (referred to as **Action-gen**).

$$Plan_i = LLM(o_i) \tag{2'}$$

Environments We experiment with two goal-oriented, partially-observed simulated environments, or text games, that span a variety of difficulty and flavor.

Coin Collector Yuan et al. (2019) focuses on navigation, which is an indispensable element of

	random	GPT 3.5 Turbo			GPT 4 Turbo		
		Action-gen	PDDL-gen†	PDDL-edit†	Action-gen	PDDL-gen†	PDDL-edit†
Coin	4%	68%	26%	28%	**94%**	58%	78%
Cooking (easy)	0%	0%	70%	68%	4%	94%	**98%**
Cooking (hard)	0%	0%	4%	6%	0%	16%	**46%**

Table 5.4: The percentage where the agent succeeds by taking no more than the maximum steps on the test set. The † sign specifies methods under our proposed PDDLEGO methodology.

most simulations. The agent's task is to explore rooms, some connected by locked doors, and find a coin, similar to the running example above. Just as previously discussed, the previous approach on generating a PDDL problem file cannot be applied to Coin Collector because the location of the coin is unknown until the agent enters the same room as the coin. Therefore, the sub-goal structure for this tasks is defined as:

(i). pick up coin (requires the location of the coin)

(ii). go to a room that has not been visited (reveals location of the coin)

The sub-goal of "going to an unvisited room" results in monotonously increasing progress to the final-goal of "finding the coin". In similar search-related tasks, this singular sub-goal or strategy suffices, though it may not work for all situations.

Cooking World Madotto et al. (2020) subsumes Coin Collector with more complex tasks. The agent' task is to first explore rooms to find ingredients required by a recipe, much like Coin Collector. Next, it should cook the ingredient in some specified location using some specified appliance. Finally, when all ingredients are cooked correctly, a meal can be successfully prepared. Therefore, the sub-goal structure for this tasks is defined as:

(i). prepare meal (requires having obtained each ingredient and located relevant appliances)

(ii). pick up each ingredient (requires the location of each ingredient; obtains ingredients)

(iii). go to a room that has not been visited (reveals location of ingredients and appliances)

To better understand these simulations, an example trajectory for Cooking World is shown in Figure 5.9.

```
< examine cookbook
> Gather all following ingredients and follow the directions to prepare this
    tasty meal.
Ingredients:
  black pepper, block of cheese, salt, red apple, yellow potato
Directions:
  slice the block of cheese, dice the red apple, chop the yellow potato, grill
      the yellow potato, prepare meal
< look around
> You are in the kitchen. In one part of the room you see a stove. There is
    also an oven. You also see a fridge that is closed. In another part of the
     room you see a counter that has a knife, a red apple, a raw yellow potato
    , and a cookbook on it. In one part of the room you see a kitchen cupboard
     that is closed. There is also a cutlery drawer that is closed. You also
    see a trash can that is closed. In another part of the room you see a
    dishwasher that is closed. In one part of the room you see a dining chair,
     that has nothing on it.
To the South you see a closed plain door. To the West you see the corridor.
< open fridge
> You open the fridge. The fridge contains a block of cheese.
< take block of cheese
> You take the block of cheese.
< take knife
> You take the knife.
< take red apple
> You take the red apple.
< take yellow potato
> You take the yellow potato.
< open kitchen cupboard
> You open the kitchen cupboard. It's empty inside.
< move west
> You are in the corridor. In one part of the room you see a key holder, that
    has nothing on it. There is also a shoe cabinet that is closed. You also
    see a umbrella stand, that has nothing on it. In another part of the room
    you see a hat rack, that has nothing on it. In one part of the room you
    see a coat hanger, that has nothing on it.
To the North you see a closed wood door. To the East you see the kitchen. To
    the West you see a closed sliding patio door.
< open shoe cabinet
> You open the shoe cabinet. It's empty inside.
< open door to north
> You open the wood door, revealing the bedroom.
< move north
> You are in the bedroom. In one part of the room you see a dressing table,
    that has nothing on it. There is also a desk chair, that has nothing on it
    . You also see a desk, that has nothing on it. In another part of the room
     you see a chest of drawers that is closed. In one part of the room you
    see a wardrobe that is closed. There is also a night stand, that has
    nothing on it. You also see a bed, that has nothing on it.
Through an open wood door, to the South you see the corridor.
```

```
< open chest of drawers
> You open the chest of drawers. It's empty inside.
< open wardrobe
> You open the wardrobe. It's empty inside.
< move south
> You are in the corridor. (description omitted)
< open door to west
> You open the sliding patio door, revealing the backyard.
< move west
> You are in the backyard. In one part of the room you see a barbeque. There
    is also a workbench, that has nothing on it. You also see a patio chair,
    that has nothing on it. In another part of the room you see a patio table,
     that has nothing on it. In one part of the room you see a clothes line,
    that has nothing on it. There is also a garden, that has nothing on it.
Through an open sliding patio door, to the East you see the corridor.
< move east < move east
> You are in the kitchen. (description omitted)
< open door to south
> You open the plain door, revealing the pantry.
< move south
> You are in the pantry. In one part of the room you see a folding chair, that
     has nothing on it. There is also a shelf that has some black pepper, and
    some salt on it.
Through an open plain door, to the North you see the kitchen.
< take black pepper
> You take the black pepper.
< take salt
> You take the salt.
< dice red apple
> You dice the red apple.
< slice block of cheese
> You slice the block of cheese.
< chop yellow potato
> You chop the yellow potato.
< move north
> You are in the kitchen. (description omitted)
Through an open plain door, to the South you see the pantry. To the West you
    see the corridor.
< move west < move west
> You are in the backyard. (description omitted)
< cook yellow potato in barbeque
> You grill the yellow potato with the barbeque.
< move east < move east
> You are in the kitchen. (description omitted)
< prepare meal
> Adding the meal to your inventory.
< eat meal
> You eat the meal.  It is delicious.
```

Figure 5.9: An example trajectory performed by GPT 4 Turbo and PDDL-edit on Cooking World (hard).

Evaluation For both simulations, we use the implementation from Jansen and Côté (2022). For Coin Collector, we use the most complex setting; for Cooking World, we consider an easy and a hard setting with varying number of locations and ingredients. For the choice of LLM, we consider `gpt-3.5-turbo-1106` (GPT 3.5 Turbo) and `gpt-4-1106-preview` (GPT 4 Turbo) across baseline methods (i.e., Action-gen, PDDL-gen, and PDDL-edit). For Action-gen, we prompt the LLM with a full description of the simulation, and for PDDL methods, with a hand-annotated domain file containing well-defined actions. For the PDDL-edit setting, we prompt the LLM to generate templated edits (add, replace, and delete lines in the problem file). The prompt of each method include a 1-shot demonstration of the output format.

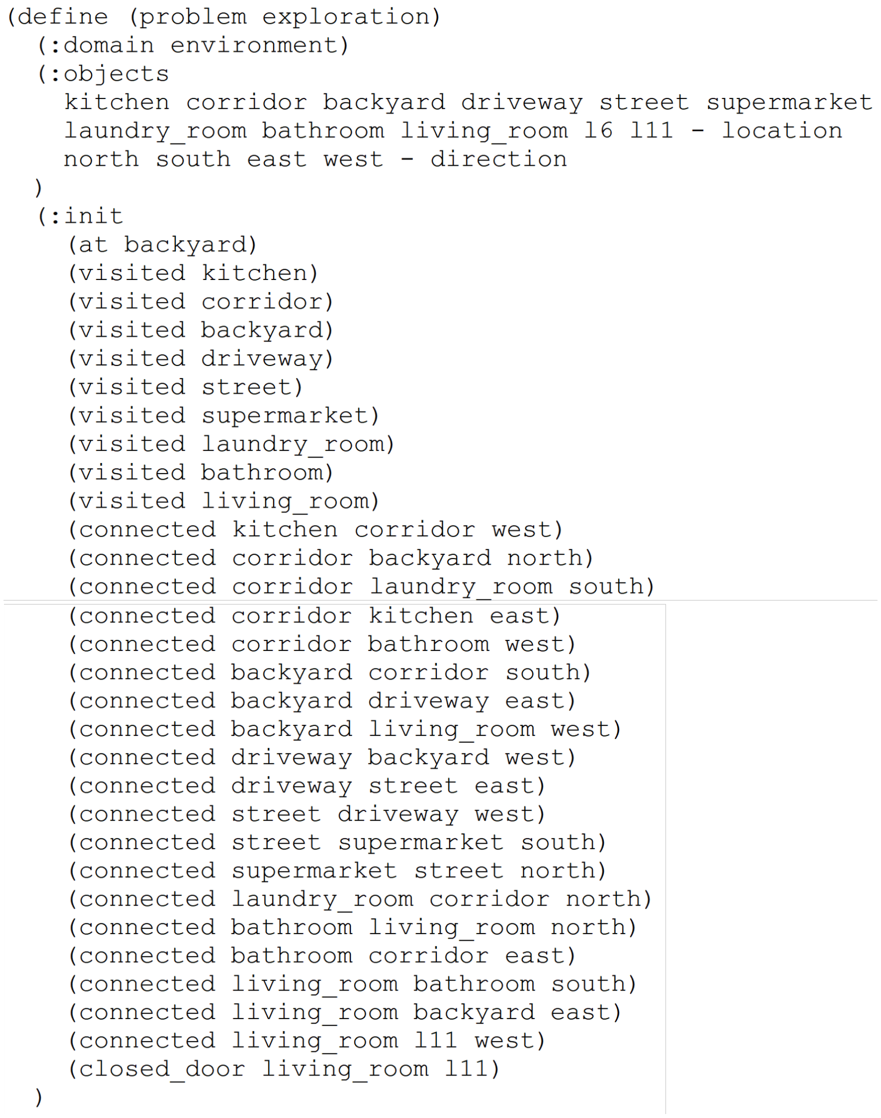
Regarding **performance**, Table 5.4 shows a drastic performance degradation of Action-gen moving from Coin Collector (only 2 valid actions: move, open door each with 4 direction arguments) to the much more complex Cooking World (with 8 more actions with infinite possible arguments, like processing an ingredient). Moreover, in Cooking World, an agent would fail if an ingredient is processed incorrectly (e.g., fried instead of grilled, was not chopped before roasted). Therefore, LLMs generating actions on the fly are more likely to make irrevocable mistakes and fail the task. In contrast, our two-stage PDDL generation approaches ensure the correctness of the plan to process the ingredients (in the second stage) *assuming* that the ingredients are gathered and that the appliances are identified (in the first stage). Logically, the failures of PDDLEGO indicates an inconsistency between the environmental observation and the problem file. For example, the connectivity of the rooms may not be updated correctly upon entrance to a new room, causing no plan or invalid plans to be found. By lessen the burden on LLMs, PDDL-edit notably ameliorates but cannot eliminate this issue. On Coin Collector, issues frequently arise in a loop, where opening a new door leads to a visited room. Notably, GPT3.5 is far worse than GPT4 in generating PDDL, in line with the observations by Zhang et al. (2024b) and Silver et al. (2023).

Regarding **efficiency**, Figure 5.11 shows that on Coin Collector, PDDL-edit is no less efficient than Action-gen on 7 out 8 examples (red crosses are often lower than the blue circles) in the development set where PDDL-edit terminates successfully. Scaling up to the entire test set, with GPT4, PDDL-edit has an average step to success of 7.8 compared to Action-gen's 13.6 among successful attempts, a 43% improvement on efficiency. Among these steps, 3.3 of Action-gen are invalid (e.g., moving through a closed door) compared to merely 0.2 of PDDLEGO, a significant difference when trials and errors are expensive. PDDLEGO also shows better **stability**. In Figure 5.11, PDDL-edit exhibits a much smaller variance across runs than Action-gen. For example, if the coin happens to be immediately to the west of the initial room, deciding to go west initially would result in a prompt success, while exploring the east portion initially would result in a notable detour. Our approach of PDDL generation leaves only the task of parsing environmental configuration to the LLM, while the planning task is done deterministically by the solver, leading to more consistent plans across runs.

Regarding **interpretability** and **correctability**, the black-box nature of LLMs results in no faithful interpretation behind the decisions (c.f., thought-process). In Coin Collector, for example, if the coin has not be found at the maximum permitted steps, a problematic Action-gen trajectory is almost impossible to manually correct unless a human is to plot a map and keep track of the exploration. On the other hand, both PDDL-gen and PDDL-edit guarantees the correctness of the plan assuming that the generated or edited problem file is correct. Hence, upon failure, a human only needs to inspect and correct the most recent observation and the PDDL. For PDDL-edit, the job is even easier as only the change in the problem needs to be considered. An example learned problem file can be found in Figure 5.10.

In conclusion, we propose PDDLEGO, the first approach to use LLMs to generate PDDL instead of actions while exploring partially-observed environments. We quantitatively show the improvement of performance, efficiency and stability, while qualitatively argue the benefit of interpretability and correctability by construction of this method. Future work should improve its flexibility by automatically predicting, instead of annotating, the domain file and the sub-goal hierarchy of each task, and show such flexibility on more involved simulations.

The work above was published in Zhang et al. (2024a), in which I formulated the task and primarily contributed to all components. I have obtained approval from all collaborators to exclusively include



```
(define (problem exploration)
  (:domain environment)
  (:objects
    kitchen corridor backyard driveway street supermarket
    laundry_room bathroom living_room l6 l11 - location
    north south east west - direction
  )
  (:init
    (at backyard)
    (visited kitchen)
    (visited corridor)
    (visited backyard)
    (visited driveway)
    (visited street)
    (visited supermarket)
    (visited laundry_room)
    (visited bathroom)
    (visited living_room)
    (connected kitchen corridor west)
    (connected corridor backyard north)
    (connected corridor laundry_room south)
    (connected corridor kitchen east)
    (connected corridor bathroom west)
    (connected backyard corridor south)
    (connected backyard driveway east)
    (connected backyard living_room west)
    (connected driveway backyard west)
    (connected driveway street east)
    (connected street driveway west)
    (connected street supermarket south)
    (connected supermarket street north)
    (connected laundry_room corridor north)
    (connected bathroom living_room north)
    (connected bathroom corridor east)
    (connected living_room bathroom south)
    (connected living_room backyard east)
    (connected living_room l11 west)
    (closed_door living_room l11)
  )
```

Figure 5.10: An example PDDL problem file learned throughout exploration in Coin Collector.

this work in this thesis.

5.5. Summary

In this chapter, I introduced a neurosymbolic approach to use LLMs to generate a structured representation, before that representation is executed by a symbolic solver. In the context of planning, LLMs are only in charge of world modeling but not deciding upon the actions. Compared to the semi-symbolic event-entity schema discussed in Chapter 4, this pipeline promises more determinism and interpretability by having the symbolic tools play a bigger role. For tasks where symbolic

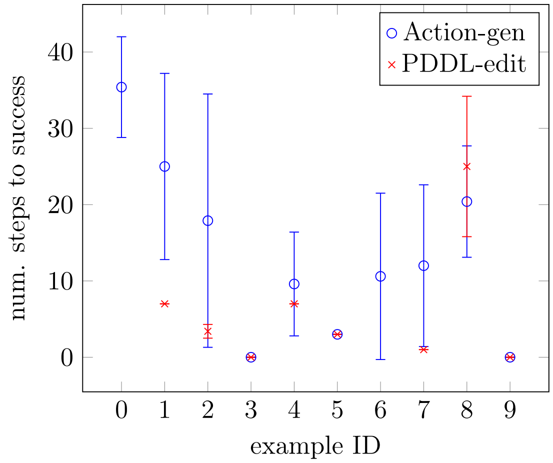


Figure 5.11: On Coin Collector, the mean and standard deviation of number of steps to success (less is better) for each development example, each over 5 trials with different random seeds of `gpt-4-1106-preview`, comparing Action-gen and PDDL-edit. The error bar represents the sample standard deviation. On example 0 and 6, PDDL-edit fails and thus not shown.

deduction plays a significant role, such as kinship deduction and classical planning discussed in this chapter, I have demonstrated how the synergy of LLMs and symbolic solvers lead to improved performance.

The ability of LLMs to generate executable structured representations (e.g., PDDL, Python) offers an exciting outlook for neurosymbolic methods as a whole that can tackle many tasks that include but are not limited to event reasoning. However, I have also shown significant challenges regarding the struggle of generating low-resource domain-specific languages (syntactically correct) as well as maintaining the consistency of the world representation during iterative generation (semantically correct). Furthermore, it awaits to be tested how well LLMs can generate such representation in a variety of domains.

CHAPTER 6

Conclusion

This thesis has focused on automatically reasoning about events, a line of work that spans many fronts of efforts in NLP including question answering, commonsense inference, symbolic reasoning, etc. I start by establishing that LLMs are suitable tools for event reasoning due to their strong ability of cross-domain adaptation. I then systematically explore LLMs' failure on a variety of tasks and datasets. To address this, I propose the general methodology to combine LLMs with some structured event representation. I introduce three types of such representation depending on various factors such as the task, the capability of the LLMs, and the need for flexibility versus determinism. Each of these representations is used as a component of a pipeline with models including but not limited to LLMs. Across a variety of tasks, I have demonstrated the benefit of the synergy between LLMs and structured representations. Notably are the following takeaways.

Event reasoning is semi-symbolic in nature. The choice to study the task of event reasoning in this thesis is practically motivated. However, event reasoning is representative of a superset of semi-language, semi-symbolic tasks (examples: the math question and relational inference tasks described in Section 5.1; non-examples: creative writing, summarization, emotion detection) that are ubiquitous and challenging in real-life scenarios. When using generative models (e.g., LLMs) for these tasks, we practitioners must decide the trade-off between language-modeling and symbol-modeling. This trade-off usually depends on the resources provided and the metric to optimize for. For example, to answer the question "*Is it safe to touch the pan?*" in Chapter 4, one may fully rely on an unstructured data representation and end-to-end LLMs if the LLMs are capable, the example is well-represented in the their training data, and stochasticity is acceptable. However, one may be better off decomposing the question into symbols as a multi-hop reasoning problem if the LLMs cannot directly answer the question, the example is out-of-distribution of the training data, and determinism must be guaranteed.

The long-tail problem thwarts fast-advancing models. The work discussed in this thesis was done over a span of five years (2019 to 2024) where models like LLMs have witnessed extraordinary growth in capability. Despite their improvement, the strength and weakness of these data-driven models have roughly remained constant: they become increasingly capable with problems that are well-represented in the training (or pre-training) data, but remain challenged by those that are not. This is known as the long-tail problem. In event reasoning and similar tasks, the long-tail problem can emerge in two ways. The first is the context-wise complexity. In Chapter 3, the questions are relatively short, generic (e.g., should one first *use the washer* or *user the dryer*)), and heavily represented in the training data, so that they are more likely to be solved by larger-scale LLMs. In Chapter 4, the questions become more specific (e.g., Is it *safe to touch the pan* after doing step A, B, and C), while in Chapter 5 even more so (e.g., find a plan to *retrieve a coin* given a configuration of a maze). For AI technology to be more useful for a variety of users and circumstances (especially those who are underrepresented), the long-tail problem should remain in focus. In this thesis, in the context of using LLMs, I have explored both the in-context end-to-end approach and the structured approach, empirically arguing the latter is superior in many scenarios.

Neurosymbolic methods are empowered by code-trained LLMs. My work described in Chapter 4 and 5 are novel and special cases of neurosymbolic methods because it leverages LLMs to consider and predict the symbols. As discussed in those chapters and related work, this has not been possible until approximately 2021 when LLMs are trained to work with structured data such as code. Historically, the advantage of neurosymbolic methods is their precision and determinism while the disadvantage is their brittleness across domains. Similarly, historically, the advantage of data-driven neural methods is their flexibility and the disadvantage is their inability to work with structured data. In the present day, neurosymbolic methods do not just compete with, but are empowered by code-trained LLMs as symbol generators and validators. In Chapter 5), the preliminary results show that this line of working is promising despite challenges.

The three methodologies introduced in this thesis each call for future work that includes examining more domains, integrating future LLMs, and so on. However, perhaps a even more crucial direction is to develop a "hierarchical organizer" that can reduce the amount of hard-coding when designing these pipelines. For example, a high-level switch may decide to use the neurosymbolic usage for a task where symbolic reasoning plays a significant role. Then, a mid-level switch may decide what to symbolically model and what to rely on end-to-end generation. Finally, a low-level switch may decide on implementation details, such as whether to include techniques like chain-of-thought or self-refine. Though ambitious, the prospect of hierarchical planning and problem solving will surely lead to AI systems' ability and versatility.

APPENDIX A

APPENDIX FOR CHAPTER 3

A.1. Quality Control Filters

As described in Section 3.1.1 and Section 3.1.2, we apply a collection of hand-crafted filters to the automatically generated examples to remove low-quality ones. The details of each filter are as follows:

Category filter: We remove examples involving articles under certain wikiHow categories. The categories we leave out are either too obscure (e.g. Astrology Relationships) or require expert domain knowledge to reason about (e.g. Car Engine Repairs), with the hope that the remaining categories contain more what we would call “common sense" knowledge that an average human has.

Lexical-Overlap filter: We remove examples where there is a lexical overlap between the prompt and each candidate. We exclude stopwords and lemmatize each word using spaCy before computing the overlap.

TF-IDF filter: We remove examples with overly uninformative prompts or candidates. We exploit TF-IDF as a proxy for how indicative a certain step is of the article it comes from. The motivation is that in Step Inference, for example, given a prompt step, the task is to choose its corresponding goal; then a prompt step like “gather your materials” is almost not informative at all for humans/models to tell which goal it serves, as a large number of articles may include a step like this. Thus, we treat each wikiHow article as a document and calculate the TF-IDF of each token, and only retain steps that have at least one token whose TF-IDF value surpasses a certain threshold.

Length filter: We remove examples with overly short prompts or candidates. The motivation is similar to that of the TF-IDF filter, i.e. too short goals/steps may not be informative enough to make a clean example. For example, steps like “Finished!” or “Serve!” are hard to tell apart if one of them is the positive candidate while the other is negative. To reduce such kind of noise in the automatically generated examples, we filter out steps/goals that are shorter than a specific threshold.

Similarity filter: We use similarity-based filters to remove examples where some negative candidate is also likely to be a plausible answer. The similarity scores are calculated using cosine similarity between BERT embeddings described in Section 3.1.1. In Step Inference, we set an upper threshold on the similarity between any negative step and any step from the prompt goal, with the motivation that negative steps should not serve the prompt goal. For Goal Inference, likewise, we ensure that the similarity between the prompt step and all steps from any negative goal is lower than a threshold, thus trying to minimize the cases where the prompt step also helps achieve negative goals.

A.2. More Open-Ended Examples

In addition to the examples in Section 3.1.6, we provide more open-ended examples for each task here.

A.2.1. Step Inference

For these open ended examples, our Step Inference model is trained in a 100-choose-1 format with 99 negative samples, instead of 4-choose-1, given 3 steps instead of 1. During evaluation, we use the softmax value in the final layer as the probability for each candidate. We rank the probabilities and report the top 3. Here are some more examples:

Input goal: Choose a Role Model
Predicted steps: learn about their successes and failures (correct), show interest in their lives, ask about their life

Input goal: End a Letter of Apology
Predicted steps: use a signature that conveys your emotions (correct), try to personalize the letter as much as possible, focus on the facts of the situation

A.2.2. Goal Inference

For Goal Inference, we follow the same procedure as above. Here are some more examples:

Input steps: buy or rent a good hammer drill, drill a pilot hole, insert a high quality masonry drill bit

Predicted goals: Drill Into Concrete (correct), Drill Holes Through Glass, Dig a Hole

Input steps: cultivate a memorable persona, keep an equal balance between your vlogging and your work life. review your channel

Predicted goals: Become a YouTube Guru (correct), Become a Film Buff, Become a Videographer

A.2.3. Step Ordering

For Step Ordering, the model can perfectly order the steps in many wikiHow articles unseen during training. To perfectly order an article, the model needs to correctly order all possible pairs of steps in an article. Here are 2 example articles with 10 steps:

Change Your Name of a Minor in Colorado: (1) make sure the child is eligible for a name change, (2) choose the right court, (3) download and review your forms, (4) get a fingerprint-based criminal background check, (5) complete the necessary forms, (6) get consent from the non-custodial parent, (7) file your petition with the appropriate court, (8) serve the non-custodial parent, (9) publish the proposed name change, (10) attend the hearing on your petition.

Draw a Simple Teddy Bear: (1) draw a circle for the teddy bear's head and an oblong for its body, (2) add two curved lines on each side of the oblong for the bear's arms, (3) draw two small circles below the oblong for the bear's feet, (4) add the ears using two small circles on each side of the head, (5) draw details of the face, (6) add details on the bear's pads using three small circles and a bean shape below it, (7) draw a shirt for the bear, (8) make the bear look furry by using small strokes in drawing its body, (9) erase unnecessary lines, (10) color the drawing.

A.3. Modeling Details of Intent Detection

After experimenting with base and large models, we use RoBERTa-large for the English datasets and XLM-RoBERTa-base for the multilingual dataset for best performances. All our models are implemented using the HuggingFace Transformer library[37].

[37] https://github.com/huggingface/transformers

We tune our model hyperparameters on the validation sets of the datasets we experiment with. However, in all cases, we use a unified setting which empirically performs well, using the Adam optimizer (Kingma and Ba, 2014) with an epsilon of $1e^{-8}$, a learning rate of $5e^{-6}$, maximum sequence length of 80 and 3 epochs. We variate the batch size from 2 to 16 according to the number of candidates in the multiple-choice task, to avoid running out of memory. We save the model every 1,000 training steps, and choose the model with the highest validation performance to be evaluated on the test set.

We run our experiments on an NVIDIA GeForce RTX 2080 Ti GPU, with half-precision floating point format (FP16) with O1 optimization. Each epoch takes up to 90 minutes in the most resource intensive setting, i.e. running a RoBERTa-large on around 100,000 training examples of our wikiHow pretraining dataset.

A.4. Modeling Details of Script Learning

All our models are implemented using the HuggingFace Transformer service[38]. For all experiments, we hold out 5% of the training data for development.

The pretrained models we use include: the `bert-base-multilingual-uncased` checkpoint (168M parameters) for mBERT, the `xlm-roberta-base` checkpoint (270M parameters) for XLM-RoBERTa, the `roberta-base` checkpoint (125M parameters) for RoBERTa[39], and the `mT5-Large` checkpoint (1B parameters) for mT5[40].

For mBERT, XLM-RoBERTa and RoBERTa, we finetune the pretrained models on our dataset using the standard `SequenceClassification` pipeline on HuggingFace[41]. For mT5, we refer to the official fine-tuning scripts[42] from the project's Github repository. We will release the fine-tuning source code for reproducing the results upon publication.

[38] https://github.com/huggingface/transformers

[39] The above 3 models are available at https://huggingface.co/transformers/pretrained_models.html

[40] https://github.com/google-research/multilingual-t5

[41] https://huggingface.co/transformers/model_doc/auto.html?highlight=sequence%20classification#transformers.AutoModelForSequenceClassification

[42] https://colab.research.google.com/github/google-research/text-to-text-transfer-transformer/blob/master/notebooks/t5-trivia.ipynb

For each in-domain evaluation experiment, we perform grid search on learning rate from $1e-5$ to $5e-8$, batch size from 16 to 128 whenever possible, and the number of epochs from 3 to 10. As mBERT and XLM-RoBERTa have a large number of hyperparameters, most of which remain default, we do not list them here. Instead, the hyperparameter values and pretrained models will be available publicly via HuggingFace model sharing. We choose the model with the highest validation performance to be evaluated on the test set. For the Retrieval setting, we consider the accuracy of contracted scripts; for the Generation setting, we consider perplexity.

We run our experiments on an NVIDIA GeForce RTX 2080 Ti GPU, with half-precision floating point format (FP16) with O1 optimization. The experiments in the Retrieval setting take 3 hours to 5 days in the worst case for all languages. The experiments in the Generation setting take 2 hours to 1 day in the worst case for all languages.

A.5. Crowdsourcing Details of Recursive Schema Construction

As discussed previously, we use Amazon Mechanical Turk (mTurk) to collect human judgements of linked wikiHow articles. Our mTurk task design HTML is attached in the supplementary materials. Each task includes an overview, examples of ratings, and 11 questions including 1 control question. Each question has the following prompt:

> Imagine you're reading an article about the goal `c_goal`, which includes a step `step`. Then, you're presented with a new article `r_goal`. Does this new article help explain how to do the step `step`?

where `c_goal` is the original corresponding goal of the `step`, and `r_goal` is the retrieved goal by the model. Both `c_goal` and `r_goal` have hyperlinks to the wikiHow article. The options of rating are:

1. The article explains exactly how to do the step.

2. The article is helpful, but it either doesn't have enough information or has too much unrelated information.

3. The article explains something related, but I don't think I can do the step with the instructions.

4. The article is unhelpful/unrelated.

5. I don't know which option to choose, because: [text entry box]

The control question contains either a `step` and `r_goal` with the exact same texts once lower-cased (in which case the expected answer is always #1), or a `step` and a randomly selected unrelated `r_goal` (in which case the expected answer is always #4). We estimate that answering each question would take 30 seconds, with a pay of $0.83 per task which equates to an hourly rate of $9.05. We require workers to be English-speaking, with the mTurk Master qualification and a lifetime approval rate of over 90%.

To sample examples to annotate, we first obtain all the steps corresponding to the same 1000 goals as previously discussed. To evaluate the DeBERTa-ul's ability to predict `unlinkable`, we randomly sample 500 steps predicted as `unlinkable` and another 500 predicted as otherwise. Then, for these 1000 steps, we obtain linked goal predictions of our three models: DeBERTa-ul, DeBERTa, and the SP model. If DeBERTa-ul predicts a step to be `unlinkable` by ranking the placeholder token first, the second ranked goal is instead considered. After removing duplicates of predicted step-goal pairs, we are left with 1448 examples.

When performing analyses, we only consider the responses from crowdworkers that pass more control questions than they fail.

BIBLIOGRAPHY

Andrea Addis and Daniel Borrajo. From unstructured web knowledge to plan descriptions. In *Information Retrieval and Mining in Distributed Environments*, 2011.

Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl| the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

Mohamed Ali. Character level convolutional neural network for Indo-Aryan language identification. In *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*, pages 283–287, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/W18-3932.

James Allan, Jaime G Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. 1998.

Farida Aouladomar. A preliminary analysis of the discursive and rhetorical structure of procedural texts. In *Symposium on the Exploration and Modeling of Meaning*. Citeseer, 2005.

Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013. doi: 10.1162/tacl_a_00209. URL https://aclanthology.org/Q13-1005.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Emmon Bach. The algebra of events. *Linguistics and philosophy*, pages 5–16, 1986.

Stephen H. Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged S. Al-shaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-Jian Jiang, and Alexander M. Rush. Promptsource: An integrated development environment and repository for natural language prompts, 2022. URL https://arxiv.org/abs/2202.01279.

Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998.

José Manuel Benítez, Juan Luis Castro, and Ignacio Requena. Are artificial neural networks black boxes? *IEEE Transactions on neural networks*, 8(5):1156–1164, 1997.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1159. URL https://aclanthology.org/D14-1159.

Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. Codekgc: Code language model for generative knowledge graph construction, 2023.

S Bielsa and M Donnell. Semantic functions in instructional texts: A comparison betw een english and spanish. In *Proceedings of the 2nd International Contrastive Linguistics Conference*, pages 723–732, 2002.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. *arXiv preprint arXiv:1711.05313*, 2017.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/P09-1010.

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yu Cao, Meng Fang, and Dacheng Tao. BAG: Bi-directional attention entity graph convolutional network for multi-hop reasoning question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 357–362, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1032. URL https://aclanthology.org/N19-1032.

Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://aclanthology.org/P08-1090.

Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/P09-1068.

Wei-Lun Chao, Hexiang Hu, and Fei Sha. Being negative but constructively: Lessons learnt from creating better visual question answering datasets. 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 431–441, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1040. URL https://aclanthology.org/N18-1040.

David L. Chen and Raymond J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 859–865. AAAI Press, 2011.

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

Muhao Chen, Hongming Zhang, Qiang Ning, Manling Li, Heng Ji, Kathleen McKeown, and Dan Roth. Event-centric natural language processing. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Tutorial Abstracts*, pages 6–14, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-tutorials.2. URL https://aclanthology.org/2021.acl-tutorials.2.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.91. URL https://aclanthology.org/2020.findings-emnlp.91.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6045–6049. IEEE, 2016.

Alexandr Chernov, Nikolaos Lagos, Matthias Gallé, and Ágnes Sándor. Enriching how-to guides by linking actionable phrases. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, page 939–944, Republic and Canton of Geneva, CHE,

2016. International World Wide Web Conferences Steering Committee. ISBN 9781450341448. doi: 10.1145/2872518.2890585. URL https://doi.org/10.1145/2872518.2890585.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. Distilling task knowledge from how-to communities. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 805–814. ACM, 2017. doi: 10.1145/3038912.3052715. URL https://doi.org/10.1145/3038912.3052715.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 25(5):564–577, 2003.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*, 2018.

Stephen N Cresswell, Thomas L McCluskey, and Margaret M West. Acquiring planning domain models using locm. *The Knowledge Engineering Review*, 28(2):195–213, 2013.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. 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 1595–1604, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1144. URL https://aclanthology.org/N18-1144.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. Everything happens for a reason: Discovering the purpose of actions in procedural text. 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 4496–4505, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1457. URL https://aclanthology.org/D19-1457.

Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei, editors. *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, Toronto, Canada, June 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.nlrse-1.0.

Donald Davidson. The logical form of action sentences. 1967.

Ernest Davis and Gary Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

Judith E Dayhoff and James M DeLeo. Artificial neural networks: opening the black box. *Cancer: Interdisciplinary International Journal of the American Cancer Society*, 91(S8):1615–1635, 2001.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. 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 2306–2317, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1240. URL https://aclanthology.org/N19-1240.

M. de Rijke, Harry Bunt, Jeroen Geertzen, and Elias Thijsse. Question answering: What's next? 2005.

Estelle Delpech and Patrick Saint-Dizier. Investigating the structure of procedural texts for answering how-to questions. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/20_paper.pdf.

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. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1259. URL https://aclanthology.org/P19-1259.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/5.pdf.

Milan Dojchinovski, Dinesh Reddy, Tomáš Kliegr, Tomáš Vitvar, and Harald Sack. Crowdsourced corpus with entity salience annotations. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3307–3311, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1527.

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. Aligning actions across recipe graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6930–6942, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.554. URL https://aclanthology.org/2021.emnlp-main.554.

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. doi: 10.18653/v1/P16-1004. URL https://aclanthology.org/P16-1004.

Yijiang River Dong, Lara J. Martin, and Chris Callison-Burch. Corrpus: Detecting story inconsistencies via codex-bootstrapped neurosymbolic reasoning, 2022.

Xinya Du and Claire Cardie. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.49. URL https://aclanthology.org/2020.emnlp-main.49.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. Be consistent! improving procedural text comprehension using label consistency. 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 2347–2356, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1244. URL https://aclanthology.org/N19-1244.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:

10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

Nan Duan, Duyu Tang, and Ming Zhou. Machine reasoning: Technology, dilemma and future. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 1–6, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-tutorials.1. URL https://aclanthology.org/2020.emnlp-tutorials.1.

Jesse Dunietz and Daniel Gillick. A new entity salience task with millions of training examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 205–209, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-4040. URL https://aclanthology.org/E14-4040.

Joe Ellis, Jeremy Getman, and Stephanie M Strassel. Overview of linguistic resources for the tac kbp 2014 evaluations: Planning, execution, and results. In *Proceedings of TAC KBP 2014 Workshop, National Institute of Standards and Technology*, pages 17–18, 2014.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8823–8838, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.710. URL https://aclanthology.org/2020.emnlp-main.710.

Edward A Feigenbaum, Avron Barr, and Paul R Cohen. The handbook of artificial intelligence. 1981.

Fuli Feng, Jizhi Zhang, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Empowering language understanding with counterfactual reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2226–2236, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.196. URL https://aclanthology.org/2021.findings-acl.196.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.139. URL https://aclanthology.org/2020.findings-emnlp.139.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. Zero-shot semantic parser for spoken language understanding. In *Sixteenth Annual Conference of the International Speech Communication*

Association, 2015a.

Emmanuel Ferreira, Bassam Jabaian, and Fabrice Lefèvre. Online adaptative zero-shot learning spoken language understanding using word-embedding. 04 2015b. doi: 10.1109/ICASSP.2015.7178987.

Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1033. URL https://aclanthology.org/P18-1033.

Kenneth D Forbus. Qualitative process theory. *Artificial intelligence*, 24(1-3):85–168, 1984.

Lea Frermann, Ivan Titov, and Manfred Pinkal. A hierarchical Bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1006. URL https://aclanthology.org/E14-1006.

Christian Fritz and Yolanda Gil. A formal framework for combining natural instruction and demonstration for end-user programming. In *Proceedings of the 16th international conference on intelligent user interfaces*, pages 237–246, 2011.

Michael Gamon, Tae Yano, Xinying Song, Johnson Apacible, and Patrick Pantel. Identifying salient entities in web pages. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2375–2380, 2013.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021. doi: 10.1162/tacl_a_00370. URL https://aclanthology.org/2021.tacl-1.21.

Yolanda Gil. Human tutorial instruction in the raw. *ACM Trans. Interact. Intell. Syst.*, 5(1), mar 2015. ISSN 2160-6455. doi: 10.1145/2531920. URL https://doi-org.proxy.library.upenn.edu/10.1145/2531920.

Yolanda Gil, Varun Ratnakar, and Christian Frtiz. Tellme: Learning procedures from tutorial instruction. In *Proceedings of the 16th international conference on intelligent user interfaces*, pages 227–236, 2011.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2118. URL https://aclanthology.org/N18-2118.

Mark Granroth-Wilding and Stephen Clark. What happens next? event prediction using a compositional neural network model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Aditya Gupta and Greg Durrett. Tracking discrete and continuous entity state for process understanding. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 7–12, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-1502. URL https://aclanthology.org/W19-1502.

Thomas Hayton, Julie Porteous, Joao Ferreira, Alan Lindsay, and Jonathon Read. Storyframer: From input stories to output planning models. In *Workshop on Knowledge Engineering for Planning and Scheduling (KEPS). The 27th International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

Thomas Hayton, Julie Porteous, Joao Ferreira, and Alan Lindsay. Narrative planning model acquisition from text summaries and descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1709–1716, 2020.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

James Higginbotham. On semantics. *Linguistic inquiry*, 16(4):547–593, 1985.

Pascal Hitzler and Md Kamruzzaman Sarker. Neuro-symbolic artificial intelligence: The state of the art. 2022.

Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, page 471–480, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584874. doi: 10.1145/1526709.1526773. URL https://doi-org.proxy.library.upenn.edu/10.1145/1526709.1526773.

Rijul Jain, Wode Ni, and Joshua Sunshine. Generating domain-specific programs for diagram authoring with large language models. In *Companion Proceedings of the 2023 ACM SIGPLAN*

International Conference on Systems, Programming, Languages, and Applications: Software for Humanity, SPLASH 2023, page 70–71, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400703843. doi: 10.1145/3618305.3623612. URL https://doi.org/10.1145/3618305.3623612.

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344, Avignon, France, April 2012. Association for Computational Linguistics. URL https://aclanthology.org/E12-1034.

Peter A. Jansen and Marc-Alexandre Côté. Textworldexpress: Simulating text games at one million steps per second. *arXiv*, 2022. URL https://arxiv.org/abs/2208.01174.

Kebing Jin, Huaixun Chen, and Hankz Hankui Zhuo. Text-based action-model acquisition for planning. *arXiv preprint arXiv:2202.08373*, 2022.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Yuchul Jung, Jihee Ryu, Kyung-min Kim, and Sung-Hyon Myaeng. Automatic construction of a large-scale situation ontology by mining how-to instructions from the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2-3):110–124, 2010.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, USA, 1st edition, 2000. ISBN 0130950696.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1114. URL https://aclanthology.org/D15-1114.

Junko Kimura and Russell Belk. Christmas in Japan: Globalization versus localization. *Consumption Markets & Culture*, 8(3):325–338, 2005.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Leila Kosseim and Guy Lapalme. Content and rhetorical status selection in instructional texts. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, 1994.

Leila Kosseim and Guy Lapalme. Choosing rhetorical structures to plan instructional texts. *Computational Intelligence*, 16(3):408–445, 2000. doi: https://doi.org/10.1111/0824-7935.00118. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/0824-7935.00118.

Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305, 2018. URL http://arxiv.org/abs/1810.09305.

Anjishnu Kumar, Pavankumar Reddy Muddireddy, Markus Dreyer, and Björn Hoffmeister. Zero-shot learning across heterogeneous overlapping domains. 2017.

Philippe Laban and Marti Hearst. newsLens: building and visualizing long-ranging news stories. In *Proceedings of the Events and Stories in the News Workshop*, pages 1–9, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2701. URL https://aclanthology.org/W17-2701.

Faisal Ladhak, Esin Durmus, Claire Cardie, and Kathleen McKeown. WikiLingua: A new benchmark dataset for cross-lingual abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4034–4048, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.360. URL https://aclanthology.org/2020.findings-emnlp.360.

Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, and Paolo Traverso. Online learning of action models for pddl planning. In *IJCAI*, pages 4112–4118, 2021.

Tessa A Lau, Clemens Drews, and Jeffrey Nichols. Interpreting written how-to instructions. In *IJCAI*, pages 1433–1438, 2009.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

David Lewis. *Counterfactuals*. John Wiley & Sons, 2013.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O Riedl. Crowdsourcing narrative intelligence. *Advances in Cognitive systems*, 2(1), 2012.

Manling Li, Alireza Zareian, Qi Zeng, Spencer Whitehead, Di Lu, Heng Ji, and Shih-Fu Chang. Cross-media structured common space for multimedia event extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2557–2568, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.230. URL https://aclanthology.org/2020.acl-main.230.

Manling Li, Sha Li, Zhenhailong Wang, Lifu Huang, Kyunghyun Cho, Heng Ji, Jiawei Han, and Clare Voss. The future is not one-dimensional: Complex event schema induction by graph modeling for event prediction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5203–5215, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.422. URL https://aclanthology.org/2021.emnlp-main.422.

Ruiqi Li, Leyang Cui, Songtuan Lin, and Patrik Haslum. Automated action model acquisition from narrative texts. *arXiv preprint arXiv:2307.10247*, 2023.

Angela Lin, Sudha Rao, Asli Celikyilmaz, Elnaz Nouri, Chris Brockett, Debadeepta Dey, and Bill Dolan. A recipe for creating multimodal aligned datasets for sequential tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4871–4884, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.440. URL https://aclanthology.org/2020.acl-main.440.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.713. URL https://aclanthology.org/2020.acl-main.713.

Alan Lindsay, Jonathon Read, Joao Ferreira, Thomas Hayton, Julie Porteous, and Peter Gregory. Framer: Planning models from natural language action descriptions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 27, pages 434–442, 2017.

Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling, 2016.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3?, 2021.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.deelio-1.10. URL https://aclanthology.org/2022.deelio-1.10.

Xiao Liu, Zhunchen Luo, and Heyan Huang. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1156. URL https://aclanthology.org/D18-1156.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1138. URL https://aclanthology.org/P16-1138.

Qing Lyu, Li Zhang, and Chris Callison-Burch. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 184–200, Aberdeen, Scotland, UK, August 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.inlg-1.19.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning, 2023.

Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification, 2019.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1015.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. *Def*, 2(6):4, 2006.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.90. URL https://aclanthology.org/2022.emnlp-main.90.

Andrea Madotto, Mahdi Namazifar, Joost Huizinga, Piero Molino, Adrien Ecoffet, Huaixiu Zheng, Alexandros Papangelis, Dian Yu, Chandra Khatri, and Gokhan Tur. Exploration based language learning for text-based games. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1488–1494. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/207. URL https://doi.org/10.24963/ijcai.2020/207. Main track.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60, Bilbao, Spain, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-2206. URL https://aclanthology.org/W15-2206.

Claudia Maienborn, Klaus von Heusinger, and Paul Portner. *Semantics: An international handbook of natural language meaning*, volume 1. Walter de Gruyter, 2011.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization, 2023.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1048.

John McCarthy. Programs with common sense, 1959.

Chris Mellish and Roger Evans. Natural language generation from plans. *Computational Linguistics*, 15(4):233–249, 1989.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

LA Miller. Natural language procedures: Guides for programming language design. In *International Ergonomics Association Meeting, University of Maryland*, volume 1, page 76, 1976.

Mayank Mishra, Prince Kumar, Riyaz Bhat, Rudra Murthy, Danish Contractor, and Srikanth Tamilselvam. Prompting with pseudo-code instructions. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15178–15197, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.939. URL https://aclanthology.org/2023.emnlp-main.939.

Ashutosh Modi and Ivan Titov. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1606. URL https://aclanthology.org/W14-1606.

Yoshio Momouchi. Control structures for actions in procedural texts and PT-chart. In *COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics*, 1980. URL https://aclanthology.org/C80-1016.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://aclanthology.org/N16-1098.

Nasrin Mostafazadeh, Lucy Vanderwende, Wen-tau Yih, Pushmeet Kohli, and James Allen. Story cloze evaluator: Vector space representation evaluation by predicting what happens next. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 24–29, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi: 10.18653/v

1/W16-2505. URL https://aclanthology.org/W16-2505.

Alexander PD Mourelatos. Events, processes, and states. *Linguistics and philosophy*, 2:415–434, 1978.

Dena Mujtaba and Nihar Mahapatra. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424, 2019a. doi: 10.1109/CSCI49370.2019.00082.

Dena Mujtaba and Nihar Mahapatra. Recent trends in natural language understanding for procedural knowledge. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 420–424. IEEE, 2019b.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4007. URL https://aclanthology.org/W19-4007.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In Stefan Riezler and Yoav Goldberg, editors, *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1028. URL https://aclanthology.org/K16-1028.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1206. URL https://aclanthology.org/D18-1206.

Dai Quoc Nguyen, Dat Quoc Nguyen, Cuong Xuan Chu, Stefan Thater, and Manfred Pinkal. Sequence to sequence learning for event prediction. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 37–42, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://aclanthology.org/I17-2007.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.441. URL https://aclanthology.org/2020.acl-main.441.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Simon Ostermann, Michael Roth, Stefan Thater, and Manfred Pinkal. Aligning script events with narrative texts. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 128–134, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-1016. URL https://aclanthology.org/S17-1016.

Artemis Panagopoulou, Manni Arora, Li Zhang, Dimitri Cugini, Weiqiu You, Yue Yang, Liyang Zhou, Yuxuan Wang, Zhaoyi Hou, Alyssa Hwang, Lara Martin, Sherry Shi, Chris Callison-Burch, and Mark Yatskar. Quakerbot: A household dialog system powered by large language models. In *Alexa Prize TaskBot Challenge Proceedings*, 2022. URL https://www.amazon.science/alexa-prize/proceedings/quakerbot-a-household-dialog-system-powered-by-large-language-models.

Paolo Pareti. Representation and execution of human know-how on the web. 2018.

Paolo Pareti, Ewan Klein, and Adam Barker. A semantic web of know-how: Linked data for community-centric tasks. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 1011–1016, New York, NY, USA, 2014a. Association for Computing Machinery. ISBN 9781450327459. doi: 10.1145/2567948.2578846. URL https://doi-org.proxy.library.upenn.edu/10.1145/2567948.2578846.

Paolo Pareti, Benoit Testu, Ryutaro Ichise, Ewan Klein, and Adam Barker. Integrating know-how into the linked data cloud. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 385–396. Springer, 2014b.

Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. A support tool for writing multilingual instructions. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1398–1404. LAWRENCE ERLBAUM ASSOCIATES LTD, 1995.

Cécile Paris, Keith Vander Linden, and Shijian Lu. Automated knowledge acquisition for instructional text generation. In *Proceedings of the 20th Annual International Conference on Computer Documentation*, SIGDOC '02, page 142–151, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135432. doi: 10.1145/584955.584977. URL https://doi.org/10.1145/584955.584977.

Cécile Paris, Nathalie Colineau, Lu Shijian, and Keith Vander Linden. Automatically generating effective online help, 2005. URL https://go-gale-com.proxy.library.upenn.edu/ps/i.do?p=ITOF&u=upenn_main&id=GALE%7CA132555479&v=2.1&it=r&sid=summon.

Hogun Park and Hamid Reza Motahari Nezhad. Learning procedures from text: Codifying how-to procedures in deep neural networks. In *Companion Proceedings of the The Web Conference 2018*, pages 351–358, 2018.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Basic Books, Inc., USA, 1st edition, 2018. ISBN 046509760X.

Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, page 573–582, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 158113844X. doi: 10.1145/988672.988750. URL https://doi-org.proxy.library.upenn.edu/10.1145/988672.988750.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Karl Pichotta and Raymond Mooney. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1024. URL https://aclanthology.org/E14-1024.

Karl Pichotta and Raymond Mooney. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16, Austin, TX, November 2016a. Association for Computational Linguistics. doi: 10.18653/v1/W16-6003. URL https://aclanthology.org/W16-6003.

Karl Pichotta and Raymond J. Mooney. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi: 10.18653/v1/P16-1027. URL https://aclanthology.org/P16-1027.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-2023. URL https://aclanthology.org/S18-2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Dheeraj Rajagopal, Niket Tandon, Peter Clark, Bhavana Dalvi, and Eduard Hovy. What-if I ask you to explain: Explaining the effects of perturbations in procedural text. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3345–3355, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.300. URL https://aclanthology.org/2020.findings-emnlp.300.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2124. URL https://aclanthology.org/P18-2124.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Schema-guided dialogue state tracking task at dstc8. *arXiv preprint arXiv:2002.01359*, 2020a.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696, 2020b.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-1100.

F. Ren and S. Xue. Intention detection based on siamese neural network with triplet loss. *IEEE Access*, 8:82242–82254, 2020.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006.

Lance J Rips. Reasoning. *Annual review of psychology*, 41(1):321–353, 1990.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code, 2023.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1195. URL https://aclanthology.org/D15-1195.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic, November 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.184. URL https://aclanthology.org/2021.findings-emnlp.184.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021b.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022b.

Kailash Karthik Saravanakumar, Miguel Ballesteros, Muthu Kumar Chandrasekaran, and Kathleen McKeown. Event-driven news stream clustering using entity-aware contextual embeddings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2330–2340, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.198. URL https://aclanthology.org/2021.eacl-main.198.

Roger C. Schank. *Scripts, plans, goals, and understanding : an inquiry into human knowledge structures /*. L. Erlbaum Associates ;, Hillsdale, N.J. :, 1977. ISBN 978-0-470-99033-9.

Sebastian Schuster, Sonal Gupta, Rushin Shah, and Mike Lewis. Cross-lingual transfer learning for multilingual task oriented dialog. 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 3795–3805, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1380. URL https://aclanthology.org/N19-1380.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. What does bert learn from multiple-choice reading comprehension datasets? *ArXiv*, abs/1910.12391, 2019.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models, 2023.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Push Singh, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237. Springer, 2002.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. 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 4506–4515, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1458. URL https://aclanthology.org/D19-1458.

Samuel L Smith, Andrew Brock, Leonard Berrada, and Soham De. Convnets match vision transformers at scale. *arXiv preprint arXiv:2310.16764*, 2023.

Evangelia Spiliopoulou, Artidoro Pagnoni, Yonatan Bisk, and Eduard Hovy. EvEntS ReaLM: Event reasoning of entity states via language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1982–1997, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.129. URL https://aclanthology.org/2022.emnlp-main.129.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam

Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Mineki Takechi, Takenobu Tokunaga, Yuji Matsumoto, and Hozumi Tanaka. Feature selection in categorizing procedural expressions. In *Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages*, pages 49–56, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1118935.1118942. URL https://aclanthology.org/W03-1107.

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. doi: 10.18653/v1/N18-1059. URL https://aclanthology.org/N18-1059.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. WIQA: A dataset for "what if..." reasoning over procedural text. 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 6076–6085, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1629. URL https://aclanthology.org/D19-1629.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.520. URL https://aclanthology.org/2020.emnlp-main.520.

Carol Tenny and James Pustejovsky. A history of events in linguistic theory. *Events as grammatical objects*, 32:3–37, 2000.

Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR, 2020.

Mokanarangan Thayaparan, Marco Valentino, Viktor Schlegel, and André Freitas. Identifying supporting facts for multi-hop question answering with document graph networks. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 42–51, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5306. URL https://aclanthology.org/D19-5306.

Scott M Thede and Mary Harper. A second-order hidden markov model for part-of-speech tagging. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 175–182, 1999.

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

Salvatore Trani, Claudio Lucchese, Raffaele Perego, David E Losada, Diego Ceccarelli, and Salvatore Orlando. Sel: A unified algorithm for salient entity linking. *Computational Intelligence*, 34(1): 2–29, 2018.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, and Thomas Rist. Plan-based integration of natural language and graphics generation. *Artificial intelligence*, 63 (1-2):387–427, 1993.

Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775.

Xingyao Wang, Sha Li, and Heng Ji. Code4Struct: Code generation for few-shot event structure prediction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3640–3663, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.202. URL https://aclanthology.org/2023.acl-long.202.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3494–3501, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://aclanthology.org/L16-1556.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018. doi: 10.1162/tacl_a_00021. URL https://aclanthology.org/Q18-1021.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Chuan Wu, Evangelos Kanoulas, Maarten de Rijke, and Wei Lu. WN-salience: A corpus of news articles with entity salience annotations. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2095–2102, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.257.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models, 2023.

P. Xu and R. Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 78–83, 2013.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. MEGATRON-CNTRL: Controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.226. URL https://aclanthology.org/2020.emnlp-main.226.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1166. URL https://aclanthology.org/D18-1166.

Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107–143, 2007.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1522. URL https://aclanthology.org/P19-1522.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. Visual goal-step inference using wikiHow. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2167–2179, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.165. URL https://aclanthology.org/2021.emnlp-main.165.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019b.

Majid Yazdani and James Henderson. A model of zero-shot learning of spoken language understanding. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 244–249, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1027. URL https://aclanthology.org/D15-1027.

Kristina Y Yordanova and Thomas Kirste. Learning models of human behaviour from textual instructions. In *ICAART (2)*, pages 415–422, 2016.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. Counting to explore and generalize in text-based games, 2019.

Zhongqi Yue, Tan Wang, Qianru Sun, Xian-Sheng Hua, and Hanwang Zhang. Counterfactual zero-shot and open-set visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15404–15414, 2021.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1009. URL https://aclanthology.org/D18-1009.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019a.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019b. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip Yu. Joint slot filling and intent detection via capsule neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5259–5267, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1519. URL https://aclanthology.org/P19-1519.

Hanlin Zhang, Ziyang Li, Jiani Huang, Mayur Naik, and Eric Xing. Improved logical reasoning of language models via differentiable symbolic programming. In *First Workshop on Pre-training: Perspectives, Pitfalls, and Paths Forward at ICML 2022*, 2022. URL https://openreview.net/forum?id=8lNy3QCaxHX.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.119. URL https://aclanthology.org/2020.emnlp-main.119.

Li Zhang. Reasoning about procedures with natural language processing: A tutorial, 2022. URL https://arxiv.org/abs/2205.07455.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Intent detection with WikiHow. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 328–333, Suzhou, China, December 2020b. Association for Computational Linguistics. URL https://aclanthology.org/2020.aacl-main.35.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online, November 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.374. URL https://aclanthology.org/2020.emnlp-main.374.

Li Zhang, Liam Dugan, Hainiu Xu, and Chris Callison-burch. Exploring the curious case of code prompts. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 9–17, Toronto, Canada, June 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.nlrse-1.2. URL https://aclanthology.org/2023.nlrse-1.2.

Li Zhang, Hainiu Xu, Abhinav Kommula, Niket Tandon, and Chris Callison-Burch. Openpi2. 0: An improved dataset for entity tracking in texts. *arXiv preprint arXiv:2305.14603*, 2023b.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. Causal reasoning of entities and events in procedural texts. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia, May 2023c. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-eacl.31. URL https://aclanthology.org/2023.findings-eacl.31.

Li Zhang, Peter Jansen, Tianyi Zhang, Peter Clark, Chris Callison-Burch, and Niket Tandon. Pddlego: Iterative planning in textual environments. In *Proceedings of the 13th Joint Conference on Lexical and Computational Semantics (*SEM 2024)*, Mexico City, Mexico, June 2024a. Association for Computational Linguistics.

Tianran Zhang, Muhao Chen, and Alex AT Bui. Diagnostic prediction with sequence-of-sets representation learning for clinical events. In *Artificial Intelligence in Medicine: 18th International Conference on Artificial Intelligence in Medicine, AIME 2020, Minneapolis, MN, USA, August 25–28, 2020, Proceedings 18*, pages 348–358. Springer, 2020d.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019b.

Tianyi Zhang, Li Zhang, Zhaoyi Hou, Ziyu Wang, Yuling Gu, Peter Clark, Chris Callison-Burch, and Niket Tandon. Proc2pddl: Open-domain planning representations from texts, 2024b.

Z. Zhang, Z. Zhang, H. Chen, and Z. Zhang. A joint learning framework with bert for spoken language understanding. *IEEE Access*, 7:168849–168858, 2019.

Ziqi Zhang, Philip Webster, Victoria Uren, Andrea Varga, and Fabio Ciravegna. Automatically extracting procedural knowledge from instructional texts using natural language processing. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 520–527, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/244_Paper.pdf.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Ben Zhou, Qiang Ning, Daniel Khashabi, and Dan Roth. Temporal common sense acquisition with minimal supervision. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7579–7589, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.678. URL https://aclanthology.org/2020.acl-main.678.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.214. URL https://aclanthology.org/2022.acl-long.214.

Yilun Zhou, Julie Shah, and Steven Schockaert. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56, Macau, China, August 2019. Association for Computational Linguistics. URL https://aclanthology.org/W19-5808.