Causal Reasoning About Entities and Events in Procedural Texts

Li Zhang[♣,*] **Hainiu Xu**[♣,*] **Yue Yang**[♣], **Shuyan Zhou**[♠],
Weiqiu You[♣], **Manni Arora**[♣], **Chris Callison-Burch**[♣]
[♣]University of Pennsylvania [♠]Carnegie Mellon University
{zharry,seacow,yueyang1,weiqiuy,manni,ccb}@seas.upenn.edu
{shuyanzh}@cs.cmu.edu

Abstract

Entities and events have long been regarded as the crux of machine reasoning. Procedural texts have received increasing attention due to the dynamic nature of involved entities and events. Existing work has focused either on entity state tracking (e.g., *the temperature of a pan*) or on counterfactual event reasoning (e.g., *how likely am I to burn myself by touching the pan*), while these two tasks are tightly intertwined. In this work, we propose CREPE, the first benchmark on causal reasoning about event plausibility based on entity states. We experiment with strong large language models and show that most models, including GPT3, perform close to chance at .30 F1, lagging far behind the human performance of .87 F1. Inspired by the finding that structured representations such as programming languages benefit event reasoning as a prompt to code language models such as Codex, we creatively inject the causal relations between entities and events through intermediate variables and boost the performance to .67 to .72 F1. Our proposed event representation not only allows for knowledge injection but also marks the first successful attempt of chain-of-thought reasoning with code language models.[1][2]

1 Introduction

Event-centric natural language processing (Chen et al., 2021b) is one of the leading paradigms in machine understanding of texts. This line of work focuses on first extracting entities and events from texts (Yang et al., 2019; Du and Cardie, 2020), and then reason about them (Li et al., 2020; Du et al., 2021). Even with the recent advances of large language models (LLMs), reasoning about events remains challenging as it requires highly contextual information and ample common-sense knowledge.

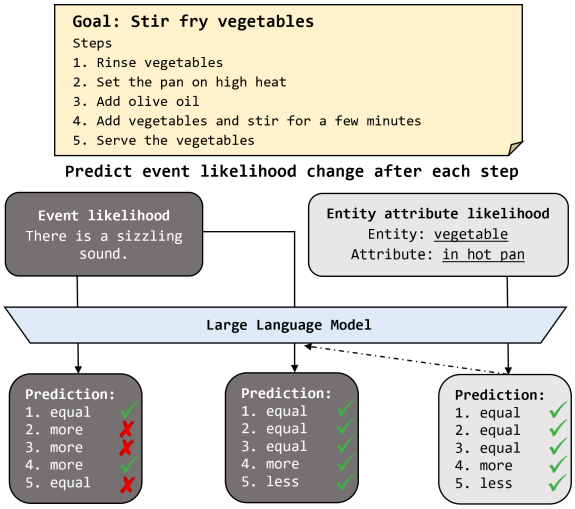

Figure 1: Example of our task CREPE. A procedure including a goal and some steps are provided. A model needs to predict the change in the likelihood of an event throughout the procedure. We show that predicting entity states as an intermediate step improves performance.

For example, given the context that "*hot oil is in the pan*", it is likely that "*adding water to the pan makes a sizzling sound*," while "*heat up an empty pan*" does not. Any model that can draw the correct conclusion given these contexts is expected to have access to some implicit knowledge about these entities and events (Zellers et al., 2018).

One type of text which demonstrates these challenges is procedural text, namely sequences of events, such as how-to instructions, recipes, scientific protocols, etc. Procedural texts describe an environment that changes dynamically through a sequence of steps. Therefore, the exact environment configuration is often implicit. In the previous cooking example, whether "*adding water to the pan makes a sizzling sound*" depends on what steps have taken place. With these interesting challenges coupled with the added benefit of application to robotics (Brohan et al., 2022) and household smart assistants such as Alexa (Panagopoulou et al., 2022), reasoning about procedures has received

[*]Equal contribution.
[1]Accepted to Findings of EACL 2023.
[2]Data and code can be found at https://github.com/zharry29/causal_reasoning_of_entities_and_events.

great attention from the NLP community (Zhang, 2022).

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., 2020a; 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 "*adding water to the pan makes a sizzling sound*" is entailed by whether "*the pan is very hot*." Because of this entailment, we claim that many event prediction tasks are multi-hop 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) 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, infer their states based on the current context, and finally make predictions about an 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 1. 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. We provide a handcrafted, high-quality benchmark containing 183 procedures, 1219 steps, and 324 changes in the likelihood of events along with the corresponding underlying entity state changes. In an in-context learning setting, we show that most LLMs including GPT3 (Brown et al., 2020) perform no better (.350 F1) than chance (.297 F1), greatly underperforming the human performance of .868 F1. Naively providing ground-truth entity state changes to GPT3 shows no performance gain, indicating that it cannot leverage this causal signal. Instead, we draw inspiration from Madaan et al. (2022) who represented texts as programming languages and used the code language model Codex (Chen et al., 2021a) to perform event reasoning. We propose a novel Python code representation of procedures that achieves .585 F1. Furthermore, our code-like representation allows us to effectively encode and leverage predicted or labeled entity state changes by generating them as an intermediate process (namely, chain-of-thought), boosting the performance to .667 using predicted entity state changes and .715 F1 using labeled entity state changes.

Our contributions are summarized as follows:

- We propose a novel task, a dataset, and several strong baselines for causal reasoning about events and entities in procedural texts.
- We devise an effective code-like representation of procedures, leading to superior performance and allowing the injection of structured knowledge for reasoning.
- We are among the first to show that code language models can apply chain-of-thought to tackle multi-hop reasoning.

2 Task and Hypothesis

A procedure P of length n consists of a goal G and some steps $s_1 \dots s_n \in S$, each represented as a short sentence. Each procedure is associated with a set of hypothetical events $e_1 \dots e_m \in E$ 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):

$$\delta_i = \text{pr}(e_j|s_i, \dots, s_1, G) - \text{pr}(e_j|s_{i-1}, \dots, s_1, G)$$

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

Predicting the likelihood of hypothetical events, also known as counterfactual reasoning, is extremely important for machine reasoning (Pearl and Mackenzie, 2018) (see more in Section 7). 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 $a_1 \dots a_m \in A$.

$$\delta_i = \text{pr}(a_j|s_i, \dots, s_1, G) - \text{pr}(a_j|s_{i-1}, \dots, s_1, G)$$

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

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 1: Statistics of the CREPE dataset.

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 pair of entity and attribute and its 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.

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 annotation instructions are attached as Supplementary Materials. To protect the privacy of the annotators, their personal identifiable information is anonymized in the released CREPE dataset.

The statistics of the development and test are in Table 1. 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.

4 Event Likelihood Prediction

CREPE can be seen as a ternary classification task, where the likelihood change of each event after each step is labeled as one of "more likely", "less likely", or "equally likely". In this section, all models have no access to the annotated entity attribute changes until later sections.

4.1 Baselines

To show the challenge CREPE brings to existing models, we first consider 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 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 (see Appendix for details), we compare the model's probability of generating "the answer is no|yes." We pick the `T0-3B`[3] of 3 billion parameters. The inference is performed in a zero-shot manner with no parameter update.
- **T0** (Sanh et al., 2022) 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 choose the `T0pp`[4] checkpoint, which has 11 billion parameters.
- **GPT3** (Brown et al., 2020) is an LLM that excels at few-shot learning using the prompting mechanism. We use the second largest model `text-curie-001`[5] with 7 billion parameters. We use default parameters with a temperature of 0 for deterministic predictions. An example of the prompt is shown in Figure 2.
- **GPT3 finetuned on StrategyQA** is the GPT3 above finetuned with StrategyQA (Geva et al., 2021), a dataset of factual multi-hop questions

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

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

[5] We find `text-davinci-002` to perform no significantly better and thus do not consider it due to high costs. See Table 2 for the performance of `text-davinci-002`.

```
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 2: Our GPT3 prompt, which is typical for a question-answering task. Each likelihood label is compared with the previous one to get the label for the change.

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 5.1). Details about prompt formulation are shown in Appendix B.

Table 2 shows that all state-of-the-art LLMs we have attempted to the best of our ability achieve close-to-chance performance on CREPE. We have also conducted experiments on the effect of the number of in-context examples and the prompt sensitivity (Appendix A).

4.2 Representing Procedures as Python Code

Codex (Chen et al., 2021a) is a variation of GPT3 that was designed to generate code given natural language instructions, while one contemporaneous work (Madaan et al., 2022) has found that Codex also works for NLP tasks by converting the text to some structured representation 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 the best empirical performance is described below, later shown to greatly outperform all baseline models. The representation is exemplified in Figure 3.

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` with 175B parameters and default hyperparameters with a temperature of 0.

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

4.3 Results

As CREPE is a ternary classification task, we set the evaluation metric as the macro F1 score across the three classes. As shown in Table 2, T5 and T0 perform only slightly better (.343 and .336 F1) than chance (.297 F1). GPT3, one of the most dominant models across a variety of NLP tasks, is no better (.336 F1), whereas finetuning it on another multi-hop reasoning dataset StrategyQA does not bring about any improvement (.341 F1).

On the other hand, creatively using the code LLM, Codex, with our proposed representation greatly outperforms all other models with .585 F1. As Codex is trained on public Github code in addition to the internet texts that GPT3 is trained on, it is pleasantly surprising that it can effectively reason about texts with code-like structures. In our case, a procedure has many analogies to a class in object-oriented programming.

4.4 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

	Naive		Large Language Models						Human
	Random	Majority	T5	T0	GPT3-C	GPT3-D	GPT3+S	Codex (ours)	
Params	-	-	3B	11B	13B	175B	13B	10B	-
Dev	.262	.297	.343	.336	.346	0.350	.341	**.585**	.868
Test	.251	.296	.343	.337	.356	0.533	.346	**.591**	-

Table 2: 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. GPT3-C represents the `text-curie-001` model. GPT3-D represents the `text-davinci-002` model. GPT3+S represents GPT3 `davinci` model finetuned on StrategyQA.

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

Table 3: Macro F1 of the ablations of our Codex prompt.

- Use flat variables to encode goals, steps, and events (no hierarchical class functions)

Examples of these empirically inferior representations are shown in Appendix B. As seen in Table 3, 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.

5 Causal Reasoning with 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.

5.1 Predicted Entity States as CoT

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 (Wei et al., 2022). Thus, we try to answer the following question:

```
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: Our GPT3 prompt with intermediate questions, mimicking the CoT prompt (top) and the Self-Ask prompt (bottom).

CoT with GPT3

CoT Codex with Soft Entity Representation To the best of our knowledge, we are the first to use Codex for CoT reasoning. We modify our Codex prompt in Figure 3, 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 5). During inference, Codex is provided with the code up to the step function header and predicts the entity and event changes for 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.

	Naive	LLMs		CoT Large Language Models				Human
	Majority	GPT3	Codex	GPT3 + CoT	GPT3+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: Macro F1 of chain-of-thought models on the CREPE dataset. GPT3 + CoT represents the `text-davinci-002` model prompted with the CoT style prompt.

```
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 5: Our Codex prompt with a soft representation of entity state changes as strings.

CoT Codex with Hard Entity Representation 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 6. 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 positive.

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

	Dev	Test
Majority	.297	.296
GPT3 CoT	.342	.345
w/ gold entity changes	.351	.380
Codex CoT	.667	.609
w/ gold entity changes	**.715**	**.722**
Human	.868	-

Table 5: Macro F1 of GPT3 and Codex with chain-of-thought provided with gold entity state changes.

5.2 Annotated Entity States as CoT

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. In this subsection, we try to answer the following

question:

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

Here, instead of having LLMs predict entity state changes, we provide the annotated entity state changes in the CREPE dataset to GPT3 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 5, our Codex representation with access to gold entity changes leads to improved performance of .715 F1 on the development set. In contrast, GPT3 does not see any gain. Hence, the answer to **Q2** is 'yes' for the code-trained LLMs but 'no' for standard LLMs.

5.3 Externally Predicted Entity States

As we will discuss further in Section 7, entity state tracking is an established task in NLP with a handful of 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] afterwards." While the original paper proposed a GPT2 model (Radford et al., 2019), we opt to finetune the superior GPT3 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 negative, suggesting that existing entity state tracking datasets may be insufficient for our causal reasoning task.

6 Performance 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) (see Appendix E).

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 F1 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.

7 Related Work

Event & Entity Extraction and Representation Event-centric NLP has been a dominant strand of approaches to machine reasoning. Myriad work has focused on extracting events from the news or web data (Liu et al., 2018; Yang et al., 2019; Du and Cardie, 2020). The effort of structurally representing scripts, groups of events in certain scenarios including procedures, started decades ago (Abelson and Schank, 1977) and is receiving revived attention in present years (Li et al., 2020;

Wang et al., 2022a). While this line of work mostly focuses on the representation as relations (e.g., temporal, hierarchical) among events, we recognize entities as a cause of event relations and thus propose a more granular representation. Furthermore, structured representations of events typically cannot take advantage of the power of textual LLMs for challenging downstream tasks. In contrast, we advance towards the best of two worlds by working with code language models.

Besides, existing work on jointly extracting and representing events and entities (Lee et al., 2012; Wadden et al., 2019; Barhom et al., 2019) neglects the causal relation therein and treats entities and events simply as two related tasks to be tackled simultaneously. We causally bridge the two.

Entity State Tracking 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 or an open-ended set Tandon et al. (2020). In all prior work, entity state track is treated as an end-task, whereas we treat it as a critical intermediate step for event reasoning, a more practical application.

Counterfactual Reasoning In this work, we hope to provide evidence that signals of entities effectively help models reason about events. We specifically focus on hypothetical event reasoning because it 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).

Multi-hop Reasoning Prior studies on multi-hop reasoning mainly focus on question answering from a passage (Welbl et al., 2018; Talmor and Berant, 2018; Yang et al., 2018; Kočiskỳ et al., 2018; Mihaylov et al., 2018; Khot et al., 2020) and representing and utilizing multi-hop information in the form of structured data (De Cao et al., 2019; Ding et al., 2019; Qiu et al., 2019; Cao et al., 2019; Fang et al., 2020; Thayaparan et al., 2019; Zhang et al., 2020b, 2021; Huang and Yang, 2021).

There are also efforts such as DecompRC, StrategyQA, and CGDe-FGIn that attempt to conduct multi-hop reasoning by decomposing the original task to a series of logically related sub-tasks (Min et al., 2019; Geva et al., 2021; Cao and Liu, 2022). Such an approach has recently seen great success with the Chain-of-Thought (CoT) prompting of GPT3, which significantly improves numerous multi-hop reasoning tasks (Kojima et al., 2022; Wei et al., 2022; Wang et al., 2022b). Following CoT prompting, Self-Ask further elicits CoT by demanding GPT3 to explicitly generate the reasoning questions raised during its chain-of-thought process (Press et al., 2022).

8 Conclusion and Future Work

We present CREPE, a benchmark for causal reasoning about events and entities in procedural texts. We show that mainstream LLMs such as GPT3 perform close to chance on CREPE, while using code-like event representation as a prompt to code language model Codex greatly improves the performance. Further, we experiment with various ways to encode entity information into this representation and find that eliciting chain-of-thought reasoning from Codex further improves performance while existing CoT approaches with GPT3 are ineffective. We clearly show that LLMs benefit from lower-level entity information when making predictions about higher-level events. Future work should explore related tasks such as next-event prediction, event temporal ordering, etc., by injecting relevant information about entities into our representation. Our code-representation of events allows more powerful expressions than simply entailment and negation considered in this work. Future work may explore other forms of code chain-of-thought such as first-order logic. These expressions generated by LLMs can be computed objectively, thus ameliorating LLMs' hallucinations and improving the interpretability and faithfulness of predictions.

9 Limitations

Despite our best efforts, our CREPE dataset has inherent limitations. First, the choice of studying procedure texts, despite many discussed advantages, limits the domain, writing style, and other semantic features of the texts. As a result, porting our meth-

ods and findings to other text styles such as stories or news might require domain adaptation. Second, we prioritize quality over quantity when creating this benchmark, which suffers from small size and contains biases from the annotators, even though we address the latter by having different annotators label a test set.

When annotating the hypothetical events, our intention is that they represent a wild variety that doers of the procedures, humans or machines, would care about. However, we also have to ensure these events are unambiguously bound to some entities in order to challenge models for their causal reasoning ability. While we do our utmost to balance these two conflicting objectives, the issue might still persist.

In CREPE, each event likelihood change is caused by exactly one entity state change. This is an over-simplification made to facilitate evaluation. In real life, many complex events require many entity states to be reasoned about, which in turn may have complex logical relations among them. We leave this for future work.

While we intend our representation of events and entities to be a general and effective one, we have only shown that it works well empirically using Codex, which is one of the only code language models at present. Whether the idea of our structured representation applies to other models remains to be explored.

10 Acknowledgements

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), the IARPA BETTER Program (contract 2019-19051600004), and the NSF (Award 1928631). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, IARPA, NSF, or the U.S. Government.

We thank the students in the Artificial Intelligence class at the University of Pennsylvania in Fall 2022 who participated in the annotation of the test set of CREPE. We thank Niket Tandon and Qing Lyu for valuable discussions about this work.

References

Robert Abelson and Roger C Schank. 1977. Scripts, plans, goals and understanding. *An inquiry into human knowledge structures New Jersey*, 10.

Shany Barhom, Vered Shwartz, Alon Eirew, Michael Bugert, Nils Reimers, and Ido Dagan. 2019. Revisiting joint modeling of cross-document entity and event coreference resolution. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4179–4189, Florence, Italy. Association for Computational Linguistics.

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

Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*.

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

Xing Cao and Yun Liu. 2022. Coarse-grained decomposition and fine-grained interaction for multi-hop question answering. *Journal of Intelligent Information Systems*, 58(1):21–41.

Yu Cao, Meng Fang, and Dacheng Tao. 2019. 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. Association for Computational Linguistics.

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. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Muhao Chen, Hongming Zhang, Qiang Ning, Manling Li, Heng Ji, Kathleen McKeown, and Dan Roth. 2021b. 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. Association for Computational Linguistics.

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

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. 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. Association for Computational Linguistics.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. 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. Association for Computational Linguistics.

Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. 2019. 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. Association for Computational Linguistics.

Li Du, Xiao Ding, Ting Liu, and Bing Qin. 2021. Learning event graph knowledge for abductive reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5181–5190, Online. Association for Computational Linguistics.

Xinya Du and Claire Cardie. 2020. 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. Association for Computational Linguistics.

Xinya Du, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. 2019. 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. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. 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. Association for Computational Linguistics.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2020. 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. Association for Computational Linguistics.

Fuli Feng, Jizhi Zhang, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Empowering language understanding with counterfactual reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2226–2236, Online. Association for Computational Linguistics.

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

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Aditya Gupta and Greg Durrett. 2019. 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. Association for Computational Linguistics.

Yongjie Huang and Meng Yang. 2021. Breadth first reasoning graph for multi-hop question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5810–5821.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500, Jeju Island, Korea. Association for Computational Linguistics.

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

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695, Online. Association for Computational Linguistics.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018. 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. Association for Computational Linguistics.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. 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. Association for Computational Linguistics.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Abu Dhabi, UAE.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. 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. Association for Computational Linguistics.

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. 2022. Quakerbot: A household dialog system powered by large language models. In *Alexa Prize TaskBot Challenge Proceedings*.

Judea Pearl. 2019. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60.

Judea Pearl and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*, 1st edition. Basic Books, Inc., USA.

Ofir Press, Sewon Min Muru Zhang, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models.

Lin Qiu, Yunxuan Xiao, Yanru Qu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6140–6150.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

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. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Alon Talmor and Jonathan Berant. 2018. 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. Association for Computational Linguistics.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. 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. Association for Computational Linguistics.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. 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. Association for Computational Linguistics.

Mokanarangan Thayaparan, Marco Valentino, Viktor Schlegel, and André Freitas. 2019. 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. Association for Computational Linguistics.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. 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 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Chieh-Chih Wang, Charles Thorpe, Sebastian Thrun, Martial Hebert, and Hugh Durrant-Whyte. 2007. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916.

Hongwei Wang, Zixuan Zhang, Sha Li, Jiawei Han, Yizhou Sun, Hanghang Tong, Joseph P Olive, and Heng Ji. 2022a. Schema-guided event graph completion. *arXiv preprint arXiv:2206.02921*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. 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. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. 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. Association for Computational Linguistics.

Zhongqi Yue, Tan Wang, Qianru Sun, Xian-Sheng Hua, and Hanwang Zhang. 2021. Counterfactual zero-shot and open-set visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15404–15414.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. 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. Association for Computational Linguistics.

Li Zhang. 2022. Reasoning about procedures with natural language processing: A tutorial.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020a. 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.

Min Zhang, Feng Li, Yang Wang, Zequn Zhang, Yanhai Zhou, and Xiaoyu Li. 2020b. Coarse and fine granularity graph reasoning for interpretable multi-hop question answering. *IEEE Access*, 8:56755–56765.

Yuyu Zhang, Ping Nie, Arun Ramamurthy, and Le Song. 2021. Answering any-hop open-domain questions with iterative document reranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 481–490.

A Primitive Experiments with GPT3

In addition to the results reported in Table 2, we also investigated (1) the effect of the number of in-context examples and (2) the prompt sensitivity.

Number of in-context examples The context window of `text-davinci-002` maximally fits 3 shots. We experiment with 1-shot (0.245 f1), 2-shots (0.348 f1), and 3-shots (0.359 f1) learning using text-002 with CoT prompting. We see that hav-

ing more context provides limited improvements in model performance.

Prompt sensitivity with random examples We tested the `text-davinci-002` model with CoT prompt on the dev set using randomly chosen examples from our example bank. The F1 scores for 5 runs with randomly chosen in-context examples are 0.333, 0.327, 0.359, 0.336, 0.331. The mean score is 0.337 and the standard deviation is 0.011, implying low sensitivity towards in-context example selection.

B Prompt Engineering

In Section 4 and 5, we have discussed our best-performing prompts for GPT3 and Codex. Here, we elaborate on inferior Codex prompts and shed light on why they do not work well empirically.

Best prompt As discussed, our best-performing prompt represents procedures as classes and steps as functions.

```
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.
```

Nested functions Instead of representing procedures as classes as in our best-performing prompt, we can also represent them as nested functions.

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

No step comments The comments displaying the steps immediately after the class declaration are removed.

```
class Wash_Sneakers:
  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.
```

No event comments The comments displaying the events in step functions except init are removed.

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

Two-step In this approach, we hypothesize that providing entity state change at every step is helpful. To do this, we first prompt Codex to generate entity states corresponding to a specific event:

```
class Wash_Sneakers:
    def remove_shoelaces(self):
        event = "My feet get wet by wearing
                 the sneakers."
        event.precondition = \
                     ("sneakers", "wet")
    def rinse(self):
        event = "My feet get wet by wearing
                 the sneakers."
        event.precondition = \
                     ("sneakers", "wet")
```

We select event-related entities by majority vote. The resulting entity state bank is used to prompt Codex to first deduce entity state at every step and then answer the likelihood of the event.

Flat variables Instead of defining functions using `def` or creating class with `class`, we use only variables to define relevant information.

```
Goal = "Wash Sneakers"

Context = "Remove shoelaces. After this,
           the shoelaces are removed"
Question = "What is the likelihood that my feet
           get wet by wearing the sneakers?
Options = [
           "more likely",
           "less likely",
           "equally likely",
          ]
Answer = Options[2]

Context = "Rinse the sneakers. After this,
          the sneakers are damp."
Question = "What is the likelihood that my feet
           get wet by wearing the sneakers?
Options = [
           "more likely",
           "less likely",
           "equally likely",
          ]
Answer = Options[0]
```

C GPT3 Finetuned on StrategyQA

For GPT-3 finetuned with StrategyQA, we ask two questions regarding the likelihood of the events, namely whether it is more/less likely that some event occurs. After obtaining the result, we conduct a consistency check. For consistent likelihood estimates, where only one of the two questions gives a positive answer, or both questions give negative answers, we assign the corresponding label to the event state change. For inconsistent estimates, where both questions give positive answers, we assign the event change likelihood to the majority label, which is "equally likely". An example of a finetuning prompt-completion pair is shown as follows

```
Prompt:
Context: Julius Caesar had three children.
Genghis Khan had sixteen children.
Modern geneticists have determined
that out of every 200 men today
has DNA that can be traced to
Genghis Khan.
Question: Are more people today
related to Genghis Khan than Julius Caesar?
Take it step by step:

Completion:
#1 How many kids did Julius Caesar have?
two
#2 How many kids did Genghis Khan have?
fourth
#3 Is fourth greater than two?
no
Therefore, the answer to the original
question is True
```

An example of our StrategyQA GPT-3 prompt on the CREPE task is as follows

```
Context: Remove shoelaces. Rinse. Srub the
shoes with cleaning solution. Rinse the shoes
again. Air dry the shoes and put the shoelaces
back on.
Question: Is it more likely that my feet get
wet by wearing the sneakers?
Take it step by step:

Completion:
#1 Is the sneaker wet?
Yes
#2 Will my feet get wet by wearing wet shoes?
Yes
Therefore, the answer to the original question
is True.
```

D T5/T0 Prompts

We design the following prompt for T5 and T0 to perform our task:

```
Goal: [The name of the goal]
Step: [The list of steps]
Question: Is that okay that [question]?
Answer: [yes or no, generated by the model]
```

E Error Analysis

In Section 6, we conclude that the performance of Codex is not influenced by (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.

Factors	Dev
Procedure Length > 7	.629
Procedure Length ≤ 7	.700
EoI Mentioned	.481
EoI NOT Mentioned	.496
Entailment	.482
Contradiction	.461

Table 6: Macro F1 Score of error analysis. The scores for EoI and Logical relation are lower since we do not consider the majority label, "equally likely", in the error analysis.