CAT-BENCH: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans

Yash Kumar Lal[1*], **Vanya Cohen**[2*], **Nathanael Chambers**[3], **Niranjan Balasubramanian**[1], **Raymond Mooney**[2]

[1]Stony Brook University, [2]University of Texas, Austin
[3]US Naval Academy
[1]{ylal,niranjan}@cs.stonybrook.edu, [2]{vanya,mooney}@utexas.edu,
[3]nchamber@usna.edu

Abstract

Understanding the abilities of LLMs to reason about natural language plans, such as instructional text and recipes, is critical to reliably using them in decision-making systems. A fundamental aspect of plans is the temporal order in which their steps need to be executed, which reflects the underlying causal dependencies between them. We introduce CAT-BENCH, a benchmark of Step Order Prediction questions, which test whether a step must necessarily occur before or after another in cooking recipe plans. We use this to evaluate how well frontier LLMs understand causal and temporal dependencies. We find that SOTA LLMs are underwhelming (best zero-shot is only 0.59 in F1), and are biased towards predicting dependence more often, perhaps relying on temporal order of steps as a heuristic. While prompting for explanations and using few-shot examples improve performance, the best F1 result is only 0.73. Further, human evaluation of explanations along with answer correctness show that, on average, humans do not agree with model reasoning. Surprisingly, we also find that explaining *after* answering leads to better performance than normal chain-of-thought prompting, and LLM answers are not consistent across questions about the same step pairs. Overall, results show that LLMs' ability to detect dependence between steps has significant room for improvement.

1 Introduction

Planning is central to decision making and has been studied in various domains such as robotics and embodied environments (LaValle, 2006; Jiang et al., 2019). To follow, revise, or customize a plan, one must be able to reason about the steps involved as well as their causes and effects (Brahman et al., 2023; Lal et al., 2024). Recent work on evaluating reasoning in plans focuses on classical problems

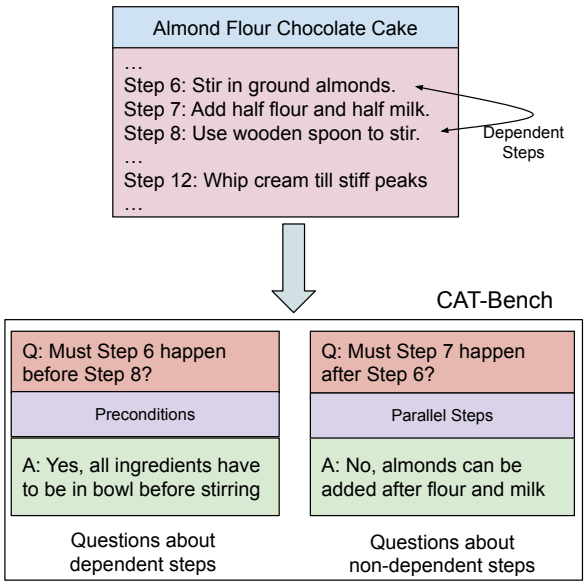


Figure 1: We use step-pair dependency annotations to create CAT-BENCH, a question-driven evaluation framework for plan-based reasoning. Questions in this benchmark elicit reasoning about different causal relations such as preconditions, effects and step independence.

such as Blocksworld (Slaney and Thiébaux, 2001; Valmeekam et al., 2023), simulated environments like AlfWorld (Shridhar et al., 2021), or restricted language such as PDDL (Zhang et al., 2024b). However, real-world natural language plans cannot be executed to test for correctness and reliability. This paper describes a new question-driven evaluation to better study the detailed causal and temporal connections within such plans.

Given a plan, such as making a cake in Figure 1, one must understand its various aspects to answer questions about it. Answering if ground almonds should be added before stirring the mixture requires understanding that a *precondition* for mixing evenly is that all ingredients should be added already. But reasoning about whether flour should be added after almonds requires figuring out *step independence* since the order of adding ingredients

[*]Equal Contribution

doesn't matter here. Such causal aspects are encoded in temporal step dependencies of plans. We modify the Recipe Flow Graph Corpus (Yamakata et al., 2020), containing recipes with substep procedure dependencies, to construct a new question-based dependency benchmark. CAT-BENCH contains 4260 questions about causal dependencies spanning 57 unique plans.

West et al. (2024) show that LLMs create expert-like outputs, but their generative capability does not necessarily indicate a correspondingly strong capability to understand underlying phenomena. While LLMs appear to generate good plans, it's unclear how well they understand important aspects of the steps themselves. We thus use CAT-BENCH to test whether LLMs can identify step dependencies that reflect the causal and temporal structure of the plan. We find that current LLMs struggle to identify step dependencies, often performing close to random chance, raising more questions about their understanding of instructional text.

Using notions of consistency to evaluate their robustness, we also show that almost all out-of-the-box LLMs are largely unreliable. Few-shot prompting with retrieved exemplars improves performance and consistency ($0.49 \rightarrow 0.68$ F1 for `gpt-4o`). Explanation-based generation offers another route to improve model performance and reliability on reasoning tasks (Camburu et al., 2018; Rajani et al., 2019; Kumar and Talukdar, 2020). Prompting LLMs to explain their decisions also improves performance on CAT-BENCH ($0.49 \rightarrow 0.7$ F1). Despite these gains, there is still a large room for improvement in identifying step dependencies. When also considering the quality of the explanations, the average human ratings for satisfactory answers from SOTA LLM's is only $\sim$3 (out of 5). Further, contrary to prior findings, using chain-of-thought prompting (CoT), i.e., reasoning before answering (Wei et al., 2022b), performs worse than answering first and then explaining it, indicating inconsistencies in model reasoning.[1]

In summary, this paper:

- Introduces CAT-BENCH, a benchmark to evaluate the causal and temporal reasoning abilities of LLMs over instructional plans.
- Demonstrates that current LLMs cannot predict causal dependencies in plans well, and highlights what aspects are most difficult.
- Evaluates explanations for correctness and as a prompting mechanism to improve reasoning.
- Analyzes successes and failures of LLMs, finding that generating a prediction followed by an explanation is significantly better than CoT.

[1] CAT-BENCH is available at https://huggingface.co/datasets/vanyacohen/CaT-Bench and the code is at https://github.com/StonyBrookNLP/CaT-Bench.

2 Related Work

Early work in text understanding argued for the importance of understanding plans and goals (Schank and Abelson, 1977). Generating plans (Aouladomar and Saint-Dizier, 2005) involves different types of understanding such as temporal reasoning and entity state tracking. NaturalPlan (Zheng et al., 2024) present real-world tasks with natural language interaction, but are only limited to three tasks. PlanBench (Valmeekam et al., 2023) showed that LLMs were unable to generate executable and effective plans, but focused on simulated worlds with restrictive PDDL syntax. Lyu et al. (2021) proposed the Goal-Oriented Script Construction task, where a model produces a sequence of steps (or a plan) to accomplish a given goal. ChattyChef (Le et al., 2023) uses the conversational setting to generate cooking instructions and iteratively refine its step ordering. CoPlan (Brahman et al., 2023) collects conditions associated with a revised list of steps for the task of plan revision to satisfy constraints. Lal et al. (2024) study the use of LLMs for plan customization according to user requirements. LLMs have been shown to generate plans well but it is unclear how well they truly understand all aspects of these plans.

Plan understanding tasks involve multiple aspects such as tracking entity states (Bosselut et al., 2018; Henaff et al., 2017), linking actions (Pareti et al., 2014; Lin et al., 2020; Donatelli et al., 2021), next event prediction (Nguyen et al., 2017; Zellers et al., 2019; Zhang et al., 2020a) and more. OpenPI (Tandon et al., 2020; Zhang et al., 2024a) enables entity tracking in how-to procedures. ProPara (Dalvi et al., 2018) focuses on describing and understanding scientific processes. XPAD (Dalvi et al., 2019) extend ProPara by adding the new task of explaining actions by predicting their dependencies. Zhang et al. (2020b) formalize several multiple-choice tasks related to step- and goal- relations in procedures. Kiddon et al. (2015) explore predicting dependencies in cooking recipes and related tasks. Similar work has been done on identifying dependencies in multimodal instructions with im-

ages and text (Pan et al., 2020; Wu et al., 2024). PizzaCommonsense (Diallo et al., 2024) is a dataset for learning commonsense about intermediate and implicit steps for cooking recipes, and contains explicit input/output pairs of each action with fine-grained annotations. Choice-75 (Hou et al., 2023) aims to study decision branching in plans by generating user scenarios and choices for various steps within the plan. CREPE (Zhang et al., 2023) measures how well LLMs understand the comparative likelihood of two events occurring in a procedure. There are a variety of datasets evaluating different aspects of plans, but there is a lack of one that clearly studies the prediction and explanation of temporal ordering constraints on the steps of an instructional plan.

3 CAT-BENCH

Understanding plans requires reasoning about how different steps in a plan relate to each other. In this work, we focus on the ability to recognize *temporal dependencies* between steps i.e., deciding if a one step *must* happen before another. Typically, step i must happen before a step j if the effects (outcomes) of step i satisfy one or more preconditions necessary for the proper execution of step j, or if the effects of step j aggregate or modify the effects of step i in service of accomplishing a (sub-)goal. For example, in the plan for baking shortcakes shown in Figure 2, step 10 which involves moving the (implicitly mentioned) baked cake to the wire rack for cooling, requires that the cake be baked first, which in turn requires the dough to be placed in the baking tray. Thus, recognizing such dependencies requires the ability to infer many important logical connections such as preconditions, causes, sub-goals, and effects of the steps. This suggests that a simple test of whether a step must happen before another step (or after) can be an effective test of reasoning about the various logical dependencies between the steps in a plan.

We build on this idea to create CAT-BENCH, a new dataset of causal dependency questions defined over cooking recipes. Specifically, we make use of the Recipe Flow Graph Corpus (Yamakata et al., 2020) containing 300 English language cooking recipes annotated with substep procedure dependencies. For each recipe, this dataset provides a directed acyclic graph (DAG), in which the nodes are steps and directed edges indicate the temporal edge between those steps. If the nodes corresponding to two steps are not connected by a directed path, then they can be performed in any order (with respect to themselves) without changing the recipe result. In other words, two steps are temporally dependent if and only if there is a directed path from one to the other, and independent otherwise.

Goal: lemon zested strawberry shortcakes
Steps:
...
6. Divide dough in half.
7. Add sugar; beat until stiff peaks form.
8. Place 5cm apart on an ungreased baking tray.
9. Bake at 200 C / Gas 6 for 8-10 minutes.
10. Remove to a wire rack; cool for 15 minutes.
11. In bowl, combine butter and lemon zest; set aside.
12. In mixing bowl, beat cream until it begins to thicken.
13. Gently pat or roll each half into a 1.75cm thick circle.
14. To assemble, split shortcakes in half.
...

DEP Q: Must Step 8 happen before Step 10?

DEP A: Yes, removing a cake for cooling needs dough to be placed on a baking tray first.

NONDEP Q: Must Step 12 happen after Step 7?

NONDEP A: No, adding sugar is a part of making dough but beating the cream makes the filling.

Figure 2: Examples of different types of questions in a plan from CAT-BENCH. To correctly answer these questions, one must understand preconditions and effects (to answer DEP), some steps need not be performed in any particular order and that plans can contain subplans within them (to answer NONDEP).

For all ordered pairs of steps (i, j) in a plan, we create two binary (yes/no) questions: (i) *Must $step_i$ happen before $step_j$?* (ii) *Must $step_j$ happen after $step_i$?*. These questions primarily test for for precondition relations (e.g. first question in Figure 2), and the ability to understand effects of steps and how they relate to sub-goals or overall goals of the plan (e.g second question in Figure 2). We pool all such questions from dependent pairs of steps (i.e, the steps where there is a directed path from one step's node to the other in the recipe DAG) into DEP, and the rest into NONDEP.[2]

In total, CAT-BENCH contains 2,840 questions about causal dependencies of steps for 57 unique plans. We undersample the non-dependent questions to ensure that NONDEP and DEP are of the same size (i.e., 1,420 questions each). Half of CAT-BENCH tests the "before" temporal relation and the other half tests the "after" relation. It is, thus, balanced in terms of both question types and temporal relation type. We also annotate the questions based

[2]Note that the answers to all the questions in the DEP set are 'yes', and the answers to NONDEP questions are 'no'.

		DEP			NONDEP			Macro Avg		
		P	R	F	P	R	F	P	R	F
gpt-3.5-turbo	(A)	0.62	0.50	0.55	0.58	**0.69**	0.63	0.60	0.60	0.59
	(A+E)	0.56	0.71	0.63	0.61	0.45	0.52	0.58	0.58	0.57
gpt-4-turbo	(A)	0.57	0.81	0.67	0.67	0.39	0.49	0.62	0.60	0.58
	(A+E)	0.66	0.79	0.72	0.74	0.59	0.66	0.70	0.69	0.69
gpt-4o	(A)	0.53	0.92	0.67	0.71	0.19	0.30	0.62	0.55	0.49
	(A+E)	0.66	0.86	0.75	0.80	0.57	0.66	0.73	0.71	0.70
gpt-4o-mini	(A)	0.53	0.88	0.76	0.64	0.22	0.33	0.59	0.55	0.50
	(A+E)	0.62	0.78	0.69	0.70	0.52	0.59	0.66	0.65	0.64
Llama3-8B	(A)	0.52	0.84	0.64	0.59	0.23	0.33	0.56	0.54	0.49
	(A+E)	0.53	0.82	0.64	0.59	0.26	0.36	0.56	0.54	0.50
gemini-1.0-pro	(A)	0.57	0.45	0.50	0.55	0.66	0.60	0.56	0.55	0.55
	(A+E)	0.56	0.65	0.60	0.59	0.50	0.54	0.58	0.57	0.57
gemini-1.5-pro	(A)	0.55	0.77	0.64	0.61	0.37	0.46	0.58	0.57	0.55
	(A+E)	**0.67**	0.93	**0.78**	0.89	0.54	**0.67**	**0.78**	**0.74**	**0.73**
gemini-1.5-flash	(A)	0.55	0.69	0.61	0.58	0.43	0.49	0.56	0.56	0.55
	(A+E)	0.54	0.75	0.63	0.59	0.37	0.46	0.57	0.56	0.54
claude-3.5-sonnet	(A)	0.58	0.76	0.66	0.65	0.46	0.54	0.62	0.61	0.60
	(A+E)	0.63	**0.97**	0.76	**0.93**	0.44	0.60	**0.78**	0.70	0.68

Table 1: Performance of all models on Step Order Prediction when just providing an answer (A) and when also explaining that answer (A+E). We report per-label as well as macro average precision, recall and F1 score.

on the distance between the pairs of steps. Two steps are deemed close if they are within 3 steps of each other, $(j - i) \leq 3$, there are 1,256 questions about *close* steps and 1,584 about *distant* steps.

CAT-BENCH enables two tasks. Step Order Prediction elicits binary judgments about dependencies between pairs of steps in a plan, and performance on this task can be evaluated automatically. Step Order Explanation requires models to provide explanations for their judgments about step dependencies. This involves understanding causal relationships and expressing relevant knowledge about actions in the steps being asked about. Since this is a free-form generation task, these explanations require human evaluation. Note that CAT-BENCH does not contain gold, human-written explanations, and we advocate for reference-free human evaluation since there can be multiple valid explanations.

3.1 Automatic Metrics

We evaluate model performance on Step Order Prediction on standard metrics of precision, recall and F1 score. We measure robustness of these models using two metrics of consistency. Models must provide consistent predictions when asked before/after questions about the same step pair, i.e., if a model judges that $step_i$ happens before $step_j$, it must also judge that $step_j$ happens after $step_i$. We define this metric as Temporal Consistency (TC).

3.2 Human Evaluation

For open-ended text generation tasks such as Step Order Explanation, the absence of an automatic metric that correlates well with human judgments is a major challenge (Chen et al., 2019; Ma et al., 2019; Caglayan et al., 2020; Howcroft et al., 2020). So, we utilize human evaluation with a standardized interface to compare different models. We aim to measure whether a model output is a valid explanation for the given question. We present answers from different models and ask crowd-workers on Amazon Mechanical Turk to assess their correctness. Workers are asked to rate the validity of each answer on a 5-point Likert scale (Likert, 1932) (1 to 5)[3]. For each plan, question, and model answer, we ask 3 distinct annotators to provide judgments. An explanation is considered invalid if it does not give a plausible reason that is also relevant to the question. We provide more details in Appendix C.

[3]Integer scores correspond to the labels: strongly disagree, disagree, neutral, agree, strongly agree.

4 Benchmarking Models on CAT-BENCH

We benchmark the performance of a variety of models on CAT-BENCH.

4.1 Models

We evaluate `gpt-4-turbo`, `gpt-3.5-turbo`, `gpt-4o`, `claude-3.5-sonnet`, `gemini-1.0-pro`, `gemini-1.5-pro`, `gemini-1.5-flash`, `gpt-4o-mini` and `Llama3-8B`. These represent a diverse set from different model families and sizes. We evaluate them primarily in zero-shot prompting modes. We consider two settings: (i) generating only an answer (A), and (ii) generating an explanation along with the answer (A + E). The latter represents answering the question and then generating an explanation for it. We also analyze few-shot results for the answer-only (A) setting, and evaluate generic CoT (E + A) prompting. More details about each model can be found in Appendix A and the prompts used in Appendix D.

4.2 How Good Are Model Predictions?

Table 1 presents the performance of all the models in different settings on Step Order Prediction. We present per-class (DEP and NONDEP) precision, recall and F1 score as well as macro average metrics on the class balanced CAT-BENCH. We make three main observations.

Models struggle at predicting step order. In the zero-shot answer-only setting (A), `claude-3.5-sonnet` records the highest F1 score overall of 0.60. `gpt-3.5-turbo` and `gpt-4-turbo` are close behind with 0.59 and 0.58 respectively. Surprisingly `gpt-4o`, the most recent frontier model, fares significantly worse at 0.49 F1. It's smaller version, `gpt-4o-mini`, also performs similarly. All three Gemini models (`gpt-3.5-turbo`, `gemini-1.0-pro`, and `gemini-1.5-flash`) also only manage an F1 of around 0.55. `Llama3-8B` also fares poorly with an F1 of 0.49. Most models are comparable or barely better than a random baseline F1 of 0.5 on this balanced dataset showing that they are not able to directly answer the dependence question.

Generating explanations improves performance. Results for adding explanations to answers is shown in the (A + E) rows in Table 1. Seven of the nine models, `gpt-3.5-turbo` and `gemini-1.5-flash` being the exceptions, have higher performance when also generating explanations. The biggest improvement in F1 is seen in `gpt-4o` (+0.21). With explanations, the best result is the 0.73 F1 when using `gemini-1.5-pro`. While this is substantially better than a random baseline, there is still significant room for improvement.

Models are biased towards predicting dependence. Most models exhibit a higher recall for the DEP set and significantly lower recall for the NONDEP set. This is particularly true for the answer only setting ((A) rows), the exceptions being `gpt-3.5-turbo` and `gemini-1.0-pro`. Coupled with the substantially lower precision values on the DEP set, this suggests that most models exhibit a bias towards predicting dependence between any given pair of steps. We hypothesize that they use temporal order of steps as a heuristic i.e, if a step appears before another step it is more likely to be dependent than not, and thus becoming biased towards predicting dependencies.

As noted earlier, using explanations improves the overall performance, translating to more balanced precision/recall values on DEP than when predicting answers alone. Since the bias towards DEP necessarily means bias against NONDEP, reduction in bias towards DEP also translates to a more balanced performance on both DEP and NONDEP sets. However, even with explanations, the bias towards predicting dependence still remains to some extent for all models. Explanations improve `gpt-4o` performance the most (+0.36) on NONDEP questions. They do not help smaller models (`Llama3-8B`) identify dependencies better.

4.3 How Good Are Model Explanations?

On a random subset of 480 questions (240 DEP and 240 NONDEP), we conduct a crowdsourced human evaluation of the explanations generated by `gpt-4o`, `gpt-4-turbo`, `gemini-1.5-pro` and `Llama3-8B`, the three best LLMs for Step Order Prediction and an open-source model. Annotators rate how much they agree (1 to 5) with the fact that the answer contains all the relevant details to address what the question requires.

For each explanation, we compute the mean Likert rating from three distinct annotators. First, we report AVG, the overall average of these mean ratings across all 480 instances. To account for cases where the answer is incorrect, we also devise a new metric MODAVG that accounts for cases where the step order prediction is incorrect. To calculate

MODAVG, we modify AVG by zeroing out human judgments for explanations where the corresponding prediction is incorrect.

We use weighted Fleiss Kappa to calculate inter-annotator agreement. The weighted agreement score on a 5 point scale was 0.76, indicating high agreement between annotators. Details about the calculation can be found in Appendix C.3.

	AVG	MODAVG
`Llama3-8B`	3.26	1.87
`gpt-4-turbo`	3.85	2.90
`gpt-4o`	3.84	2.93
`gemini-1.5-pro`	3.83	2.69

Table 2: Human evaluation metrics for explanations generated by various models in the (A+E) setting.

Table 2 presents the quality of model generated explanations as judged by human annotators. As expected, larger models are clearly better than the much smaller `Llama3-8B` on all metrics. There is very little difference between the frontier models, `gpt-4o`, `gpt-4-turbo` and `gemini-1.5-pro`. AVG performance indicates that there is significant room for improvement in the quality of model explanations. On MODAVG, we see that even the best model performance is below 3 ('neither agree nor disagree' with a model's explanation). By this metric, `gemini-1.5-pro` explanations are worse than GPT-4 even though it generates more correct answers. The difference between AVG and MODAVG indicates models are capable of generating convincing explanations for their wrong answers. They produce explanations which justify the opposite of their answer a significant number of times. In fact, `Llama3-8B` does so almost half the time. These results show that models have a lot of room for improvement in their ability and reliability to reason about step dependencies in plans.

5 Analysis

To better understand the strengths and weaknesses of these models, we analyze their performance on CAT-BENCH organized by different characteristics of the questions and model prompts.

5.1 Robustness of Models

Table 3 presents two measures of consistency to quantify the robustness of the models, similar to (Verma et al., 2023; Elazar et al., 2021).

Goal: lemon zested strawberry shortcakes

Steps:
...
6. Divide dough in half.
7. Add sugar; beat until stiff peaks form.
8. Place 5cm apart on an ungreased baking tray.
...
12. In mixing bowl, beat cream until it begins to thicken.
13. Gently pat or roll each half into a 1.75cm thick circle.
...

NONDEP Q: Must Step 12 happen after Step 7?

NONDEP A: No, adding sugar is a part of making dough but beating the cream makes the filling.

Steps:
...
6. Divide dough in half.
7. In mixing bowl, beat cream until it begins to thicken.
8. Place 5cm apart on an ungreased baking tray.
...
12. Add sugar; beat until stiff peaks form.
13. Gently pat or roll each half into a 1.75cm thick circle.
...

NONDEP-S Q: Must Step 12 happen after Step 7?

NONDEP-S A: No, making the filling with cream can be done in parallel to adding sugar in the dough.

Figure 3: Since two steps that are not dependent on each other can be performed in any order, we swap their order in the plan and ask binary questions about them similar to NONDEP. Note that, while the plan itself is altered, the question remains the same.

Temporal Consistency For a pair of steps $(step_i, step_j)$, the answer to must $step_i$ happen before $step_j$ should be the same as the answer to must $step_j$ happen after $step_i$ regardless of question type. As described in subsection 3.1, we measure this notion of consistency through TC. We make two main observations: (i) Even the most consistent models `gpt-4o`, `gemini-1.5-pro` and `claude-3.5-sonnet` change their answers to the before and after versions of questions in 20+% of the cases. The rest are far more inconsistent with `gemini-1.5-flash` changing its answers for more than 55% of the questions; (ii) Surprisingly, adding explanations reduces answer consistency for most models, with `gemini-1.5-pro` (+24%), `gpt-4-turbo` (+14%) and `claude-3.5-sonnet` (+31%) being the only exceptions showing improved consistency upon generating explanations.

Order Contrastive Consistency Since step pairs without dependencies can be performed in any order, we introduce a twist on Step Order Prediction in which the step pairs in NONDEP are switched in the plan itself. For each modified plan, we create similar binary questions to NONDEP and refer

	TC	OCC
gpt-3.5-turbo (A)	52.39%	70.42%
gpt-3.5-turbo (A+E)	49.23%	73.31%
gpt-4-turbo (A)	48.87%	70.28%
gpt-4-turbo (A+E)	55.00%	66.97%
gpt-4o (A)	**79.86%**	47.96%
gpt-4o (A+E)	67.46%	58.17%
gpt-4o-mini (A)	70.56%	54.79%
gpt-4o-mini (A+E)	57.54%	56.97%
Llama3-8B (A)	60.42%	**83.87%**
Llama3-8B (A+E)	55.77%	83.38%
gemini-1.0-pro (A)	53.38%	73.80%
gemini-1.0-pro (A+E)	49.79%	66.90%
gemini-1.5-pro (A)	55.14%	58.24%
gemini-1.5-pro (A+E)	79.65%	60.21%
gemini-1.5-flash (A)	45.92%	79.44%
gemini-1.5-flash (A+E)	43.10%	76.62%
claude-3.5-sonnet (A)	45.14%	50.21%
claude-3.5-sonnet (A+E)	76.83%	48.10%

Table 3: Robustness of different models on two consistency metrics, TC and OCC.

to them as NONDEP-S. This helps test whether a model uses the step order as a heuristic to answer the question. We show an example in Figure 3.

The answer to dependency questions about an independent pair of steps should stay the same regardless of the order in which the steps are presented in the plan. Order Contrastive Consistency (OCC) measures the fraction of times models provide consistent answers to the same question across NONDEP and NONDEP-S. We observe a similar overall inconsistency on OCC as with TC, even from the best models. For most models, generating explanations hurt consistency. Surprisingly, Llama3-8B is the most robust according to OCC even though its task performance is lowest. In contrast, gemini-1.5-pro, which has the highest task performance, is the least robust as per this metric.

5.2 Chain-of-Thought Struggles

In the experiments thus far, we have asked models to generate explanations for their answers. This can be seen as a answer-then-explain (A + E) approach. In contrast, the standard Chain-of-Thought (CoT) prompting strategy (Wei et al., 2022b) can be seen as an explain-then-answer approach (E + A), where we ask the model to generate reasoning or explanation that leads to its answer. In practice, this step-by-step reasoning can be seen as allowing the use of intermediate decoding tokens (like a scratchpad) in service to coming up with a possibly more accurate final answer for many tasks. Table 4 compares performance of CoT prompting[4] (E + A) to first predicting the answer and then explaining it (A + E) and simply providing an answer (A), all in the zero-shot setting when using gpt-4o.[5]

	P	R	F1	TC
(A)	0.62	0.55	0.49	79.86%
(E+A)	0.77	0.65	0.6	83.66%
(A+E)	0.73	0.71	0.7	67.46%

Table 4: Performance of gpt-4o on the Step Order Prediction task when just predicting the dependency (A) vs predicting and explaining the judgment (A+E) vs using chain-of-thought prompting (E+A).

While chain-of-thought (E + A) results in an improvement over the answer-only setting (A), its performance is far below its counterpart (A + E). This contradicts the expectation that it is better to use CoT for intermediate reasoning rather than answering and then generating explanations. However, we note that (E + A) does lead to the highest temporal robustness amongst all approaches.

Looking closer, Table 5 shows the performance of these methods on DEP questions. CoT (E + A) has a higher bias towards predicting dependencies compared to both (A) and (A + E).

	P	R	F1
(A)	0.53	0.92	0.67
(E+A)	0.59	0.98	0.74
(A+E)	0.66	0.86	0.75

Table 5: Performance of gpt-4o on the Step Order Prediction task when just predicting the dependency (A) vs predicting and explaining the judgment (A+E) vs using chain-of-thought prompting (E+A) on DEP.

Further manual analysis of the explanations shows that (E + A) generates more vacuous explanations (e.g. just stating that a pair of steps are independent without digging deeper into what makes them independent). Additionally, (E + A) makes errors by misunderstanding entity states for a step. Unsurprisingly, both settings suffer from some hallucinations, sometimes missing the content of the steps being asked about. We present an example of

[4]We tried multiple CoT prompts, all with $temp = 0$, but it had little effect on performance.

[5]We use gpt-4o instead of gemini-1.5-pro due to rate limits on the latter.

Goal: lightly spiced tomato and red lentil soup
Steps:
1. Heat the oil in a saucepan.
2. Add onion and cook over low heat for 7-8 mins...
3. Stir in celery and carrot and cook for 3 mins...
...
9. Blend the soup using hand-held blender.
10. Check the consistency.
11. Season to taste.
...
Question: Explain why or why not Step 10 must happen after Step 3.
Model Answer: Step 10 does not need to happen after Step 3 because the eggplant needs to be cooked before blending.
Human Score: 1.0

Figure 4: Example of hallucinations produced by GPT-4 in the (E + A) setting.

such hallucinations in Figure 4. These results are further indicators of brittleness and inconsistencies in models' reasoning about step dependencies.

We also include zero-shot results with `o1-preview`, which was released just before the time of publication. `o1-preview` uses search over chain of thought explanations as part of its inference process.[6] On CAT-BENCH, this model achieves state-of-the-art performance scoring 0.80 F1. This is better than both zero- and few-shot performance of any other model, and even achieves the best TC at 85%. However, even this powerful model shows a bias towards predicting dependence (F1 of 0.83) between steps more than their non-dependence (F1 of 0.76). Due to rate limits and prohibitive costs ($32 for each Step Order Prediction experiment), we were unable to investigate `o1-preview` further.

5.3 Effect of Improved Prompting Techniques

We also experiment with self-consistency (Mitchell et al., 2022) and few-shot prompting (or in-context learning) (Brown et al., 2020; Wei et al., 2022a) on `gpt-4o` (A)[7] for the Step Order Prediction.

For self-consistency, we use k=$\{3, 5\}$ and temperature=$\{0.6, 0.8\}$ to sample binary predictions and take the majority of the predicted labels as the model's final answer. Table 6 shows the results for one setting. Contrary to previous findings (Mitchell et al., 2022), self-consistency does not provide any improvement over the vanilla zero-shot model performance. We report performance with other parameters in Table 7.

	P	R	F
Zero-shot	0.62	0.55	0.49
Self-Consistency	0.62	0.55	0.49
Few-shot	0.79	0.70	0.68

Table 6: Performance of different prompting techniques with `gpt-4o` on Step Order Prediction. For self-consistency, we report $k = 3$ and $temp = 0.6$ here, and use 5 exemplars for few-shot experiments.

We use in-context learning (Wei et al., 2022a) with examples selected from the balanced training set using the BM25[8] (Robertson and Zaragoza, 2009) algorithm. We use k=5 exemplars and dynamically retrieve exemplars from a held-out set that are closest to the test instance. As expected, few-shot prompting improves binary prediction performance a lot (+0.19). In fact, few-shot performance is almost as good as predicting then explaining with `gpt-4o`.

5.4 Error Analysis

To better understand model failures, we sampled and analyzed 50 explanations generated by `gpt-4o` (A + E) where it produces an incorrect answer. We identify 4 major types of errors:

- Multi-hop dependency (40%): Failure to understand that two steps might be related through an intermediate step. For instance, to make shortcakes in Figure 5, removing the cake from the oven (Step 10) is dependent on baking the cake (Step 9) which, consequently, is dependent on combining the ingredients (Step 2). The model does not seem to understand the transitive nature of such dependencies.
- Effects (20%): Failure to understand that an effect of the preceding step leads to the succeeding step, e.g., serving a cake in Figure 6 must happen after mixing ingredients and consequently baking. LLMs fail to identify additive effects of steps in a plan which enable a later step, leading to goal completion.
- Preconditions (18%): Failure to understand a condition that needs to be satisfied for a step to happen. For instance in Figure 6, to add sauce in Step 20, meatballs need to cooked in Step 15 so they can be added to the sauce in Step 17.

[6] https://openai.com/index/introducing-openai-o1-preview/

[7] Due to the lack of gold explanations, we are unable to run these variations for (A + E) or (E + A).

[8] https://github.com/dorianbrown/rank_bm25

Goal: vanilla rye cheesecake
Steps:
1. Preheat the oven to 150 degrees C
2. Place the biscuits into a plastic bag and crush.
...
10. Place baking dish onto bottom rack of preheated oven and fill halfway with boiling water.
...
Question: Explain why or why not Step 10 must happen after Step 2.

Model Answer: Step 10 must happen after Step 2 because the baking dish with boiling water needs to be placed in the oven just before baking the cheesecake to create a moist environment, which is necessary after the biscuit base has been prepared and the filling has been added.

Human Score: 5.0

Goal: lemon zested strawberry shortcakes
Steps:
1. In bowl, combine strawberries and sugar...
2. In bowl, combine flour, sugar, baking powder and salt.
3. Rub butter into flour until forming coarse crumbs.
...
9. Bake at 200 C / Gas 6 for 8-10 minutes.
10. Remove to a wire rack; cool for 15 minutes.
11. In bowl, combine butter and lemon zest; set aside.
...
Question: Explain why or why not Step 10 must happen after Step 2.

Model Answer: Step 10 involves cooling the baked shortcakes, which are prepared and baked in steps 5 through 9, and does not depend on the flour mixture preparation in step 2.

Human Score: 1.0

Figure 5: Examples of cases where GPT-4 comes up with good (upper box) and bad (lower box) answers. This error is of the multi-hop dependency type. To make shortcakes, removing the cake from the oven (Step 10) is dependent on baking the cake (step 9) which is later dependent on combining the ingredients (Step 2). Examples of other error types can be found in Figure 6.

- Irrelevant Answers - Model produces answers that are unrelated to the step being asked about, e.g., in Figure 6 to make chocolate cake, the model's answer does not address a relevant step (Step 7) at all. It is surprising to see that LLMs mistakenly produce an answer about an unrelated step, particularly given that the input context is short (well below maximum context length) and can be easily used for grounding.

6 Conclusion

Understanding plans requires reasoning about its different aspects such as preconditions and effects. This paper introduces CAT-BENCH, a new benchmark to evaluate the causal and temporal reasoning abilities about plans. Despite the remarkable strength of current SOTA LLMs, we find that none of them are very good at understanding whether one step in a plan must precede (or succeed) another. Particularly, they are much worse at knowing when there is *not* a dependency between steps. We also find that LLM predictions are not robust as measured by two metrics of consistency. Prompting LLMs to provide an answer and then to explain it improves performance significantly, and is even better than chain-of-thought (reasoning followed by answering). Human evaluation of these explanations shows that models have a long way to go at understanding dependencies.

Limitations

While our work only considers cooking recipes as procedural texts, our methods can in principle be applied to many other domains. Medical practice guidelines, repair manuals, and software tutorials among others are domains worth investigating. Our work only investigates English-language documents and this limits the generalizability of our findings to other languages.

We benchmark a reasonably diverse set of LLMs. Currently, we cover 3 model families and models of varying sizes. Due to the current fast-paced landscape of LLM development, we will continue to evaluate more LLMs on CAT-BENCH.

It is difficult for any one person to adequately evaluate the various aspects of plans, particularly recipes. To alleviate this problem, we use 3 crowdsourced annotators to judge model explanations and consider their average judgment (Lal et al., 2022), but recognize the limitations of this solution. We do show high inter-annotator agreement (Lal et al., 2021) using Weighted Fleiss Kappa (Marasini et al., 2016), demonstrating the reliability of our results. While human evaluation is expensive and time-consuming and the number of experiments per model balloons costs exponentially, it is critical for open-ended generation tasks. We evaluate enough explanations to obtain statistically significant results.

We use BM25 as a reasonable choice to find similar exemplars. We acknowledge that there are more modern techniques for selecting in-context examples, but this step is not the focus of our current work. We leave further exploration of exemplar selection methods to future work. For a domain like recipe text where texts are long and less amenable to a single embedding vector approach, keyword-based retrieval such as BM25 is very effective.

Since there are no gold explanations, we cannot combine few-shot prompting and chain-of-thought (or answer then explain) settings for `gpt-4o`. Note that we also do not advocate for using gold explanations along with automatic metrics to judge model explanations due to established inadequacies in using automatic metrics for free-form generations.

Due to very strict rate limits on the recently released Gemini models, we are unable to analyze `gemini-1.5-pro` through chain-of-thought and other prompting techniques. For consistency, we analyze `gpt-4o` since it has similar performance (A + E) on CAT-BENCH.

Ethical Considerations

Prior work has shown that LLMs exhibit various types of bias. While they do not generate free-form language for our binary prediction task, it is possible, though highly unlikely, that biases explicitly come up in the explanations. Deploying such unreliable models into critical infrastructure and relying on them for decisions can cause harm to users.

Acknowledgements

This material is based on research that is supported in part by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003 and in part by the National Science Foundation under the award IIS #2007290. This material is also based upon work supported by the DARPA's Perceptually-enabled Task Guidance (PTG) program under Contract No. HR001122C007.

References

Farida Aouladomar and Patrick Saint-Dizier. 2005. Towards generating procedural texts: An exploration of their rhetorical and argumentative structure. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*, Aberdeen, Scotland. Association for Computational Linguistics.

Mohaddeseh Bastan, Mahnaz Koupaee, Youngseo Son, Richard Sicoli, and Niranjan Balasubramanian. 2020. Author's sentiment prediction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 604–615, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. *ICLR*.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D. Hwang, Xiang Lorraine Li, Hirona J. Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. 2023. Plasma: Making small language models better procedural knowledge models for (counterfactual) planning. *Preprint*, arXiv:2305.19472.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ozan Caglayan, Pranava Madhyastha, and Lucia Specia. 2020. Curious case of language generation evaluation metrics: A cautionary tale. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2322–2328, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

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.

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

Aissatou Diallo, Antonis Bikakis, Luke Dickens, Anthony Hunter, and Rob Miller. 2024. Pizzacommonsense: Learning to model commonsense reasoning about intermediate steps in cooking recipes. *Preprint*, arXiv:2401.06930.

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. 2021. 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. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. *ICLR*.

Zhaoyi Joey Hou, Li Zhang, and Chris Callison-Burch. 2023. Choice-75: A dataset on decision branching in script learning. *arXiv preprint arXiv:2309.11737*.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Yuqian Jiang, Shiqi Zhang, Piyush Khandelwal, and Peter Stone. 2019. Task planning in robotics: an empirical comparison of pddl-based and asp-based systems. *Preprint*, arXiv:1804.08229.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. 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. Association for Computational Linguistics.

Sawan Kumar and Partha Talukdar. 2020. NILE : Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, Online. Association for Computational Linguistics.

Yash Kumar Lal, Nathanael Chambers, Raymond Mooney, and Niranjan Balasubramanian. 2021. TellMeWhy: A dataset for answering why-questions in narratives. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 596–610, Online. Association for Computational Linguistics.

Yash Kumar Lal, Niket Tandon, Tanvi Aggarwal, Horace Liu, Nathanael Chambers, Raymond Mooney, and Niranjan Balasubramanian. 2022. Using commonsense knowledge to answer why-questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1219, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yash Kumar Lal, Li Zhang, Faeze Brahman, Bodhisattwa Prasad Majumder, Peter Clark, and Niket Tandon. 2024. Tailoring with targeted precision: Edit-based agents for open-domain procedure customization. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15597–15611, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Steven M. LaValle. 2006. *Planning Algorithms*. Cambridge University Press, USA.

Duong Minh Le, Ruohao Guo, Wei Xu, and Alan Ritter. 2023. Improved instruction ordering in recipe-grounded conversation. *Preprint*, arXiv:2305.17280.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of psychology*.

Angela Lin, Sudha Rao, Asli Celikyilmaz, Elnaz Nouri, Chris Brockett, Debadeepta Dey, and Bill Dolan. 2020. 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. Association for Computational Linguistics.

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

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 62–90, Florence, Italy. Association for Computational Linguistics.

D. Marasini, P. Quatto, and E. Ripamonti. 2016. Assessing the inter-rater agreement for ordinal data through weighted indexes. *Statistical Methods in Medical Research*, 25:2611 – 2633.

Eric Mitchell, Joseph Noh, Siyan Li, Will Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher Manning. 2022. Enhancing self-consistency and performance of pre-trained language models through natural language inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1768, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dai Quoc Nguyen, Dat Quoc Nguyen, Cuong Xuan Chu, Stefan Thater, and Manfred Pinkal. 2017. 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. Asian Federation of Natural Language Processing.

Liang-Ming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yugang Jiang, and Tat-Seng Chua. 2020. Multi-modal cooking workflow construction for food recipes. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 1132–1141, New York, NY, USA. Association for Computing Machinery.

Paolo Pareti, Benoit Testu, Ryutaro Ichise, Ewan Klein, and Adam Barker. 2014. Integrating know-how into the linked data cloud. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 385–396. Springer.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

R.C. Schank and R.P. Abelson. 1977. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

John Slaney and Sylvie Thiébaux. 2001. Blocks world revisited. *Artificial Intelligence*, 125(1):119–153.

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.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Preprint*, arXiv:2206.10498.

Dhruv Verma, Yash Kumar Lal, Shreyashee Sinha, Benjamin Van Durme, and Adam Poliak. 2023. Evaluating paraphrastic robustness in textual entailment models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 880–892, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

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

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. 2024. The generative AI paradox: "what it can create, it may not understand". In *The Twelfth International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2024. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. *Preprint*, arXiv:2110.08486.

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English recipe flow graph corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. 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. Association for Computational Linguistics.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. 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. Association for Computational Linguistics.

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

Li Zhang, Hainiu Xu, Abhinav Kommula, Chris Callison-Burch, and Niket Tandon. 2024a. 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 (Volume 1: Long Papers)*, pages 166–178, St. Julian's, Malta. Association for Computational Linguistics.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023. Causal reasoning of entities and events in procedural texts. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 415–431, Dubrovnik, Croatia. Association for Computational Linguistics.

Tianyi Zhang, Li Zhang, Zhaoyi Hou, Ziyu Wang, Yuling Gu, Peter Clark, Chris Callison-Burch, and Niket Tandon. 2024b. Proc2pddl: Open-domain planning representations from texts. *Preprint*, arXiv:2403.00092.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2024. Natural plan: Benchmarking llms on natural language planning. *Preprint*, arXiv:2406.04520.

A Benchmark Models

We provide details of each model we evaluate on CAT-BENCH. For in-context example selection for the LLMs we utilize the rank_BM25 library[9] available under the Apache 2.0 license.

gpt-4o-2024-05-13 accepts as input any combination of text, audio, image, and video and generates any combination of text, audio, and image outputs. It is especially better at vision and audio understanding compared to existing models.

gpt-3.5-turbo is an instruction-tuned pretrained language model that powered the original ChatGPT application.

gpt-4-turbo-2024-04-09 is the first turbo model in the GPT-4 series. It is more capable and cheaper than GPT-3.5, and supports a 128K context window. It is possibly a 8x220B Mixture-of-Experts model.

gemini-1.5-flash-latest is a version of `gemini-1.5-pro` optimized for low latency and inference cost.

gemini-1.5-pro-latest is built on a Mixture-of-Experts (MoE) architecture. `gemini-1.5-pro` a mid-size multimodal model, optimized for scaling across a wide-range of tasks, and also introduces a breakthrough experimental feature in long-context understanding. It is difficult to perform a wide range of experiments with this model due to the imposed rate limits.

gpt-4o-mini-2024-07-18 has a context window of 128K tokens, supports up to 16K output tokens per request. It surpasses `gpt-4-turbo` and other small models on academic benchmarks across both textual intelligence and multimodal reasoning, and supports the same range of languages as `gpt-4o`.

gemini-1.0-pro-latest is built on top of Transformer decoders that are enhanced with improvements in architecture and model optimization to enable stable training at scale and optimized inference. They are trained to support 32k context length, employing efficient attention mechanisms (for e.g. multi-query attention). Gemini models are trained to accommodate textual input interleaved with a wide variety of audio and visual inputs, and `gemini-1.0-pro` is the mid-sized model in the series.

claude-3-5-sonnet-20240620 sets new industry benchmarks for graduate-level reasoning (GPQA), undergraduate-level knowledge (MMLU), and coding proficiency (HumanEval). It shows marked improvement in grasping nuance, humor, and complex instructions, and is exceptional at writing high-quality content with a natural, relatable tone.

Meta-Llama3-8B-Instruct is a standard decoder-only transformer architecture similar to its predecessor Llama2. Compared to Llama2, `Llama3-8B` uses a tokenizer with a vocabulary of 128K tokens that encodes language much more efficiently, which leads to substantially improved model performance. It also uses grouped query attention (GQA) and was trained on sequences of 8,192 tokens, using a mask to ensure self-attention does not cross document boundaries. `Llama3-8B` is pretrained on over 15T tokens that were all collected from publicly available sources.

B Additional Analysis Results

We use scikit-learn classification report to calculate precision, recall and F1 score per-class and as macro average.

B.1 Understanding Directional Dependencies

Next, we study how models handle questions about different aspects of the same pair of steps. Typically, questions about why a step must happen *before* another require reasoning about preconditions and causes, while answering why a step must happen *after* another requires understanding the effects of any performed actions. Figure 7 shows the difference in F1 score between answering and providing an explanation (A + E) and the answer-only setting (A) with different models for these questions. Adding explanations helps all models understand effects better (*after*). We hypothesize that this is because effects in recipes can be more immediate and hence, would be easier to understand. The biggest gain is seen for `gemini-1.5-pro` and `gpt-4o`, while `Llama3-8B` and `gemini-1.0-pro` do not improve a lot. We note that explanations significantly hurt `gpt-3.5-turbo` and `gemini-1.5-flash` in understanding preconditions, which is unusual. Similar to *after* questions, `Llama3-8B` and `gemini-1.0-pro` do not improve a lot with explanations on *before* questions.

[9] https://github.com/dorianbrown/rank_bm25

Goal: fusilli and meatballs
Steps: ...
15. Cook the meatballs until browned on all sides.
16. Drain on kitchen paper.
17. Add the meatballs to the tomato sauce.
...
20. Toss with 3 ladlefuls of sauce in serving bowl..
...
Question: Explain why or why not Step 20 must happen after Step 15.

Model Answer: Step 20 can happen after Step 19 as it involves tossing the cooked pasta with the sauce, which is independent of browning the meatballs in Step 15.

Goal: chocolate cake
Steps:
1. Preheat oven to 180C / Gas Mark 4.
2. Put chocolate chips, coffee, water in bowl.
3. Microwave until melted.
...
7. Gradually beat in sugar till fluffy.
8. Gradually add and beat in egg yolks.
...

Question: Explain why or why not Step 3 must happen before Step 7.

Model Answer: The chocolate mixture needs to be cooled before it is added to the batter in Step 10.

Goal: almond and apple cake
Steps:...
7. Add half the flour and half the milk.
8. Use a wooden spoon to stir until well combined.
...
13. Set aside to cool slightly before plating.
14. Serve warm.
...

Question: Explain why or why not Step 14 must happen after Step 8.

Model Answer: Step 14 is the final step and cannot happen after mixing the ingredients.

Figure 6: Examples of types of GPT-4 errors. The top box contains an error related to misunderstanding preconditions, the second one about producing irrelevant answers and the last one about misunderstanding effects.

B.2 Reasoning as a function of Step Distance

Next, we study how the distance between the steps in question impacts model performance. A question is said to be about *close* steps $(step_i, step_j)$ if $(j - i) < 3$, and *distant* otherwise. Figure 8 shows the difference in F1 score between answering then explaining (A + E) and just answering (A) with different models as a function of step distance. Generating explanations helps models reason about distant steps, with `gpt-4o` and `gemini-1.5-pro` receiving the greatest benefit. However, they don't help understand dependencies between close steps (which are easier to reason about). In fact, producing explanations even hurts some models, particularly `gpt-3.5-turbo`. We hypothesize that models are likely to predict a dependencies between steps that are distant from each other, since it is likely that steps towards the end of a plan depend on ones near the start. We find that this is indeed true, the recall for the nondependent is very low (usually, $\sim$20%) and they are biased towards predicting a dependency between distant steps.

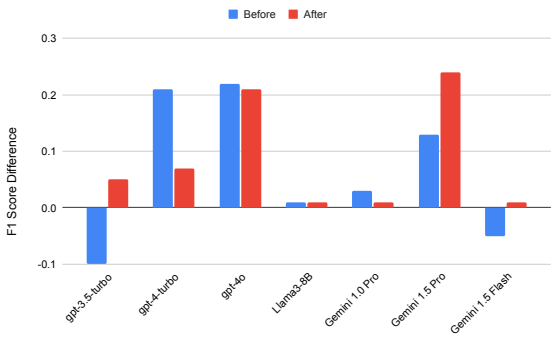


Figure 7: Difference in model performance between (A+E) and (A) settings split by temporal relation type (*before* and *after*) asked about in the question. We subtract F1 score in the (A) from the (A+E) setting.

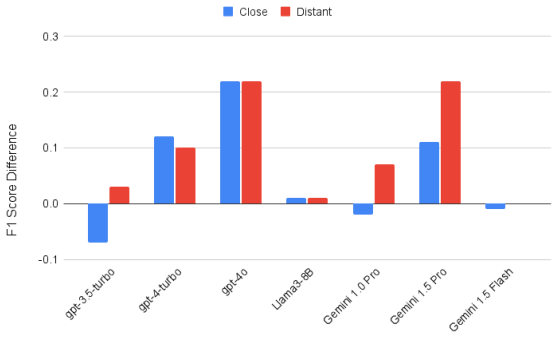


Figure 8: Difference in performance of models between (A+E) and (A) settings split by the distance between the steps being asked about in the question.

B.3 Prompting Variations

Table 7 presents more results for prompting techniques. In particular, we show more variations of self-consistency. We find that varying the temperature and number of samples does not make a significant difference.

B.4 Error Examples

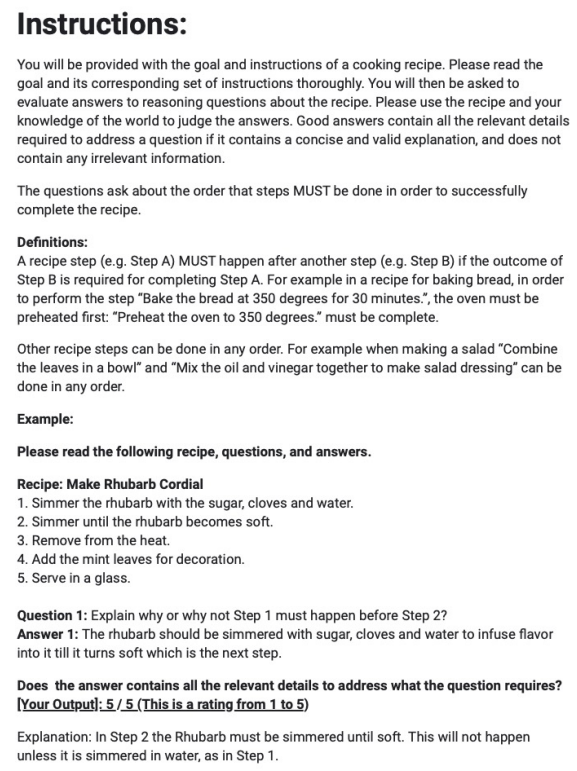
We present examples of different types of errors in Figure 5 and Figure 6.

	P	R	F	TC	OCC
Zero-shot	0.62	0.55	0.49	79.86%	47.96%
Self-Consistency [3, 0.6]	0.62	0.55	0.49	78.66%	48.17%
Self-Consistency [3, 0.8]	0.61	0.55	0.48	79.58%	47.25%
Self-Consistency [5, 0.6]	0.61	0.55	0.48	79.72%	46.76%
Self-Consistency [5, 0.8]	0.61	0.55	0.48	79.86%	45.99%
Few-shot	0.79	0.70	0.68	81.48%	45.56%

Table 7: Accuracy and consistency of different prompting techniques with `gpt-4o` on the Step Order Prediction Task. The first number within the square bracket in self-consistency experiments represents the number of samples and the second represents the temperature at which predictions were sampled. For few-shot experiments, we use k=5 exemplars and use BM25 (Robertson and Zaragoza, 2009) to dynamically retrieve exemplars from a held-out set that are closest to the test instance.

C Human Evaluation Details

C.1 Task Details



Instructions:

You will be provided with the goal and instructions of a cooking recipe. Please read the goal and its corresponding set of instructions thoroughly. You will then be asked to evaluate answers to reasoning questions about the recipe. Please use the recipe and your knowledge of the world to judge the answers. Good answers contain all the relevant details required to address a question if it contains a concise and valid explanation, and does not contain any irrelevant information.

The questions ask about the order that steps MUST be done in order to successfully complete the recipe.

Definitions:
A recipe step (e.g. Step A) MUST happen after another step (e.g. Step B) if the outcome of Step B is required for completing Step A. For example in a recipe for baking bread, in order to perform the step "Bake the bread at 350 degrees for 30 minutes.", the oven must be preheated first: "Preheat the oven to 350 degrees." must be complete.

Other recipe steps can be done in any order. For example when making a salad "Combine the leaves in a bowl" and "Mix the oil and vinegar together to make salad dressing" can be done in any order.

Example:

Please read the following recipe, questions, and answers.

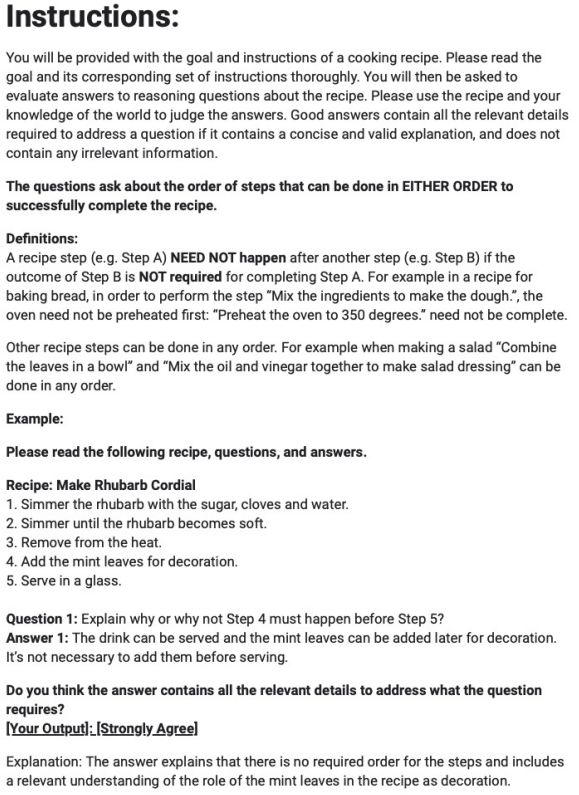
Recipe: Make Rhubarb Cordial
1. Simmer the rhubarb with the sugar, cloves and water.
2. Simmer until the rhubarb becomes soft.
3. Remove from the heat.
4. Add the mint leaves for decoration.
5. Serve in a glass.

Question 1: Explain why or why not Step 1 must happen before Step 2?
Answer 1: The rhubarb should be simmered with sugar, cloves and water to infuse flavor into it till it turns soft which is the next step.

Does the answer contains all the relevant details to address what the question requires?
[Your Output]: 5 / 5 (This is a rating from 1 to 5)

Explanation: In Step 2 the Rhubarb must be simmered until soft. This will not happen unless it is simmered in water, as in Step 1.

Figure 9: Instructions provided to annotators when making judgments about explanations for DEP questions.



Instructions:

You will be provided with the goal and instructions of a cooking recipe. Please read the goal and its corresponding set of instructions thoroughly. You will then be asked to evaluate answers to reasoning questions about the recipe. Please use the recipe and your knowledge of the world to judge the answers. Good answers contain all the relevant details required to address a question if it contains a concise and valid explanation, and does not contain any irrelevant information.

The questions ask about the order of steps that can be done in EITHER ORDER to successfully complete the recipe.

Definitions:
A recipe step (e.g. Step A) **NEED NOT happen** after another step (e.g. Step B) if the outcome of Step B is **NOT required** for completing Step A. For example in a recipe for baking bread, in order to perform the step "Mix the ingredients to make the dough.", the oven need not be preheated first: "Preheat the oven to 350 degrees." need not be complete.

Other recipe steps can be done in any order. For example when making a salad "Combine the leaves in a bowl" and "Mix the oil and vinegar together to make salad dressing" can be done in any order.

Example:

Please read the following recipe, questions, and answers.

Recipe: Make Rhubarb Cordial
1. Simmer the rhubarb with the sugar, cloves and water.
2. Simmer until the rhubarb becomes soft.
3. Remove from the heat.
4. Add the mint leaves for decoration.
5. Serve in a glass.

Question 1: Explain why or why not Step 4 must happen before Step 5?
Answer 1: The drink can be served and the mint leaves can be added later for decoration. It's not necessary to add them before serving.

Do you think the answer contains all the relevant details to address what the question requires?
[Your Output]: [Strongly Agree]

Explanation: The answer explains that there is no required order for the steps and includes a relevant understanding of the role of the mint leaves in the recipe as decoration.

Figure 10: Instructions provided to annotators when making judgments about explanations for NONDEP questions.

Figure 9 and Figure 10 show the instructions as well as one of the examples presented to annotators when eliciting judgments for model explanations for DEP and NONDEP questions. For each HIT, workers are asked to read the goal of the plan and its steps and then evaluate 6 randomized questions and corresponding answers from models, providing judgments on a Likert scale of 1 to 5. We only select US-based master turkers who have a minimum lifetime approval rating of 95%. On average, workers took 3 minutes and 51 seconds to judge 6 answers to questions about a plan. We pay them $1.5 per HIT which translates to $23.35 per hour, significantly higher than federal and local minimum wage.

C.2 Additional Human Evaluation Results

We also used two additional metrics to interpret human judgments of model answers. For AVGBIN, we transform each score into a binary value (1 if >3 and 0 otherwise), calculate the mean of these

values for an answer and average them over all the data points. We calculate the majority binary class of judgments for each explanation as MAJVOTE. We report all the metrics for DEP and NONDEP in Table 8 and Table 9 respectively. Looking at AVG-BIN, we note that there is room for improvement on the quality of model explanations. MAJVOTE indicates that model explanations are convincing even when they are wrong. Note that these metrics do not account for corresponding answer correctness for a model's explanation.

Figure 11 presents the distribution of human judgment scores for explanations generated by various models (A + E). We note that models frequently produce high quality answers (5); however, they make too many errors (<3 out of 5) to be consistently reliable .

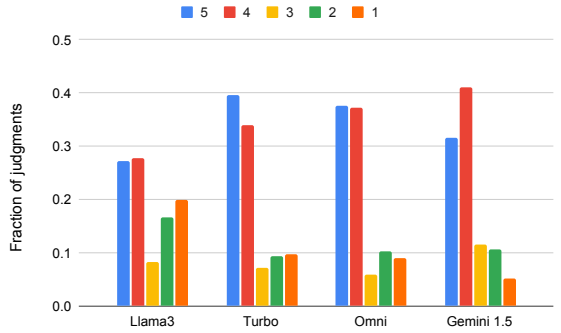


Figure 11: Distribution of human judgment scores for explanations generated by various models (A+E).

C.3 Inter-annotator Agreement

We measured the inter-rater reliability of annotators' judgments using weighted Fleiss's Kappa (Marasini et al., 2016), following the weighting scheme used by Bastan et al. (2020). This measure has a penalty for each dissimilar rating based on the distance between the two ratings. For instance, if two annotators classify a document as a positive, the agreement weight is 1, but if one classifies as a positive, and the other classifies as slightly positive the agreement weight is less. The weights between different classes are shown in Table 10 where negative, slightly negative, neutral, slightly positive, and positive classes are shown with -2, -1, 0, 1, and 2. We follow the setup used in Bastan et al. (2020) for a similar multi-class labeling task.

Table 11 presents the inter-annotator agreement for judgments on answers to different types of questions in CAT-BENCH. The high Fleiss Kappa values demonstrate strong agreement between annotators and indicate reliability of our human evaluation framework.

D Prompts Used

We present the different prompts used with the benchmark models in Figure 12. All models use the answer-only (A) prompt. All models also share the (A + E) prompt except the Gemini models which use the NL (A + E) prompt instead. We found that Gemini was better at producing free-form natural language as opposed to a structured code format.

We used a temperature of 0.0 for all the experiments with each model to select the most likely token at each step, as this setting allow for reproducibility[10].

We use the following code snippet to query any OpenAI models.

```
import openai

client = OpenAI(api_key=config["OPENAI_API_KEY"])

response = client.chat.completions.create(
                model=openai_model_name,
                messages=prompt,
                temperature=0.0,
                max_tokens=2,
                top_p=1.0,
                frequency_penalty=0.0,
                presence_penalty=0.0
            )
```

We use the following code snippet to query any Gemini models.

```
import google.generativeai as genai

genai.configure(api_key=config["GEMINI_API_KEY"])

model = genai.GenerativeModel(args.model_name)
candidatecount, temp, topp, topk = 1, 0.0, 1.0, 1
generation_config = genai.GenerationConfig(
    candidate_count = candidatecount,
    max_output_tokens = args.max_tokens,
    temperature = temp,
    top_p = topp,
    top_k = topk
)
response = model.generate_content(prompt)
```

We run inference on `Llama3-8B` locally on one 40GB Nvidia A6000 GPU using HuggingFace (Wolf et al., 2020).

[10] We note that some researchers have shown that even this setting might not make it completely reproducible: https://twitter.com/ofirpress/status/1542610741668093952?s=46&t=f9v5k9RzVKnTK1e0UyauOA

	AVG	MODAVG	AVGBIN	MAJVOTE
Llama3-8B	3.77	3.14	0.70	75.42
gpt-4-turbo	3.95	3.39	0.77	79.58
gpt-4o	4.08	3.58	0.83	87.92
gemini-1.5-pro	3.98	3.8	0.77	87.08

Table 8: Human evaluation metrics for explanations generated by various models for DEP questions.

	AVG	MODAVG	AVGBIN	MAJVOTE
Llama3-8B	2.75	0.6	0.40	35.83
gpt-4-turbo	3.74	2.41	0.70	75.83
gpt-4o	3.6	2.29	0.66	71.67
gemini-1.5-pro	3.69	1.58	0.68	75.42

Table 9: Human evaluation metrics for explanations generated by various models for NONDEP questions.

	-2	-1	0	1	2
-2	1	$\cos \pi/8$	$\cos \pi/4$	$\cos 3\pi/8$	0
-1	$\cos \pi/8$	1	$\cos \pi/8$	$\cos \pi/4$	$\cos 3\pi/8$
0	$\cos \pi/4$	$\cos \pi/8$	1	$\cos \pi/8$	$\cos \pi/4$
1	$\cos 3\pi/8$	$\cos \pi/4$	$\cos \pi/8$	1	$\cos \pi/8$
2	0	$\cos 3\pi/8$	$\cos \pi/4$	$\cos \pi/8$	1

Table 10: Inter-class weights used for computing inter-annotator agreement

	Weighted Fleiss Kappa	Weighted Binarized Fleiss Kappa
DEP	0.808	0.941
NONDEP	0.705	0.934

Table 11: Inter-annotator agreement as measured by Fleiss Kappa for each question type in CAT-BENCH

Answer-only (A)	Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence. Goal: {title} Procedure: {procedure} Must Step {i} happen before Step {j}? Select between yes or no
Answer + Explanation (A+E)	Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence. Goal: {title} Procedure: {procedure} 1. Must Step {i} happen before Step {j}? Select between yes or no 2. Explain why or why not. Format your answer as JSON with the key value pairs "binary_answer": "yes/no answer to Q1", "why_answer": "answer to Q2"
Explanation + Answer (E+A)	Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence. Goal: {title} Procedure: {procedure} 1. Explain why or why not Step {i} must happen {temporal_relation} Step {j}. Think step by step. 2. Must Step {i} happen {temporal_relation} Step {j}? Select between yes or no Format your answer as JSON with the key value pairs "why_answer": "answer to Q1", "binary_answer": "yes/no answer to Q2"
NL Answer + Explanation (A+E)	Given a goal, a procedure to achieve that goal and a question about the steps in the procedure, you are required to answer the question in one sentence. Goal: {title} Procedure: {procedure} 1. Must Step {i} happen before Step {j}? Select between yes or no 2. Explain why or why not. Format your answer as follows: Answer 1: yes/no Answer 2: your answer in one sentence

Figure 12: Different prompts used for our experiment settings and models. i and j represent step numbers and $temporal\_relation$ can be before/after.