Plan Completion using Causal Links

By: Saatvik Sandal and Daniel Seo

1. Project Overview

Today LLMs are being relied upon more and more, with the popularity of models like GPT-3.5, GPT 4 Turbo, Gemini's 1.5 Pro, DistilBERT, etc seeing no signs of slowing down in being incorporated into people's lives both personally and professionally. LLMs are often used for information gathering, learning, and problem-solving. This increase in reliance means that LLMs must continuously improve and that people should be aware of any common limitations LLMs might have so that our reliance on these tools does not become hindrances. One such area in which we examine our LLMs for effectiveness is problem-solving. There are many types of problems and the different problem-solving techniques that an LLM must be proficient in. One such subset of this that we decided to focus on is the planning problem. To plan effectively, you must understand the role of each item in a plan, and how each component of the plan changes during the plan's duration. For example, in a cooking recipe, you must be able to extrapolate information about the changes ingredients go through to make a good plan and to correct any mistakes. We tested LLMs on this specific problem of understanding the state changes of entities in this project. Can an LLM understand the nuances of each entity's state in a plan? This is not an easy task because such a question requires an understanding of the model's ability to understand the changes each item goes through and the effect time has on each state. If steps are listed in order, because step 7 comes after step 1, does step 7 depend on step 1? Not always, and it is unclear if LLMs can grasp such relationships. We tested this by using corrupted plans, and asking an LLM if various steps are dependent on each other, with the modeling outputting a 0, meaning it is not dependent, and 1 meaning it is dependent.

This field of evaluation of LLMs is relatively small, with a few papers exploring plan efficiency and even fewer papers exploring temporal and state dependencies. Most approaches involve understanding the model's generated plan, rather than focusing on post-plan evaluations. We need to evaluate if models understand plans at the core of what makes a plan.

Our proposed ideas involve various ways of evaluating and experimenting with improving a model's understanding of temporal and state relationships so that a model can work with plans with consistency and granularity. We do this with 3 methods: fine-tuning, prompting, and reversed plan and graphical thinking. We specifically utilize corrupted plans to test a model's capability to understand how entities change in a plan by asking the model to classify if a pair of steps are dependent on each other, which forces the model to use context and state evaluations of the entity to fill in the gap of the plan.

Idea 1 is to fine-tune. We fine-tune DistilBERT and Gemini's flash model to a classification task of predicting if the two steps in question are dependent on each other. The goal of this is to make the model better at this task in order to improve its understanding of state and temporal dependencies, as that is what is needed to successfully answer this question.

Idea 2 is to prompt the model with an example of a step pair that is dependent. The idea is to do in-prompt training to the model and give an example of a dependent scenario as well as a

general outline of the logic to identify the state and temporal dependencies. We use the following: prompt = """

Consider the following example as a precursor to a question:

1. Gather ingredients: flour, eggs
2. [Missing Step]
3. Mix the whisked eggs with the flour.

In this example, for "whisked eggs" to be added to the "flour," the eggs must first be cracked and whisked.

Therefore, these steps are dependent because the state of the "eggs" must change before they can be mixed with the

"flour."

For the following plan and question, return ONLY 0 if the pair in question is non-dependent and 1 if the pair is dependent.

Again, your response should only be either a 0 or 1.

"""

Idea 3 is to reverse the plan, and then give the model a way to generally think about mapping out dependencies. The third plan is a little less straightforward, and a little more unorthodox. First was to reverse the plan. We can connect more ideas immediately by starting at the end of a plan. For example, in a two step plan “break eggs” “serve sunny side up”. The major difference is “open-mindedness”. You can make sunnyside eggs without salt and peppers. But if someone said to make sunnyside eggs, you immediately think “OK I need eggs… salt and pepper? Anything else?”. There is room for questioning and wonder in the reverse order. The last part of this idea is the graphical thinking method. We prompt the model to treat each step as a node, and then connect nodes to other nodes if there is a dependence one way or another. This was done to give a general way of thinking about the problem to the model, again in hopes of getting the model to think about temporal/state relationships. Here is the prompt for this idea:
prompt = """
Consider the following example as a precursor to a question:

1. Gather ingredients: flour, eggs
2. [Missing Step]
3. Preheat the oven to 350 degrees F.

In this example, pre-heating the oven to 350 degrees F has no dependency on us gathering the ingredients flour and eggs.

For the following plan and question, return ONLY 0 if the pair in question is non-dependent and 1 if the pair is dependent.
Again, your response should only be either a 0 or 1.

"""

For each idea, we will evaluate the model using the basic metrics such as accuracy, precision, recall, and F1 scores. Furthermore, in this projec,t we specifically looked at the number of each guess (dependent vs non-dependent), as well as the distributions and order of the guesses per each idea. We utilize these metrics by comparing them to each other, using the base

model with no special prompt as a baseline. We hope to evaluate its understanding of dependencies, not just creating plans from its baseline knowledge. These ideas focus on testing this, especially by focusing on the way of thinking in the prompts. In this project we solely use a cat bench. Utilizing python, the Gemini API, and Hugging Face for DistilBERT, as well as other data related libraries to deal with data manipulation to prepare the models for evaluation.

To summarize our findings, we found that the fine-tuned model performed the best, as it had the highest F1. Then the dependent example prompt, and then the reverse plan and graph prompt. The dependent example prompt showed a bias towards guessing dependent, which was confirmed through further analysis of the results and comparison to the other ideas. The reverse plan and graph idea was more balanced and a better alternative to the one-shot dependent example.

2. Ideas

2.1 Idea 1:

Fine-tune pre-trained models to a modified cat-bench dataset that forces the model to classify question pairs on whether they are dependent or not while dealing with a missing step in each sample in the cat-bench.

2.2 Idea 2:

Provide an example of a plan where the steps asked about depend on each other.

2.3 Idea 3:

Reverse the plan (Step 8, Step 7, etc…) and prompt the model to think of it graphically, where each step is a node, and if the next step is dependent on another step, they should connect the nodes.

3. Experimental Setup

3.1 Models:

Base Model: DistilBERT

- **Type:** Transformer-based language model
- **Pre-training:** Pre-trained on a massive text dataset using a distillation process from BERT
- **Fine-tuning:** Fine-tuned on the modified Cat-Bench dataset for 3 epochs with a batch size of 4 and a learning rate of 5e-5. We used the Adam optimizer.

Base Model: Gemini 1.5 Flash

- **Type:** Multimodal language model
- **Pre-training:** Pre-trained on a diverse dataset including text, images, audio, and video
- **Fine-tuning:** Fine-tuned on the modified Cat-Bench dataset for 5 epochs with a batch size of 16 and a learning rate of 0.0002.

3.2 Dataset

Dataset: Cat-Bench

- **Input/Output Format:** Each instance consists of a prompt (a sequence of instructions) and a response (the desired output).
- **Training Instances:** 10,000 instances
- **Test Instances:** 2,000 instances

Modified Dataset:

- Created by randomly removing a step from each prompt in the original training set. The randomly removed step was made sure to occur before the last step in the step pair that was asked about, and made sure that the step pair being asked about was excluded from being possibly removed.

3.3 Evaluation Metrics

- **F1-Score:** This metric considers both precision and recall, making it a balanced measure of model performance. A higher F1 score indicates better overall performance.
- **Accuracy:** This metric measures the proportion of correctly classified instances.
- **Precision:** This metric measures the proportion of positive predictions that are correct.
- **Recall:** This metric measures the proportion of positive predictions to all positive and negative truths.

4. Results

Base Model Results:

Base Model	Accuracy	Precision	Recall	F1 Score
Gemini 1.5 Flash	0.599	0.616	0.549	0.580
DistilBert	0.4717	0.2687	0.0258	0.0471

Idea 1 Fine Tune Results:

Model	Accuracy	Precision	Recall	F1 Score
Fine-tuned Gemini 1.5 Flash	0.936	0.934	0.940	0.937
Fine-tuned DistilBert	0.676	0.699	0.661	0.743

Idea 2 One-shot Dependent Example Results:

Base Model	Accuracy	Precision	Recall	F1 Score

Gemini 1.5 Flash	0.644	0.603	0.865	0.711
DistilBert	0.621	0.594	0.791	0.679

Idea 3 Reverse and Graph Plan Results:

Base Model	Accuracy	Precision	Recall	F1 Score
Gemini 1.5 Flash	0.683	0.682	0.699	0.690
DistilBert	0.614	0.578	0.885	0.699

From these results we can see that Idea 1, fine-tuning, was the most effective tool to improve model performance on cat-bench, which boasted significant improvements for both models. Both models saw an increase for idea 2 and idea 3 from the base model, but idea 2 shows less balance between precision and recall for idea 2. The imbalance between these two metrics were also worsened in idea 3 for DisitilBERT only, and also had a better F1 score than Gemini.

5. Analysis and Discussion

Based on the results in the section above specifically for the Gemini model, I came up with 3 hypotheses:

1. The dependent one-shot example is biased towards guessing dependency. This would also explain the increase in score and doesn't show the model's improvement in temporal and state dependency understanding.

2. The reverse plan is more balanced, gets the same questions right as the one-shot but detects false negatives better.

3. The one-shot bias would be fixed or changed if we made it one-shot with a non-dependent example rather than a dependent example.

Hypothesis 1:
The recall in idea 2 was 0.865 compared to its precision of 0.603, which indicates that during testing for idea 2 the model would guess positive much more often. Additionally, the success rate of this is a little worse than the base model with no prompt the (precision of the base model with no example is 0.616, the precision of the base model with an example is 0.603).

Despite giving an example of a dependent pair of steps in a plan, the precision slightly decreased, meaning that its ability to correctly identify dependent pairs did not increase and that the model simply guessed dependent more often (seen in the table below).

Idea 3 did not follow this pattern. It guessed dependent less often, but the precision improved by ~7%. The difference in the # of guesses between idea 3 and the base model for Gemini is 40 guesses, which is 2% of the samples, meaning that the increase in precision and recall is actually a significant improvement.

Hypothesis 2:
To test if the reverse plan is more balanced than the one-shot in idea 2, we can look at the precision, recall, and the # of dependent guesses. We can see in the table below that the number of dependent guesses is much less than idea 2 and that since its precision is higher by ~8%, it is more "skillful" than idea 2. In regards to testing, if it gets the same questions right, it is hard to tell. In the figure below labeled "Dependency Guess Occurrences", in the dependent section, we can see commonalities. Using other calculations, we found that Idea 2 and Idea 3 overlapped in the same guesses of dependent steps 623 times. Within the true dependent samples, ideas 2 and 3 overlapped 452 times, and the majority of the dependent guesses were in true dependent samples, which matters more for idea 3 since it was much more balanced.
In short, yes, the reverse plan is more balanced.

Hypothesis 3:
To test this hypothesis I changed the prompt to:

prompt = """
Consider the following example as a precursor to a question:

1. Gather ingredients: flour, eggs
2. [Missing Step]
3. Preheat the oven to 350 degrees F.

In this example, pre-heating the oven to 350 degrees F has no dependency on us gathering the ingredients flour and eggs.

For the following plan and question, return ONLY 0 if the pair in question is non-dependent and 1 if pair is dependent.
Again, your response should only be either a 0 or 1.

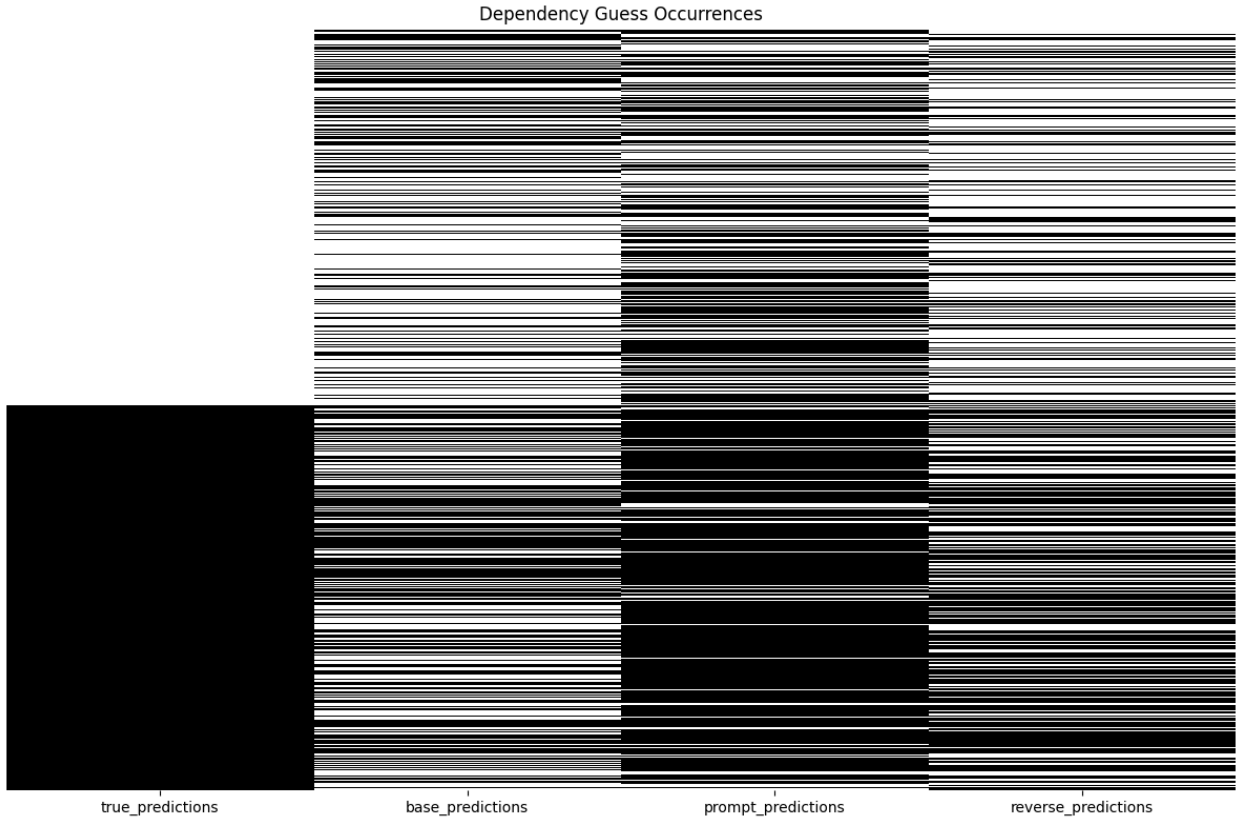
"""

I tested 50 samples in the cat-bench testing set that were all non-dependent, to see if the bias improves. The results are as follows: in the original dependent example, 70% of the guesses were dependent in the first 50 samples of non-dependent step pairs. In the new non-dependent example prompt, 56% of the guesses were dependent, a 14% improvement. Of course, the sample size can be bigger to test this. Still, considering the results from the previous hypothesis, it is acceptable to conclude that the bias improved because of the change of examples.

Answer Frequency Counts per Idea for Gemini Flash Model:

Idea / Model	Number of Dependent Guesses	Number of Non-dependent Guesses
Correct Answers (Truth)	698	682
Base Model (no idea)	622	758
Dependent Prompt Examples (Idea 2)	1001	379
Reverse and Graph (Idea 3)	716	664

Distribution Alignment of Dependent Guesses for Gemini Flash Model



6. Code

Link to source code: https://github.com/Saatvik1/LLM-Causal-Dependency-Study
Link to dataset (CAT-Bench) :
https://drive.google.com/file/d/1UjFwv-BKU0Q7CRAFFEFsMX2AUtoT5Wud/view?usp=sharing

Software Requirements:
datasets==3.0.0
transformers==4.31.0
matplotlib==3.7.1
git-lfs
google-generativeai
jsonlines
python-dotenv
torchmetrics==0.11.4
pandas==2.0.3
sklearn
scipy

To run the notebook containing the Gemini tests, you must create your own API key from Google AI Studio to call Gemini model endpoints.

7. Learning Outcomes

We learned about the extent to which LLMs like Gemini and DistilBERT can understand the underlying phenomena in planning. It is clear that an LLM is far from the level a human is at understanding and tracking entity relationships, and that LLMs are very sensitive to factors that may cause bias, as is shown in the results of idea 2. This has certain applications in planning tasks and should be thought about when utilizing or training a model in a planning domain. We also learned that it is possible to influence the model to think about how to understand these relationships, as shown by idea 3, where the prompt of teaching the model how to think improved the model's performance without causing an imbalance between precision and recall. Although this was successful, there's much room for improvement. In regards to fine-tuning, it seems to be very promising, but there needs to be more study on how effective this method is in teaching the model the underlying phenomena, specifically by testing it with a variety of plan domains.

8. Contributions

Saatvik Sandal - Gemini workflow.
Daniel Seo - DistilBERT workflow.
Both - Project write-up, and brainstorming.

Citations

Lal, Yash Kumar, et al. "CaT-BENCH: Benchmarking Language Model Understanding of Causal and Temporal Dependencies in Plans." *arXiv*, 22 Nov. 2024, https://doi.org/10.48550/arXiv.2406.15823.