Leveraging Belief Graphs to Investigate the Reasoning Capabilities of Large Language Models

SIBI RAJA, Harvard University, USA
NADA AMIN, Harvard University, USA

In this work, we delve into the application of Large Language Models (LLMs) reasoning capabilities for solving complex mystery puzzles, specifically within the context of the Minute Mysteries QA task from the BIG-Bench evaluation suite. This research draws on the REFLEX framework presented in [Kassner et al. 2023], adapting it to enhance the reasoning abilities of LLMs such as Llama 13b and 70b. Our approach involved constructing belief graphs based on LLMs' confidences of statements that indicate a particular suspect's guilt, which are then used to determine the true culprit within a given mystery story. These beliefs were then used to construct a belief graph that was then processed using a Maximum Satisfiability (Max-SAT) solver, which allows for a culprit to be determined based on the inherent beliefs an LLM held. We also investigate enhancing the construction of a belief graph through techniques such as Chain-of-Thought prompting, using more descriptive prompts, incorporating additional logical constraints, and representing more entities within the belief graphs. Despite these enhancements, our findings indicate minimal improvement in the performance of LLMs in the Minute Mysteries QA task. Nonetheless, this research contributes to the understanding of the limitations of LLMs in complex reasoning tasks and motivates future exploration in bridging the gap towards Artificial General Intelligence. Our code is available at github.com/metareflection/llm-mysteries

Additional Key Words and Phrases: Large Language Models, Chain-of-Thought prompting, fine-tuning, zero/one/few-shot learning, hallucinations, Maximum Satisfiability

1 INTRODUCTION

Large Language Models (LLMs) have taken the world by storm in the past year, especially since OpenAI's launch of Chat GPT in November 2022 [OpenAI 2022]. However, several research efforts have gone into advancing and evaluating the performance of LLMs for the past few years, extending before the launch of systems like ChatGPT. Within these research efforts, the reasoning abilities of LLMs have been of particular focus to investigate how LLMs may be able to reach levels of human intelligence, motivating the search for creating Artificial General Intelligence (AGI).

One can experiment with inputting different types of prompts to LLMs to see that prompting techniques themselves can illicit different behaviors among LLMs. Straightforward yet potentially effective prompting techniques include one-shot and few-shot learning, which are in contrast to zero-shot learning. Few-shot learning refers to including example inputs and outputs as part of the prompt to an LLM for a given task. One-shot learning refers to including just one input-output pair, while zero-shot learning includes prompting with no prior input-output examples. Chain-of-Thought (CoT) prompting builds upon the one-shot and few-shot learning frameworks by prompting with few-shot examples that contain a series of reasoning steps as part of the correct output example with the intent of getting the LLM to simulate an artificial thought process before outputting an answer [Wei et al. 2022]. Wei et al. apply this prompting technique to five datasets containing mathematical word problems that require reasoning skills to solve and found that CoT greatly improves the performance of LLMs in these reasoning tasks.

In addition to prompting, [Brown et al. 2020] from OpenAI showed in the original GPT-3 technical report that scaling up LLMs in size through pre-training has the potential to improve the performance of LLMs, but other work from [Zhong et al. 2021] have shown that this does not necessarily hold in every setting. Indeed, they state that larger language models do not necessarily have better performance on every trial than their smaller counterparts, observing this to be the case in at least

1 – 4% of instances (which is a relatively small yet non-negligible portion). While research results do seem to be mixed, the overall consensus within the LLM community is that much larger language models do tend to outperform smaller ones.

2 PROBLEM MOTIVATION

To begin investigating the capabilities of LLMs to reason well, we first looked into assessing their abilities to generate verifiable code to prove mathematical theorems. Our first set of informal experiments consisted of vanilla prompting GPT-3.5 and GPT-4 (via the Chat GPT interface) as well as Llama v2 13b and Llama v2 70b (for the rest of this paper, we will be referring to Llama v2 13b and Llama v2 70b as Llama 13b and Llama 70b for sake of brevity) [OpenAI 2023] [Touvron et al. 2023]. These did not produce promising results, with the LLMs often generating incorrect code. From a preliminary overview, GPT-4 seemed the strongest, but this was to be expected as it is the largest in raw parameter count out of the four models. Thus, we turned to other techniques to improve model performance.

Because the foundation model we experimented with may not have had enough exposure to a relatively rare domain like Coq programming, we fine-tuned Llama 13b and Llama 70b on CoqGym [Yang and Deng 2019], a dataset consisting of Coq code to prove several mathematical theorems. At a cursory glance, fine-tuning only seemed to produce little improvement, if any.

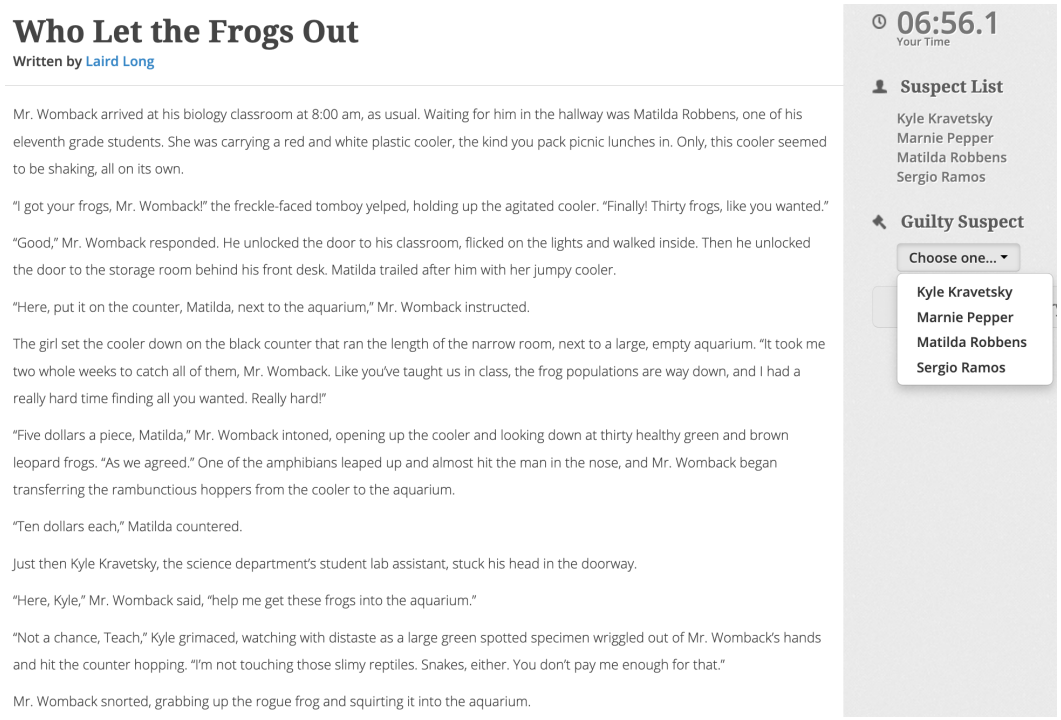


Fig. 1. Excerpt from a mystery taken from the 5MinuteMystery Website

From these informal experiments alone, it was clear that LLMs severely lack complex reasoning capabilities. This was the initial motivation to shift our focus from generating verifiable code in

Coq to investigating general reasoning capabilities of LLMs. We turned to the Beyond the Imitation Game Benchmark (BIG-bench), which consists of over 200 tasks to assess LLMs in [Srivastava et al. 2023]. Out of these 200+ tasks, we focused on Minute Mysteries QA, which is a Question-Answering task where given a short story representing a fictional crime or mystery, the goal is to correctly identify the suspect from a list of suspects. Figure 1 shows a portion of a mystery that is included in the Mysteries QA dataset, with the web interface being from the 5MinuteMystery website.

We ran an initial set of benchmarks on the Minute Mysteries QA dataset (total of 203 mysteries) using the same GPT and Llama models mentioned earlier as well as Llama v2 7b (a smaller model part of the Llama 2 model series) and Mistral 7b (a SOTA 7b LLM), which we show below.

Large Language Model	Mysteries Correctly Solved
GPT-4	**82/203**
GPT-3.5 (175 B)	55/203
Llama 70b	46/203
Llama 13b	47/203
Llama 7b	37/203
Mistral 7b	38/203

Table 1. Testing the performance of various LLMs on the Minute Mysteries QA dataset

We can see from Table 1 that GPT-4 outperforms all of the other LLMs by a significant amount, having nearly 30 more mysteries solved than the second highest performing model, GPT-3.5. This observation is to be expected, as it supports the finding from the previous works mentioned that larger models tend to have stronger performance than smaller ones (note, at the time of these experiments, OpenAI has not publicly released the parameter count of GPT-4, although the model is speculated to contain roughly 1.76 trillion parameters). After the GPT series models, the Llama 70b and Llama 13b seem to have similar performance to one another, which seems to undermine the performance increase with scale claim. However, Llama 7b and Mistral 7b performed the worst out of the LLMs we tested, which could provide additional support to the performance with scale claims. Nonetheless, it is clear from these initial benchmarks that smaller models have lots of room for improvement to perform as well as larger models like GPT-4.

3 EXPERIMENTAL APPROACH

To enhance the reasoning capabilities of LLMs on the Minute Mysteries QA task, we drew inspiration from current literature on the reasoning abilities of LLMs. [Kassner et al. 2023] apply the REFLEX framework to LLMs and show more robust chains of reasoning being produced by LLMs in common sense reasoning tasks. Their main approach consists of modeling an LLM's beliefs in a belief graph by taking the probabilities associated with individual logits and using those to represent confidences for each belief (i.e. a statement), placing logical constraints on each belief, and solving the belief graph via an external SAT solver to produce the optimal solution. After querying an LLM initially to generate hypotheses for a given question, Kassner et al. construct a belief graph by recursively asking the LLM for explanations (and refutations) for such hypotheses. These explanations are modeled as nodes in the belief graph, in addition to the initial set of generated hypotheses, and logical constraints are then placed between nodes. Note that there are two types of constraints: soft and hard. For soft constraints, a penalty is specified that will be incurred if the constraint is violated. On the other hand, hard constraints are unbreakable constraints that always must be adhered to (often done by setting a penalty of ∞). From here, the belief graph can

be reduced to a Maxmimum Satisfiability (Max-SAT) problem and solved via an external SAT solver [Battiti 2009].

For example, if an LLM believes a statement X is true with probability p, a soft constraint is placed between the belief node for that statement and the belief node corresponding to the negation of that statement, which signifies that exactly one of two nodes may be true. The costs for breaking each constraint are proportional to the LLMs' beliefs. Note that the penalty for violating a statement's negation – which is equivalent to choosing the statement itself – should be high if the LLM had low confidence in the corresponding belief and vice versa. To capture this relationship, Kassner et al. used $C(x) = e^{-p(-x)}$, where $C(x)$ represents the function to calculate the confidence score of a statement x, and $p(-x)$ represents the probability of a logit generated by the LLM for the negation of a statement. Furthermore, in a multiple-choice setting, a hard XOR constraint with ∞ penalty is placed between each of the answer choices to ensure that exactly one answer is picked. This way, a singular answer is chosen based on which constraint is cheaper to break within the Max-SAT reduction. And indeed, the REFLEX framework was found to create self-consistency within a model's generated beliefs for answering a task by reducing logical inconsistencies that may have been present.

Algorithm 1 Constructing Belief Nodes

Require: Suspect list
1: What nodes $W \coloneqq$ { mean, motive, opportunity, guilt }
2: List of belief nodes $B \coloneqq \{\}$
3: **for** $s \in$ Suspect list **do**
4: **for** $w \in W$ **do**
5: $X \coloneqq$ "$\{s\}$ has $\{w\}$: True or False?"
6: Query LLM with X and generate obtain probability $\{p\}$
7: Construct belief nodes: $b_1 : (s, w, T, p)$, $b_2 : (s, w, F, 1 - p)$
8: $B \leftarrow B + b_1 + b_2$
9: **end for**
10: **end for**
Ensure: List of belief nodes B

We adopt and apply the REFLEX framework for the Minute Mysteries QA task using the steps presented above in Algorithm 1. We also leveraged the Outlines Library [Willard and Louf 2023] to create a custom greedy sampling strategy to query an LLM and only focus on the tokens for the words "True" and "False". We queried each LLM with the following prompt structure: X + mystery + X for each question X. After all the belief nodes for a given mystery were generated, we then constructed a belief graph consisting of the following layers of nodes:

(1) Suspect nodes
(2) A what node pertaining to each suspect
(3) A node representing the confidence scores for the truths of a specific what node statement

A visualization of the three-layer belief graph is shown below in Figure 2. After constructing a belief graph that models the LLM's beliefs for each hypothesis pertaining to each suspect, we then convert the graph into a MaxSAT problem. This was done by generating two types of boolean variables:

(1) A suspect being guilty (i.e. Bool(Suspect X is guilty))
(2) A what node statement (i.e. Bool(Suspect X has motive))

For each boolean variable representing a what node statement for each suspect, we place a soft constraint on violating its negation (violating its negation is the same as choosing the statement itself) and violating itself. Note that we calculated a confidence score c of a statement X by using the exponential function in the same manner as in Kassner et al, which we mentioned earlier. Finally, we utilized an external SAT Solver from the Z3 library in Python to find the optimal solution to the constructed model with constraints.

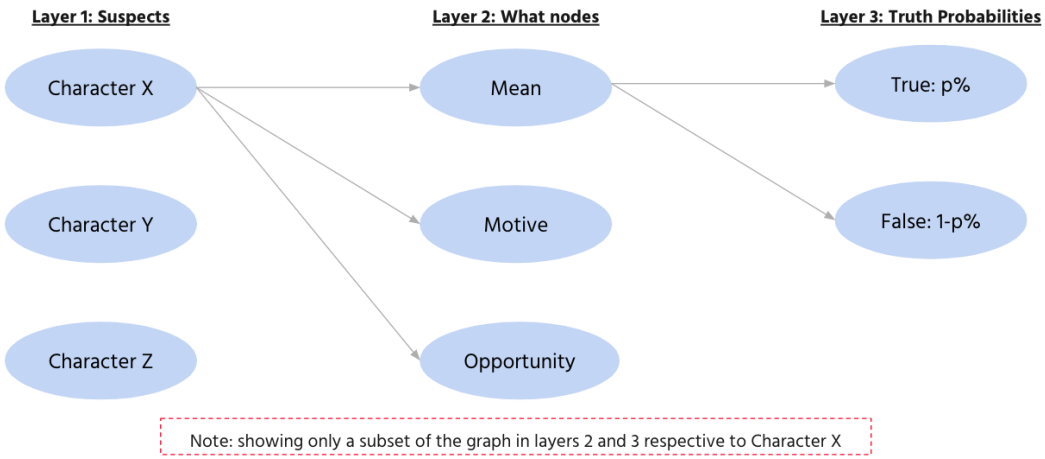


Fig. 2. Three-Layer Belief Graph for Minute Mysteries QA

4 INITIAL RESULTS

We applied our belief graph framework to Llama 70b and Llama 13b to solve the mysteries present in Minute Mysteries QA. Our results are shown below in Table 2. We observe that applying the belief graph framework produced slight improvements in both Llama 70b and Llama 13b, but the results produced are still not great, as an approximate solve rate of 25% is equivalent to the expected portion of mysteries to be solved if one were to guess randomly (as there are only four choices of suspects).

Large Language Model	Mysteries Correctly Solved
Llama 70b	**52/203**
Llama 13b	**52/203**

Table 2. Belief Graph-Augmented LLM Performance on the Minute Mysteries QA Dataset

Note that we did not run these initial tests on other LLMs because of our intent to test our initial implementation first and ideally see the potential for improvement in LLM reasoning before running several trials on other models.

5 ADDITIONAL FEATURES TO ENHANCE LLM REASONING

Because the belief graph framework did not produce much success compared to the vanilla approach, we turned to implementing additional features to our belief graph approach with intentions of inducing increased reasoning capabilities among the LLMs. In the following subsections, we discuss each of these additional features that we integrate within our belief graph framework, and we follow up with the results produced from integrating each of these features in the Final Results section.

5.1 More Descriptive Prompting

Previous work from [Chen et al. 2023] and [Reynolds and McDonell 2021] have already shown that different types of prompting can cause LLMs to produce different outputs. A simple yet potentially powerful change we implemented was querying LLMs with more descriptive prompts when generating belief nodes for a given mystery. Our original prompts to find the confidence that an LLM had were of the following format:

(1) Suspect X is guilty: True or False?
(2) Suspect X has {mean, motive, opportunity}: True or False?

We improve upon our prompting by instead using more descriptive prompts that contain instructions specific to each what node query we send to an LLM. The motivation behind this was that using different prompts for each what node may allow the LLM to better differentiate between clues for each type of what node. The prompts are presented below in Table 3:

What-node	Prompt
Guilt	"Consider all the evidence and context from the story. Is this conclusion about guilt reasonable? Answer with either 'True' or 'False' only. Is the statement True or False?"
Mean	"Reflect on the resources and tools available to the suspect as described in the story. Does this suspect have the means necessary for the crime? Answer with 'True' or 'False' only. Is the statement True or False?"
Motive	"Consider the suspect's potential reasons and incentives as depicted in the story. Does this suggest a motive for committing the crime? Answer with 'True' or 'False' only. Is the statement True or False?"
Opportunity	"Analyze the timeline and the suspect's whereabouts as narrated in the story. Did this suspect have the opportunity to commit the crime? Answer with 'True' or 'False' only. Is the statement True or False?"

Table 3. List of What-nodes and Prompts

5.2 Parsing Individual Sentences

Another prompting modification we made was not passing in the entire mystery as part of each query to an LLM for generating a belief node but rather passing in one sentence at a time. Our motive behind this change was that LLMs may get confused with the large amount of text present in a given mystery story when generating beliefs for a single suspect. We used the same approach as before but added an extra loop in 1 by iterating through each sentence within a mystery story. If M represents the entire text in a mystery story and X represents a statement of the form presented in line 5 of Algorithm 1, the query passed to an LLM is no longer $X + M + X$ but rather $X + m_i + X$,

where m_i is a singular sentence in M.

We acknowledge that this can cause duplicate belief nodes to be generated for each what node statement for a particular suspect. However, we handle this with the function $p(b) = max(p_{old}(b), p_{new}(b))$, where $p(b)$ is the final probability that we keep for a belief node b, p_{old} is the currently set probability for b, and p_{new} is the new probability we have just encountered for b via a duplicate belief. Note that if p_{old} does not exist, we simply just replace it with 0 to avoid any errors from using the *max* function. By taking the maximum probability from a previously encountered belief node (if we have already encountered the same belief node) and the current belief node we are trying to process, we ensure that we retain the stronger belief that an LLM may have.

To implement prompting via individual sentences, the initial thought was to simply use the .split(".") via Python. However, this began to fail on several common edge cases such as when dealing with suspects whose names contain prefixes such as "Mr." as well as sentences ending with punctuation that is different from periods. To work around this, we used Spacy, a common NLP library known to handle sentence extraction from texts fairly well.

5.3 Adding Negation Nodes to the Belief Graph

To have a more robust belief graph model, we added negation nodes. This consisted of adding an additional step to the algorithm presented in Algorithm 1 to also query an LLM for its belief on a statement $\neg X$. This leads to twice as many belief nodes being generated, and a fourth layer being created in the belief graph framework. However, the rest of our approach still remains as before. We create additional boolean variables for each negation statement $\neg X$ in the Z3 SAT construction. Furthermore, while we still keep the same constraints as our original SAT model construction, we also place a soft XOR constraint between each pair of boolean variables S and $\neg S$ to select only one of them as true when producing a final answer.

5.4 Soft Constraints from What-nodes to Guilt Nodes

An additional modification to our Z3 model included imposing soft constraints between each suspect's what node and their guilt node. As we mentioned before, we have two types of boolean variables (refer to the bottom of page 5). In addition to the original logical constraints placed (we are not including the negation nodes and the additional XOR constraints discussed in the previous subsection), we add a soft implication constraint for a what node for a suspect implying that a particular suspect is guilty. Our motivation for doing so was to connect the potential clues that may be present for each suspect more directly with the final decision of determining whether or not a suspect is guilty. Because an implication constraint is inherently different from an XOR constraint, we changed the calculation of the confidence function to be $C(x) = e^{p(x)-1}$, where $p(x)$ is the confidence of the statement x itself being true – meaning, we no longer use the probability of x being false. The reason behind this change is that when an implication $X \implies Y$ is violated, a cost that is proportional to the belief of X itself being true will be incurred. We introduce a -1 term in the exponent to ensure that the same range of values as the original confidence function used in the XOR constraints is produced. Please refer to Section 10.1 in the Appendix for a clear visualization of this mathematical change. Also note, this change in the $C(x)$ is a different formulation than what is presented in Kassner et al., and we acknowledge that we could implement this same feature with the cost function they propose.

6 FINAL RESULTS

We present the results from the experiments conducted using the additional features discussed in the previous section below, with N/A entries representing trials not being conducted for that particular experiment configuration:

Additional Features on Belief Graph	LLM	Mysteries Solved
Descriptive Prompts	Llama 70b	40/203
	Llama 13b	N/A
Parsing Individual Sentences	Llama 70b	N/A
	Llama 13b	51/203
Negation Nodes	Llama 70b	42/203
	Llama 13b	40/203
Soft Constraints (S.C) from What to Guilt Nodes	Llama 70b	52/203
	Llama 13b	52/203
Descriptive Prompts + S.C from What to Guilt Nodes	Llama 70b	37/203
	Llama 13b	**55/203**

Table 4. Enhanced Belief Graph-Augmented LLM Performance on the Minute Mysteries QA Dataset

Table 4 shows some unexpected yet interesting results. Overall, we observe that the performance of the Llama models varies heavily from experiment to experiment. The best-performing configuration of the belief graph framework was enhancing it with descriptive prompts and soft constraints from the what nodes to the guilt nodes for each suspect and applying it to LLama 13b. Surprisingly enough, this configuration also produced the worst results among the set of experiments when applied to LLama 70b. This was not expected as not only does this particular configuration combine several techniques to illicit higher reasoning capabilities in an LLM, but it also uses the largest version of the Llama models, contradicting the notion of performance increase with scale.

We also observe that out of the four total experiments ran with Llama 70b, none of them produced an increase in mysteries solved compared to the original belief graph approach we employed. The most successful run was the Soft Constraints from the What to Guilt Nodes experiment, which allowed Llama 70b to solve 52 out of 203 mysteries – which is equivalent to the number of mysteries it solved using the original belief graph framework. All other experiments produced worse results, which indicates that the reasoning enhancements applied to the belief graphs framework were largely unsuccessful. Compared to the vanilla results that did not use a belief graph, only the Soft Constraints from What to Guilt Nodes experiment produced better results. The other three experiments caused a decrease in the number of mysteries solved by Llama 70b.

Llama 13b seems to not perform as poorly as Llama 70b with these additional features applied to the belief graph framework. Out of the four experiments conducted, two of them either match or produce better performance than the original belief graph framework. The Parsing Individual Sentences technique causes a slight decrease in performance in Llama 13b, with one less mystery being solved, and integrating Negation Nodes within the belief graph causes twelve fewer mysteries to be solved. The results gathered on Llama 13b are far from promising, but overall it seems to roughly match or outperform Llama 70b in each experiment (among experiments in which both models were tested). Interestingly, this finding can also be observed in Tables 1 and 2, which indicates the that performance of Llama 13b compared to Llama 70b is relatively consistent throughout our

conducted experiments.

7 RELATED WORK

Previous works on improving reasoning capabilities in LLMs have seen some success in eliciting stronger performance than using vanilla LLMs. A common theme in recent LLM reasoning literature is the use of Knowledge Graphs (KGs), which are graphical representations of information [Singhal 2012]. The nodes within a KG represent entities of a given context such as subjects, states, etc., and the edges between nodes attempt to capture the relationships between nodes.

[Sen et al. 2023] showed that augmenting LLMs with KGs in the task of Question-Answering allowed smaller LLMs to reach the performance of vanilla larger LLMs. Their approach consisted of using a static KG based on Wikidata to send the top k triples to an LLM, which then chose a final answer for the given question. This work differs from ours in that the LLMs tested in their work include models from the Flan-T5 series, which are several magnitudes smaller than the LLMs we tested (hundreds of millions of parameters compared to billions of parameters). Additionally, the use of an external KG does not aim to directly increase the reasoning capabilities of LLMs but rather constrict reasoning to certain subsets of potential outputs.

[Misra et al. 2023] investigated using KGs with LLMs but in a different manner, using prompt-tuning to train LLMs to traverse the information present in KGs effectively. They ultimately found that training larger models such as Flan T5 11b on random walks led to a significant increase in performance on 2-hop reasoning Question-Answering tasks compared to using such an approach on smaller models such as Flan T5 770m. However, a limitation of their work includes only investigating 2-hop reasoning problems, which are significantly less complex than the mysteries we investigated as part of the Minute Mysteries QA dataset.

Additional approaches in investigating reasoning in LLMs include using a Hypothesis-to-Theories framework to generate a hypothesis and maintain a list of verified hypotheses (i.e. rules) in solving additional tasks in a given task domain as per [Zhu et al. 2023]. Furthermore, there have been extensions upon the CoT framework such as [Yao et al. 2023] Tree of Thoughts (ToT) and [Besta et al. 2023] Graph of Thoughts (GoT) that have shown strong potential. Each of these works has been shown to induce better performance of LLMs in reasoning tasks. ToT and GoT allow for further freedom in an LLM's reasoning process while maintaining structure, which is in the motivation of more closely mimicking human reasoning. The Hypothesis-to-Theories framework was motivated in a similar fashion, attempting to draw a parallel between LLM hallucinations and humans generating hypotheses in scientific settings, by letting LLMs accept or reject hallucinations that are produced similar to how human scientists experiment and test theories. Our work differs from these though, as the task domains studied by these works (sorting numbers, mathematical calculations) are not of as high complexity in common sense reasoning as the Minute Mysteries QA task that we investigate.

8 FUTURE WORK

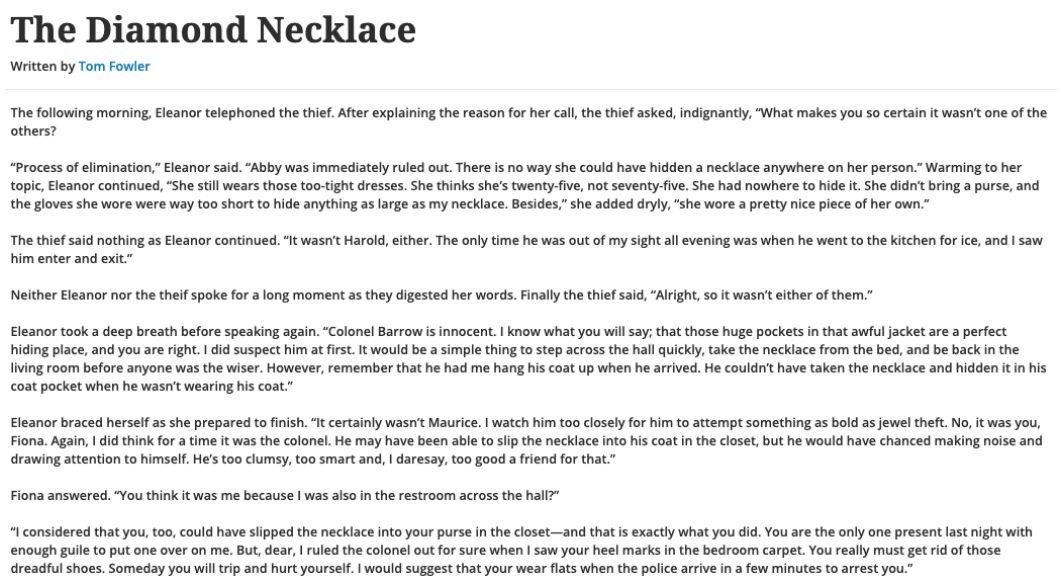
Immediate next steps for our work include running experiments using the original belief graph approach as well as our proposed enhancements to it on different LLMs such as the GPT-series and Mistral. While applying the belief graph framework to Llama 13b and 70b did not produce much success, our gathered results may be due to a flaw in our implementation, a performance ceiling on the Llama models, or a combination of both. Running experiments with stronger models such

as GPT-4 could provide helpful insight into this, although access to GPT-4 is only permitted by incurring a financial cost.

Additionally, employing a temporal logic-based approach in solving mysteries may allow for stronger performance of LLMs because reasoning among chronological tasks seems to be necessary for solving several mysteries within the Minute Mysteries QA dataset (based on manual inspection). We currently pose logical constraints representing the relationships between statements regarding a suspect's guilt or clues that could indicate guilt (what nodes), but building a timeline of events that occur within a given mystery could allow for a more structured reasoning process to take place.

Furthermore, inspiration from traditional NLP tasks could be applied to enhance the individual steps present within our belief graph framework. We currently ran a configuration of the belief graph experiment that included prompting with a singular sentence at a time with the intent to reduce confusion that may occur within an LLM as it parses a large body of text. Techniques such as event extraction could be applied to pre-process a given mystery, and the extracted events could be passed as part of the prompt to an LLM [Chen et al. 2019]. This may improve performance as the LLM no longer has to comprehend information to as high of a degree if the necessary events to reason about are already inputted in a clear manner. Within works that have studied event extraction in NLP, specific techniques pertaining to forensic text extraction could be of particular interest for us to integrate, as the nature of forensic texts can be thought of as being similar to the fictional mysteries present within the Minute Mysteries QA stories [Rodrigues et al. 2022]. Applying specialized event extraction techniques may allow LLMs to reason about crucial information only rather than having to parse unimportant information that is often present in a mystery (and risk confusion or hallucinations).

Furthermore, a quick feature that can be added to our belief graph framework is in our prompting approach. Drawing from the CoT approach presented by Wei et al. (2022), we can change our prompting to include 1-shot learning. As already mentioned in the Introduction, research has shown that 1-shot learning and few-shot learning have been successful in inducing a thought process that can aid LLMs to solve tasks that require reasoning to solve. Because the Minute Mysteries QA dataset does not have elaborate solutions beyond just the culprit name for each mystery, we can turn to the 5MinuteMysteries website, which contains the mysteries present in the Minute Mysteries QA dataset along with comprehensive solutions that describe the thought process that one can take to take to reach the conclusion that a particular suspect is the culprit. Solutions to mysteries are written by humans who wrote the original mysteries. A solution to an example mystery is presented below in Figure 3. Please refer to the Appendix for an additional note on experimenting with CoT.



The Diamond Necklace

Written by Tom Fowler

The following morning, Eleanor telephoned the thief. After explaining the reason for her call, the thief asked, indignantly, "What makes you so certain it wasn't one of the others?

"Process of elimination," Eleanor said. "Abby was immediately ruled out. There is no way she could have hidden a necklace anywhere on her person." Warming to her topic, Eleanor continued, "She still wears those too-tight dresses. She thinks she's twenty-five, not seventy-five. She had nowhere to hide it. She didn't bring a purse, and the gloves she wore were way too short to hide anything as large as my necklace. Besides," she added dryly, "she wore a pretty nice piece of her own."

The thief said nothing as Eleanor continued. "It wasn't Harold, either. The only time he was out of my sight all evening was when he went to the kitchen for ice, and I saw him enter and exit."

Neither Eleanor nor the theif spoke for a long moment as they digested her words. Finally the thief said, "Alright, so it wasn't either of them."

Eleanor took a deep breath before speaking again. "Colonel Barrow is innocent. I know what you will say; that those huge pockets in that awful jacket are a perfect hiding place, and you are right. I did suspect him at first. It would be a simple thing to step across the hall quickly, take the necklace from the bed, and be back in the living room before anyone was the wiser. However, remember that he had me hang his coat up when he arrived. He couldn't have taken the necklace and hidden it in his coat pocket when he wasn't wearing his coat."

Eleanor braced herself as she prepared to finish. "It certainly wasn't Maurice. I watch him too closely for him to attempt something as bold as jewel theft. No, it was you, Fiona. Again, I did think for a time it was the colonel. He may have been able to slip the necklace into his coat in the closet, but he would have chanced making noise and drawing attention to himself. He's too clumsy, too smart and, I daresay, too good a friend for that."

Fiona answered. "You think it was me because I was also in the restroom across the hall?"

"I considered that you, too, could have slipped the necklace into your purse in the closet—and that is exactly what you did. You are the only one present last night with enough guile to put one over on me. But, dear, I ruled the colonel out for sure when I saw your heel marks in the bedroom carpet. You really must get rid of those dreadful shoes. Someday you will trip and hurt yourself. I would suggest that your wear flats when the police arrive in a few minutes to arrest you."

Fig. 3. An example solution for a missing diamond necklace mystery taken from the 5 Minute Mysteries Website

9 CONCLUSIONS

In this work, we investigated the reasoning capabilities of LLMs in solving complex mystery puzzles as part of the Minute Mysteries QA task within the BIG-Bench evaluation suite. Initial experimentation showed that LLMs severely lack the ability to reason in a complex manner similar to humans, which motivated the application of reasoning frameworks to LLMs to illicit improved reasoning abilities. We drew inspiration from Kassner et al. and applied a belief graph framework to allow an LLM to reason about a complex story using its own beliefs about certain statements. However, results have shown that this does not produce much improvement among Llama 13b and 70b. Adding additional features to our reasoning framework such as Chain-of-Thought prompting, more descriptive prompts, more aspects to our constructed belief graphs, and additional logic constraints as part of a reduction to a Maximum Satisfiability problem did not improve performance as intended either. This work shows that LLMs are yet to overcome a large gap in reasoning before making significant advances toward Artificial General Intelligence.

ACKNOWLEDGMENTS

I would like to thank Professor Nada Amin for her support throughout the research process this semester. Her guidance and insight were truly invaluable. From introducing me to verifiable code at the beginning of the semester to pointing me in the right direction on how LLMs could be studied, Prof. Amin greatly shaped my research experience, and this study would not have been possible without her support. Additionally, I'd like to thank the following LLM researchers within the Programming Languages Lab at the Harvard School of Engineering and Applied Sciences for all the interesting ideas and discussions we've had pertaining to the subject matter: Chloe Loughridge, Rakesh Nori, and Tarun Prasad. I'd also like to thank the CS 252R Teaching Fellow Tyler Holloway for support on how to best carry out a research project of this nature. Finally, a big thank you to my peers in CS 252R for a great semester. I enjoyed meeting and learning from each and every one of you!

REFERENCES

Roberto Battiti. 2009. *Maximum satisfiability problemMaximum Satisfiability Problem.* Springer US, Boston, MA, 2035–2041. https://doi.org/10.1007/978-0-387-74759-0_364

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. arXiv:2308.09687 [cs.CL]

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *CoRR* abs/2005.14165 (2020). arXiv:2005.14165 https://arxiv.org/abs/2005.14165

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023. Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. arXiv:2310.14735 [cs.CL]

Yunmo Chen, Tongfei Chen, Seth Ebner, and Benjamin Van Durme. 2019. Reading the Manual: Event Extraction as Definition Comprehension. *CoRR* abs/1912.01586 (2019). arXiv:1912.01586 http://arxiv.org/abs/1912.01586

Nora Kassner, Oyvind Tafjord, Ashish Sabharwal, Kyle Richardson, Hinrich Schuetze, and Peter Clark. 2023. Language Models with Rationality. arXiv:2305.14250 [cs.CL]

Kanishka Misra, Cicero Nogueira dos Santos, and Siamak Shakeri. 2023. Triggering Multi-Hop Reasoning for Question Answering in Language Models using Soft Prompts and Random Walks. arXiv:2306.04009 [cs.CL]

OpenAI. 2022. Introducing ChatGPT. https://openai.com/blog/chatgpt. (Accessed on 12/12/2023).

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. *CoRR* abs/2102.07350 (2021). arXiv:2102.07350 https://arxiv.org/abs/2102.07350

Fillipe Barros Rodrigues, William Ferreira Giozza, Robson de Oliveira Albuquerque, and Luis Javier García Villalba. 2022. Natural Language Processing Applied to Forensics Information Extraction With Transformers and Graph Visualization. *IEEE Transactions on Computational Social Systems* (2022), 1–17. https://doi.org/10.1109/TCSS.2022.3159677

Priyanka Sen, Sandeep Mavadia, and Amir Saffari. 2023. Knowledge graph-augmented language models for complex question answering. In *ACL 2023 Workshop on Natural Language Reasoning and Structured Explanations*. https://www.amazon.science/publications/knowledge-graph-augmented-language-models-for-complex-question-answering

Singhal. 2012. Introducing the Knowledge Graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/. (Accessed on 12/12/2023).

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi, Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, Daphne Ippolito, Dar Gilboa, David Dohan, David Drakard, David Jurgens, Debajyoti Datta, Deep Ganguli, Denis Emelin, Denis Kleyko, Deniz Yuret, Derek Chen, Derek Tam, Dieuwke Hupkes, Diganta Misra, Dilyar Buzan, Dimitri Coelho Mollo, Diyi Yang, Dong-Ho Lee, Dylan Schrader, Ekaterina Shutova, Ekin Dogus Cubuk, Elad Segal, Eleanor Hagerman, Elizabeth Barnes, Elizabeth Donoway, Ellie Pavlick, Emanuele Rodola, Emma Lam, Eric Chu, Eric Tang, Erkut Erdem, Ernie Chang, Ethan A. Chi, Ethan Dyer, Ethan Jerzak, Ethan Kim, Eunice Engefu Manyasi, Evgenii Zheltonozhskii, Fanyue Xia, Fatemeh Siar, Fernando Martínez-Plumed, Francesca Happé, Francois Chollet, Frieda Rong, Gaurav Mishra, Genta Indra Winata, Gerard de Melo, Germán Kruszewski, Giambattista Parascandolo, Giorgio Mariani, Gloria Wang, Gonzalo Jaimovitch-López, Gregor Betz, Guy Gur-Ari, Hana Galijasevic, Hannah Kim, Hannah Rashkin, Hannaneh Hajishirzi, Harsh Mehta, Hayden Bogar, Henry Shevlin, Hinrich Schütze, Hiromu Yakura, Hongming Zhang, Hugh Mee Wong, Ian Ng, Isaac Noble, Jaap Jumelet, Jack Geissinger, Jackson Kernion, Jacob Hilton, Jaehoon Lee, Jaime Fernández Fisac, James B. Simon, James Koppel, James Zheng, James Zou, Jan Kocoń, Jana Thompson, Janelle Wingfield, Jared Kaplan, Jarema Radom, Jascha

Sohl-Dickstein, Jason Phang, Jason Wei, Jason Yosinski, Jekaterina Novikova, Jelle Bosscher, Jennifer Marsh, Jeremy Kim, Jeroen Taal, Jesse Engel, Jesujoba Alabi, Jiacheng Xu, Jiaming Song, Jillian Tang, Joan Waweru, John Burden, John Miller, John U. Balis, Jonathan Batchelder, Jonathan Berant, Jörg Frohberg, Jos Rozen, Jose Hernandez-Orallo, Joseph Boudeman, Joseph Guerr, Joseph Jones, Joshua B. Tenenbaum, Joshua S. Rule, Joyce Chua, Kamil Kanclerz, Karen Livescu, Karl Krauth, Karthik Gopalakrishnan, Katerina Ignatyeva, Katja Markert, Kaustubh D. Dhole, Kevin Gimpel, Kevin Omondi, Kory Mathewson, Kristen Chiafullo, Ksenia Shkaruta, Kumar Shridhar, Kyle McDonell, Kyle Richardson, Laria Reynolds, Leo Gao, Li Zhang, Liam Dugan, Lianhui Qin, Lidia Contreras-Ochando, Louis-Philippe Morency, Luca Moschella, Lucas Lam, Lucy Noble, Ludwig Schmidt, Luheng He, Luis Oliveros Colón, Luke Metz, Lütfi Kerem Şenel, Maarten Bosma, Maarten Sap, Maartje ter Hoeve, Maheen Farooqi, Manaal Faruqui, Mantas Mazeika, Marco Baturan, Marco Marelli, Marco Maru, Maria Jose Ramírez Quintana, Marie Tolkiehn, Mario Giulianelli, Martha Lewis, Martin Potthast, Matthew L. Leavitt, Matthias Hagen, Mátyás Schubert, Medina Orduna Baitemirova, Melody Arnaud, Melvin McElrath, Michael A. Yee, Michael Cohen, Michael Gu, Michael Ivanitskiy, Michael Starritt, Michael Strube, Michał Swędrowski, Michele Bevilacqua, Michihiro Yasunaga, Mihir Kale, Mike Cain, Mimee Xu, Mirac Suzgun, Mitch Walker, Mo Tiwari, Mohit Bansal, Moin Aminnaseri, Mor Geva, Mozhdeh Gheini, Mukund Varma T, Nanyun Peng, Nathan A. Chi, Nayeon Lee, Neta Gur-Ari Krakover, Nicholas Cameron, Nicholas Roberts, Nick Doiron, Nicole Martinez, Nikita Nangia, Niklas Deckers, Niklas Muennighoff, Nitish Shirish Keskar, Niveditha S. Iyer, Noah Constant, Noah Fiedel, Nuan Wen, Oliver Zhang, Omar Agha, Omar Elbaghdadi, Omer Levy, Owain Evans, Pablo Antonio Moreno Casares, Parth Doshi, Pascale Fung, Paul Pu Liang, Paul Vicol, Pegah Alipoormolabashi, Peiyuan Liao, Percy Liang, Peter Chang, Peter Eckersley, Phu Mon Htut, Pinyu Hwang, Piotr Miłkowski, Piyush Patil, Pouya Pezeshkpour, Priti Oli, Qiaozhu Mei, Qing Lyu, Qinlang Chen, Rabin Banjade, Rachel Etta Rudolph, Raefer Gabriel, Rahel Habacker, Ramon Risco, Raphaël Millière, Rhythm Garg, Richard Barnes, Rif A. Saurous, Riku Arakawa, Robbe Raymaekers, Robert Frank, Rohan Sikand, Roman Novak, Roman Sitelew, Ronan LeBras, Rosanne Liu, Rowan Jacobs, Rui Zhang, Ruslan Salakhutdinov, Ryan Chi, Ryan Lee, Ryan Stovall, Ryan Teehan, Rylan Yang, Sahib Singh, Saif M. Mohammad, Sajant Anand, Sam Dillavou, Sam Shleifer, Sam Wiseman, Samuel Gruetter, Samuel R. Bowman, Samuel S. Schoenholz, Sanghyun Han, Sanjeev Kwatra, Sarah A. Rous, Sarik Ghazarian, Sayan Ghosh, Sean Casey, Sebastian Bischoff, Sebastian Gehrmann, Sebastian Schuster, Sepideh Sadeghi, Shadi Hamdan, Sharon Zhou, Shashank Srivastava, Sherry Shi, Shikhar Singh, Shima Asaadi, Shixiang Shane Gu, Shubh Pachchigar, Shubham Toshniwal, Shyam Upadhyay, Shyamolima, Debnath, Siamak Shakeri, Simon Thormeyer, Simone Melzi, Siva Reddy, Sneha Priscilla Makini, Soo-Hwan Lee, Spencer Torene, Sriharsha Hatwar, Stanislas Dehaene, Stefan Divic, Stefano Ermon, Stella Biderman, Stephanie Lin, Stephen Prasad, Steven T. Piantadosi, Stuart M. Shieber, Summer Misherghi, Svetlana Kiritchenko, Swaroop Mishra, Tal Linzen, Tal Schuster, Tao Li, Tao Yu, Tariq Ali, Tatsu Hashimoto, Te-Lin Wu, Théo Desbordes, Theodore Rothschild, Thomas Phan, Tianle Wang, Tiberius Nkinyili, Timo Schick, Timofei Kornev, Titus Tunduny, Tobias Gerstenberg, Trenton Chang, Trishala Neeraj, Tushar Khot, Tyler Shultz, Uri Shaham, Vedant Misra, Vera Demberg, Victoria Nyamai, Vikas Raunak, Vinay Ramasesh, Vinay Uday Prabhu, Vishakh Padmakumar, Vivek Srikumar, William Fedus, William Saunders, William Zhang, Wout Vossen, Xiang Ren, Xiaoyu Tong, Xinran Zhao, Xinyi Wu, Xudong Shen, Yadollah Yaghoobzadeh, Yair Lakretz, Yangqiu Song, Yasaman Bahri, Yejin Choi, Yichi Yang, Yiding Hao, Yifu Chen, Yonatan Belinkov, Yu Hou, Yufang Hou, Yuntao Bai, Zachary Seid, Zhuoye Zhao, Zijian Wang, Zijie J. Wang, Zirui Wang, and Ziyi Wu. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615 [cs.CL]

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *CoRR* abs/2201.11903 (2022). arXiv:2201.11903 https://arxiv.org/abs/2201.11903

Brandon T Willard and Rémi Louf. 2023. Efficient Guided Generation for LLMs. *arXiv preprint arXiv:2307.09702* (2023).

Kaiyu Yang and Jia Deng. 2019. Learning to Prove Theorems via Interacting with Proof Assistants. *CoRR* abs/1905.09381 (2019). arXiv:1905.09381 http://arxiv.org/abs/1905.09381

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL]

Ruiqi Zhong, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. 2021. Are Larger Pretrained Language Models Uniformly Better? Comparing Performance at the Instance Level. *CoRR* abs/2105.06020 (2021). arXiv:2105.06020 https://arxiv.org/abs/2105.06020

Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. 2023. Large Language Models can Learn Rules. arXiv:2310.07064 [cs.AI]

10 APPENDIX

10.1 Soft Constraint: What Node to Guilt Implication Confidence Function

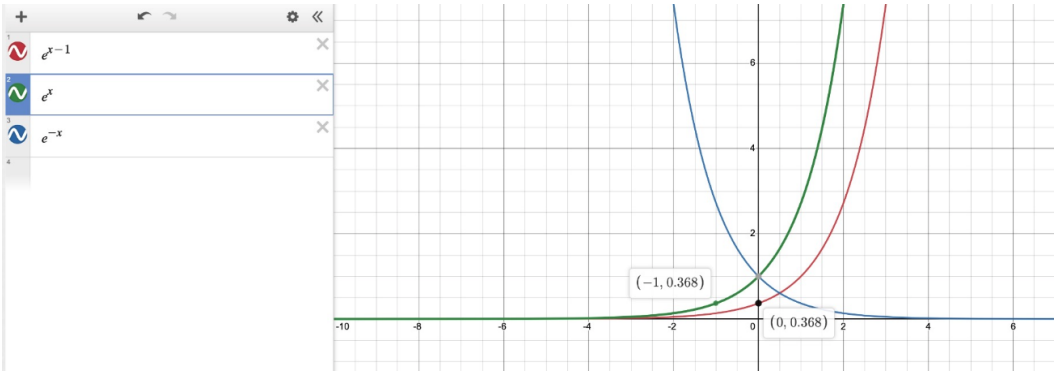


Fig. 4. Visualization of e^x transformations for choosing the confidence function in placing soft implication constraints from what nodes to guilt nodes

10.2 Note on Chain of Thought Prompting Experiment

I initially thought I had ran this experiment already, but after looking back at the codebase, I realized there was a minor error in my execution that prevented from actually passing in the new CoT prompt when querying the LLM. Thus, the results that I thought were obtained for the CoT feature, were for the original belief graph results (meaning I accidentally re-ran the experiment). I recently attempted to correct this and use CoT, but I realized that the context window limit was being hit. In an attempt to work around this, I tried using ChatGPT 4 to summarize the mystery story and solution, but the summaries being produced for the CoT example left out crucial details. As such, the example solution, which consists of reasoning on these specific details, did not seem to be as clear (or relevant) with regard to the summarized mystery. As an additional next step, I intend to explore this experiment further by querying an LLM to generate a more detailed summary or manually creating a summary for the 1-shot example to run the CoT experiment.