Boosting Theory-of-Mind Performance in small Large Language Models

1 Introduction

Theory of Mind (ToM) is the cognitive ability to attribute mental states to themselves and others. This involves beliefs, desires, intentions, emotions, and thoughts. Therefore, ToM allows us to understand, explain, and predict the behavior of others. Westby&Robinson[30] suggest that ToM in humans is a learned skill that is developed in early childhood and requires the understanding of attention, intentionality, and imitation as precursors. Humans have the cognitive capability for ToM reasoning, but up to this point, it remains an open and controversial question if non-human animals are capable of ToM[8].

Large Language Models (LLMs) are deep learning models trained on high amounts of text to understand and generate written language, often exceeding a size of 100 billion parameters. LLMs show high capabilities in understanding and generating language, semantic comprehension, grammar, translation, summarizing, and question-answering. However, research shows that LLMs struggle with problems that involve logical reasoning[11][20].

ToM tasks typically consist of a scenario description with one or more agents and a question about the scenario. The description of the scenario does not include explicit information about the state of mind of any involved agent. However, the question is about the mental state of one involved agent. That means that the solution of a ToM task is only implicitly included in the task and has to be logically derived from the given information. Additionally, the mental state of the agent is only a likely consequence of the described scenario but does not strictly follow from the given scenario.
Therefore, solving the ToM tasks requires the use of inferential inductive reasoning. Inferential reasoning is a process that uses intermediate reasoning steps to derive logical consequences given a set of premises or observations. Inference can be divided into deduction and induction. In deductive reasoning, the consequence necessarily follows from the premises, given the premises are true. In inductive reasoning, the consequence is likely to be true, given the observed evidence, but does not necessarily follow from the premises.
Given the problems of LLMs with reasoning tasks, it is no surprise that LLMs perform very poor in ToM reasoning tasks[3][23][28].

Moghaddam&Honey[14] could show that certain modern state-of-the-art Large Language Models with over 100 billion parameters (GPT-4[18] and variants of GPT-3.5[17]) were capable of correctly answering questions that involve ToM reasoning. An accuracy of 80% was exceeded for all models when using appropriate in-context prompting methods. As prompting methods Step-by-Step thinking [13] and few-shot learning in the form of Chain-of-Thought reasoning [29] were used. These methods were compared to zero-shot prompting as a baseline. The combined use of both methods yielded the highest accuracy in all models. All models could significantly increase their accuracy compared to the zero-shot prompting. GPT-4[18] even reached an accuracy of 100% when combining Step-by-Step thinking and Chain-of-Thought reasoning. However, Moghaddam&Honey[14] also discovered that another model of similar size, Davinci-2, a version of GPT-3.5[17], was not able to exploit the used in-context prompting methods and was not able to increase its accuracy compared to the zero-shot prompting. The authors assume that the reason for that was the fact that Davinci-2 was the only model that was not trained with Reinforcement Learning from Human Feedback[19][2]. They assume that RLHF enables the other observed LLMs to make use of in-context prompting.

The impressive performance of Large Language Models in different tasks involving language was mainly attributed to their high number of parameters[12]. However, Hoffmann et al.[9] could show that larger models were outperformed by smaller models if the smaller models were trained on higher amounts of data. In 2022 Open AI introduced LLaMa[26], a series of small Large Language Models with a parameter count ranging from 7B to 65B. The LLaMa model with 13 billion parameters was able to outperform GPT-3 with 175 billion parameters. The promising results of small LLMs on common performance benchmarks suggest that also small LLMs are capable of answering questions that involve ToM reasoning, given the correct form of prompting.

The goal of this research project is to investigate the Theory of Mind capabilities of Large Language Models with a small number of parameters ($\sim$7B) and the effect of in-context prompting on these capabilities. To access the capabilities of different small Large Language Models the experiment from Moghaddam&Honey[14] was reproduced for the open-source models GPT4All-J v1.3, LLaMA2 and LLaMa2 Chat. Additionally, GPT4All-J v1.3 was finetuned using two different human preference

optimization methods. The experiment was also reproduced for the resulting models GPT4All-J PPO and GPT4All-J DPO.

The evaluation scripts, the evaluation results, and the scripts for finetuning can be found in the corresponding GitHub repository[1].

2 Methods

2.1 Theory of Mind Experiment

2.1.1 Models

The models used in this experiment were GPT4All-J v1.3-groovy 7B[15], LLaMa2 7B[27] and LLaMa2 Chat 7B[27].

GPT4All-J is a family of LLMs made by Nomic AI and derived from GPT-J. GPT4All-J was trained on the GPT4All dataset[16] which is a diverse sample of 800k questions/prompts. Revision v1.3-groovy was trained on a slightly modified version of the GPT4All-J dataset in which some datapoints were filtered out and datapoints from Dolly15k[4] and ShareGPT[22] were added.

LLaMa2 is a family of LLMs made by Meta AI and is an improvement of the previous LLaMa[26] family. The improvements compared to LLaMa were made by increasing the dataset size by 40% and by increasing the context length of the model. LLaMa2 Chat was additionally trained on over one million human annotated datapoints by using RLHF[19][2].

Additionally, GPT4All v1.3 was finetuned for human preference by using using Proximal Policy Optimization[24] and Direct Preference Optimization[21]. The resulting model is referred to as "GPT4All-J PPO" and "GPT4All-J DPO". For more detailed information see Section 3.

A comparison of all used models and GPT-4 on common benchmarks can be seen in Table 1. The benchmarks for the small LLMs were conducted by using the Language Model Evaluation Harness[7]. The benchmark results of GPT-4 were obtained from the official technical report[18].

Model	Size	HellaSwag	WinoGrande	BooLQ	PIQA	LogiQA	ARC-Challenge
GPT-4	∼175B	95.3% (10-shot)	87.5% (5-shot)	-	-	-	-
GPT4All-J v1.3		63.77%	63.14%	73.49%	74.37%	27.04%	35.15%
GPT4All-J PPO		63.7%	64.96%	64.77%	74.59%	24.27%	35.84%
GPT4All-J DPO	∼7B	62.37%	63.3%	65.2%	72.25%	26.73%	32.76%
LLaMa2		75.99%	69.06%	77.77%	79.05%	30.41%	46.42%
LLaMa2 Chat		75.41%	66.38%	80.70%	76.66%	30.57%	44.37%

Table 1: Accuracy of the observed LLMs and GPT-4 on different Benchmarks.

2.1.2 Experimental Design

The Methods described in this section are identical to the methods described in the original paper by Moghaddam&Honey[14].

16 Theory of Mind tasks were used to evaluate the ToM capabilities of the LLMs. The tasks were adapted from a human study that examined the brain areas involved in solving ToM tasks[6]. The average accuracy in the human study was 87%.
The tasks consist of a scenario description that involves one or multiple agents and a question about the scenario that requires inferring the state of mind of one agent. The scenario and question were set in a prompt framework that included a task instruction and prompts that marked the beginning of the scenario description, the question, and the answer. The answer was generated by the observed models using causal language modeling. Each scenario was presented 20 times.
Each answer generated by the LLMs was analyzed manually. Answers were labeled as correct if they had the correct conclusion or if the model included the correct reasoning. Answers were labeled as incorrect if they had an incorrect conclusion or if the responses were inconclusive.

Each ToM scenario was used with four different types of in-context prompting:

[1] https://github.com/Z3R6X/ToM_in_small_LLMs

1. Zero-shot/Base prompting: No additional prompting.
2. Zero-shot with step-by-step (SbS) thinking: Prompt "Let's think step by step: " was appended to the zero-shot prompt.
3. Two-shot Chain-of-thought (CoT) reasoning: The zero-shot prompt was preceded by two example ToM tasks and correct answers.
4. Combined two-shot chain-of-thought reasoning and step-by-step thinking.

An example of all types of in-context prompting can be seen in Fig. 1. For each model this resulted in a total of 1280 questions (16 scenarios $\times$ 20 repetitions $\times$ 4 types of prompting).

```
Read the scenario and answer the following question:

Scenario: <scenario> Question: <question> Answer:
```

(a) Basic prompting.

```
Read the scenario and answer the following question:

Scenario: <scenario> Question: <question> Answer: Let's think step by step:
```

(b) Step-by-Step prompting.

```
Read the scenario and answer the following question:

Scenario: <cot-scenario> Question: <cot-question> Answer: <cot-answer>

Scenario: <cot-scenario> Question: <cot-question> Answer: <cot-answer>

Scenario: <scenario> Question: <question> Answer:
```

(c) Chain-of-Thought prompting.

```
Read the scenario and answer the following question:

Scenario: <cot-scenario> Question: <cot-question> Answer: Let's think step by step: <cot-answer>

Scenario: <cot-scenario> Question: <cot-question> Answer: Let's think step by step: <cot-answer>

Scenario: <scenario> Question: <question> Answer: Let's think step by step:
```

(d) Combined Chain-of-Thought and Step-by-Step prompting.

Figure 1: The different types of in-context prompting used in the ToM experiment.

A paired t-test was used to quantify the performance differences between different types of in-context prompting in one model. The number of correct responses for each scenario was used as observations ($N = 16$). The t-test is done between the base prompting and all other types of prompting (Base vs. {SbS,CoT,SbS+CoT}). However, it should be noted that the assumption of normality might not be fulfilled, as the number of observations is very low. It can also not be guaranteed that there exists no dependence between the observations of different scenarios. Additionally, the homogeneity of variances of two prompting types is not always fulfilled to the same extent.

3 Model Finetuning

The results from Moghaddam&Honey[14] and the results of this experiment suggest that finetuning for human preference or RLHF[19][2] in particular plays a crucial role in the performance of LLMs on ToM tasks. To verify this hypothesis GPT4All-J v1.3 was finetuned on a human preference dataset using Proximal Policy Optimization[24] and Direct Preference Optimization[21]. Both finetuning approaches were preceded by supervised finetuning on the same dataset.
The goal of both finetuning approaches was to increase the performance of the model on ToM tasks and its ability to make use of different types of prompting while maintaining the performance of the model on other tasks. The performance of the base model and the finetuned versions on a set of benchmarks can be seen in Table 2.

Model	HellaSwag	WinoGrande	BooLQ	PIQA	LogiQA	ARC-Challenge
GPT4All-J v1.3	63.77%	63.14%	73.49%	74.37%	27.04%	35.15%
GPT4All-J SFT PPO	63.96%	65.27%	65.17%	74.65%	24.12%	36.86%
GPT4All-J PPO	63.7%	64.96%	64.77%	74.59%	24.27%	35.84%
GPT4All-J SFT DPO	63.71%	64.01%	69.45%	74.05%	27.24%	35.92%
GPT4All-J DPO	62.37%	63.3%	65.2%	72.25%	26.73%	32.76%

Table 2: Accuracies of GPT4All-J v1.3 and GPT4All-J finetunes on different Benchmarks.

3.1 General Finetuning Methods

Instead of finetuning all parameters of the model, Low-rank Adaption (LoRA)[10] was used. LoRA freezes the parameters of the model and uses trainable rank decomposition matrices for some linear layers of the model. As a result, the parameters of the linear layers are updated with a low-rank approximation of the gradient matrix. Therefore, LoRA heavily reduces the number of parameters during training and reduces the memory requirements for training. Additionally, LoRA decreases the training time.

Dettmers et al.[5] showed that the use of block-wise dynamic quantization enabled the use of 8-bit optimizers that maintain the performance of 32-bit optimizers, while heavily decreasing the memory requirements. Therefore, to further decrease the memory requirements 8-bit and 4-bit quantization was used during PPO and DPO finetuning.

Different types of prompting during training and inference can decrease the performance of the model during inference. The type of prompting that was used during finetuning should be the same that is used in the ToM reasoning task. The kind of prompting that is used in the original paper of Moghaddam&Honey[14] is adjusted to the ToM scenarios, but not to in-context question answering in general. Therefore, the prompting for finetuning is adapted to more general in-context question answering. In the case of GPT4All-J PPO and GPT4All-J DPO, the same type of prompting is then also used during inference for the ToM reasoning tasks. The type of prompting used for the finetuned models can be seen in Fig. 2.

```
Answer the following question in context:

Context: <scenario> Question: <question> Answer:
```

Figure 2: The type of prompting used during finetuning and in the ToM experiment for GPT4All-J PPO and GPT4All-J DPO.

3.2 Dataset

DPO finetuning and reward modeling in PPO both require the use of a human preference dataset. Human preference datasets contain a preferred continuation and an unwanted continuation to a given prompt. When finetuning a model for the task of in-context question answering the prompt should include a question and the different continuations should be responses to this question.
There is no appropriate preference dataset specifically made for in-context question answering

available. Therefore, the models were trained on a dataset that combines preference and simple question answering: "Dahoas/instruct-synthetic-prompt-responses"[1]. It contains 30k datapoints and is intended for human preference optimization.

3.3 Proximal Policy Optimization

Proximal Policy Optimization (PPO)[24] is a policy gradient method that can be used for Reinforcement Learning from Human Feedback (RLHF)[19][2].
The goal of reinforcement learning is to optimize the policy of an agent concerning an objective. The objective or task is determined by a reward function. The reward function assigns a score to the actions of an agent. In NLP the reward is determined by a reward model, which rates the generated answer of the agent. In RLHF the goal is to use the reward model to steer the agent into generating human-preferable responses. The reward model is trained to assign higher scores to the preferred answer in a supervised fashion. This requires a comparative dataset that contains a question combined with a preferred and a rejected answer. The trained reward model is used to rate the generated answers of the agent given the questions in the same dataset. The policy of the agent is then updated given the obtained rewards by applying gradient descent to an estimator of the policy gradient.
PPO introduces a constraint in the form of a KL penalty term for every policy update step, such that the updated policy does not deviate too much from the old policy. Therefore, PPO stabilizes the reinforcement learning process. The high effectiveness, stability, and simplicity make PPO a popular choice for a reinforcement learning algorithm in many tasks and applications, also outside the field of NLP.

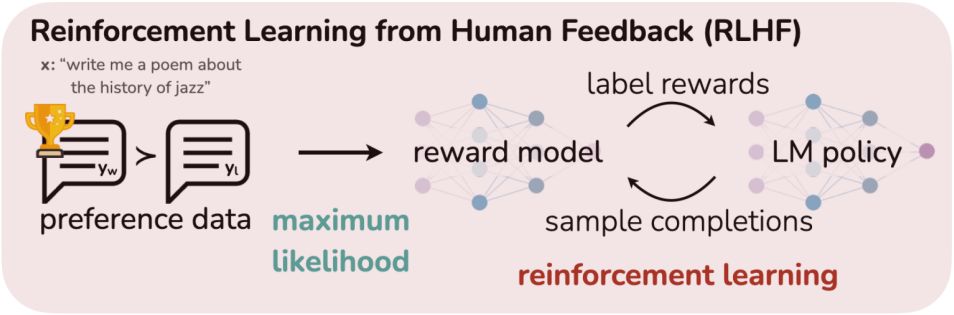

Figure 3: Reinforcement Learning from Human Feedback. Human preference optimization with reinforcement learning.

3.3.1 Finetuning

To prevent a strong distribution shift between the true preferences and the preferences during PPO finetuning, GPT4All-J v1.3 was first finetuned on "Dahoas/instruct-synthetic-prompt-responses" with supervision. For supervised finetuning the questions combined with the preferred answers were used as targets. The resulting model is referred to as "GPT4All-J PPO SFT".
Afterward, a reward model based on GPT4All-J PPO SFT was trained on "Dahoas/instruct-synthetic-prompt-responses", referred to as "GPT4All-J PPO RM". For reward modeling the question and both the preferred and rejected answer were used.
GPT4All-J PPO RM was then used as a reward model to finetune GPT4All PPO SFT with PPO. For PPO finetuning the question was used and the answer was generated by GPT4All-J PPO SFT. The final model is referred to as "GPT4All-J PPO".
Hyperparameter choices and other details can be found in the GitHub repository[2].

3.4 Direct Preference Optimization

A drawback of PPO and reinforcement learning in general is its comparatively high complexity and instability in training, that comes with using a reward model, the need for sampling, and the use of reinforcement learning.
Direct Preference Optimization (DPO) is an algorithm that avoids sampling, the use of a reward

[2] https://github.com/Z3R6X/ToM_in_small_LLMs/tree/master/PPO

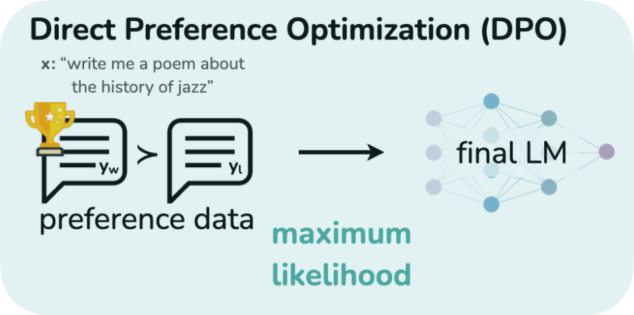

Figure 4: Direct Preference Optimization. Human preference optimization without reinforcement learning.[21]

model, and reinforcement learning. Leveraging an analytical mapping from reward functions to optimal policies, a loss function over reward functions is effectively transformed into a loss function over policies. Therefore, DPO extracts the optimal policy in closed form by fitting the implicit reward model on the preferences using a simple classification objective. This results in a more simple, more stable, and more efficient preference finetuning algorithm, that still optimizes the same objective as RLHF[21].

3.4.1 Finetuning

To prevent a strong distribution shift between the true preferences and the preferences during DPO finetuning, GPT4All-J v1.3 was first finetuned on "Dahoas/instruct-synthetic-prompt-responses" with supervision. For supervised finetuning the questions combined with the preferred answers were used as targets. The resulting model is referred to as "GPT4All-J DPO SFT".
Afterward, GPT4All-J DPO SFT was finetuned on "Dahoas/instruct-synthetic-prompt-responses" using DPO. For DPO the question and both the preferred and rejected answer were used.
Hyperparameter choices and other details can be found in the GitHub repository[3]. GPT4All-J DPO can be found on Huggingface[4]

4 Results

The number of correct answers on each question for each model and type of prompting can be found in Section 7.1. The carried out statistical tests can be found in Section 7.2.

4.1 LLMs trained without Human Preference

4.1.1 GPT4All-J v1.3

GPT4All-J v1.3 reaches the best performance for base prompting without additional Step-by-Step or Chain-of-Thought prompting. However, for the base prompting the model shows a very strong yes-bias. SbS prompting often causes rambling, which causes a decrease in performance when using SbS prompting ($p = 0.0442^*$). CoT prompting sometimes causes example repetition. The combined use of SbS and CoT prompting, prevents the model from rambling, while still causing example repetition. Both CoT and SbS+CoT prompting have a similar performance as the Base prompting. The yes-bias that was observed in the base prompting is still noticeable when using other types of prompting.

4.1.2 LLaMa2

LLaMa2 reaches the worst performance for base prompting, which is mainly caused by rambling. SbS prompting has a similar performance as base prompting since it also suffers from rambling and a yes-bias. CoT prompting strongly increases the performance of the model ($p = 0.0337^*$), as it heavily

[3] https://github.com/Z3R6X/ToM_in_small_LLMs/tree/master/DPO
[4] https://huggingface.co/Z3R6X/gpt4all_dpo_instruct

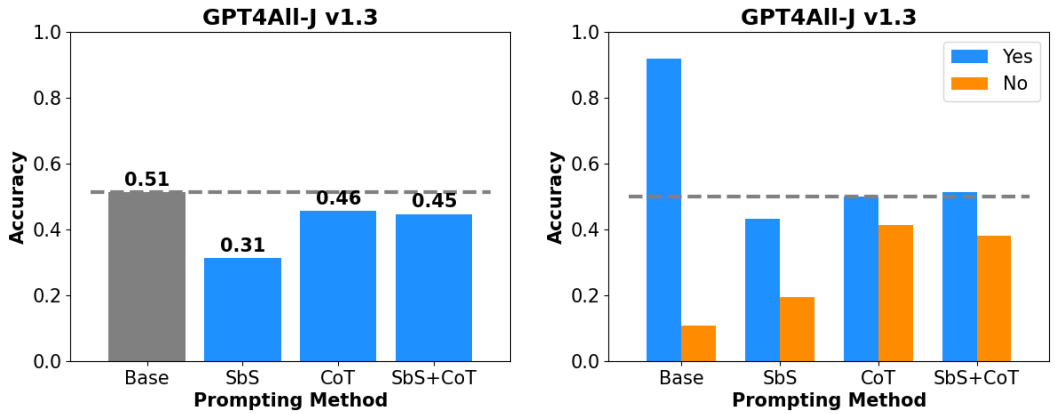

Figure 5: Effects of In-context Learning Prompts on ToM performance in GPT4All-J PPO.

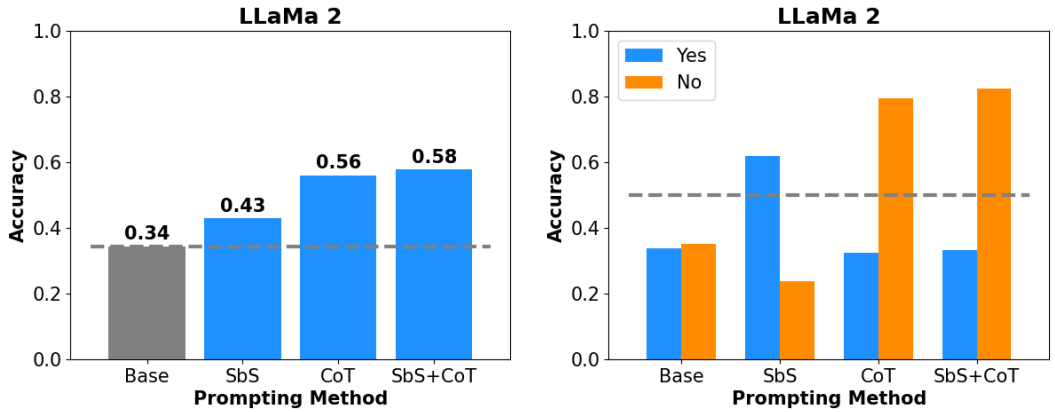

Figure 6: Effects of In-context Learning Prompts on ToM performance in LLaMa2.

reduces the rambling. However, CoT prompting causes a no-bias. Additionally, Cot prompting alone causes example repetition. Combining CoT prompting with SbS prompting decreases the amount of example repetitions. Therefore, LLaMa2 reaches the strongest increase in performance for combined CoT and SbS prompting ($p = 0.0195^*$).

4.2 LLMs trained with Human Preference

4.2.1 LLaMa2 Chat

LLaMa2 Chat reaches the worst performance for the base prompting, which is predominantly caused by multiple answers. SbS prompting heavily prevents multiple answering, which heavily increases the performance ($p = 0.0096^*$). CoT prompting also heavily prevents multiple answering, which increases the performance ($p = 0.0095^*$). However, CoT prompting causes a no-bias. Combining CoT with SbS prompting decreases the No-bias and increases the performance on some yes-questions. Therefore, LLaMa2 Chat reaches the best performance for the combined CoT and SbS prompting ($p = 0.0001^*$).

4.2.2 GPT4All-J PPO

GPT4All-J PPO reaches the best performance for base prompting without additional Step-by-Step or Chain-of-Thought prompting. SbS prompting causes a yes-bias, which causes a decrease in performance ($p = 0.0240^*$). Therefore SbS prompting reaches the worst performance. Both CoT and SbS+CoT prompting have a similar performance as the Base prompting. GPT4All-J PPO does

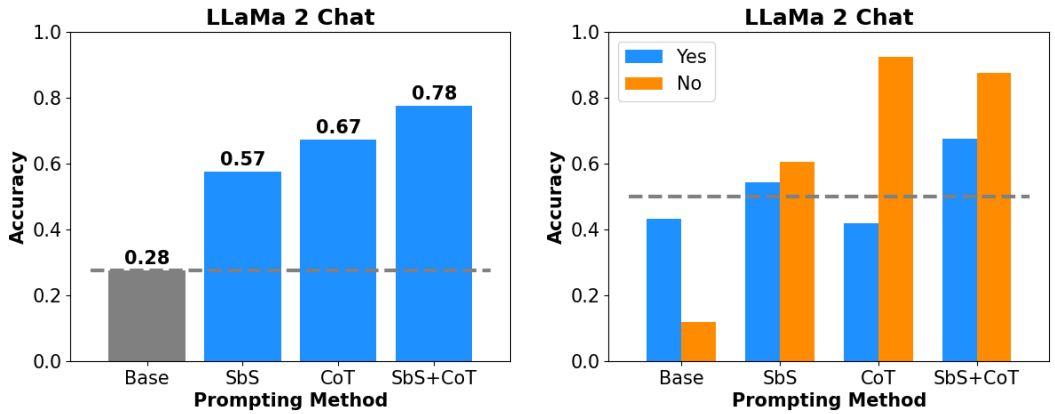

Figure 7: Effects of In-context Learning Prompts on ToM performance in LLaMa2 Chat.

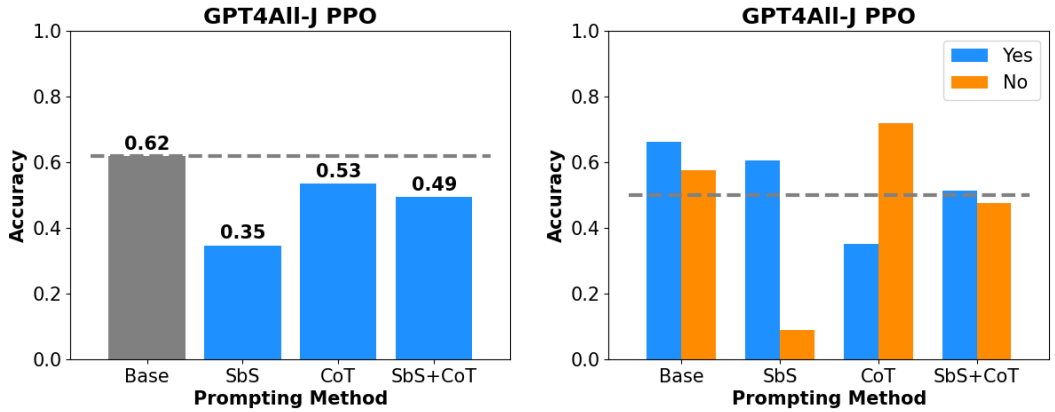

Figure 8: Effects of In-context Learning Prompts on ToM performance in GPT4All-J PPO.

not suffer from the typical types of errors for any type of prompting as the other models. The errors are almost exclusively caused by wrong reasoning steps or conclusions.

4.2.3 GPT4All-J DPO

GPT4All-J DPO reaches the worst performance for the base prompting, which is caused by a yes-bias and question evasion. SbS prompting slightly increases the performance, as it prevents question evasion, but causes rambling. Additionally the yes-bias remains. CoT prompting increases the performance ($p = 0.0290^*$), as it reduces the question evasions. However, the yes-bias still remains. The combined use of CoT and SbS prompting causes the highest increase in performance ($p = 0.0109^*$), as it prevents question evasion and prevents the yes-bias. Therefore, GPT4All-J DPO reaches the best performance for the combined CoT and SbS prompting.

5 Discussion

5.1 General Discussion

All models exhibit different behaviors when answering the questions for different types of in-context prompting. Most mistakes are not caused by wrong reasoning steps or conclusions. Instead, most mistakes are caused by the model not answering the question at all. Some common reoccurring behaviors that cause errors can be identified:

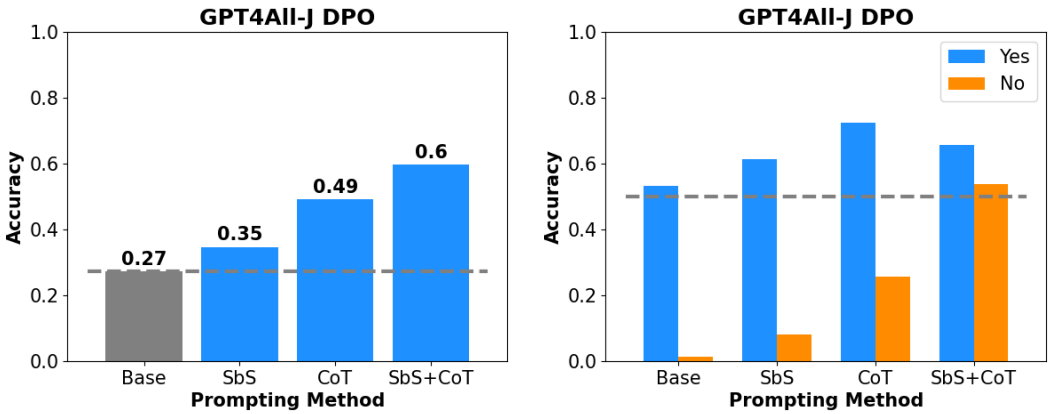

Figure 9: Effects of In-context Learning Prompts on ToM performance in GPT4All-J DPO.

- **multiple answers**: The model lists possible answers to the question, without stating which of the answers is correct.
- **task deviation/rambling**: The model generates a text that is connected to the context of the scenario, but does not answer the question.
- **example repetition**: Instead of answering the question, the model repeats the answer to one question included in the Chain-of-Thought examples.
- **question evasion**: The model evades the question by stating that it is not possible to determine the answer (without further information).

Most types of errors can be attributed to a type of in-context prompting or a type of model:

- **multiple answers** almost exclusively occur when using base prompting.
- **task deviation/rambling** mostly occurs when using Base or SbS prompting, but not when using CoT prompting.
- **example repetition** only occurs when using CoT prompting (as Base and SbS prompting alone do not contain possible examples).
- **question evasion** mostly occurs in models that were trained using human preference.

The types of error associated with the type of in-context prompting occur in every model, but to different extends.

Another behavior that occurs in some models is a **yes(/no)-bias**, which means that the model has a bias to answer questions generally with "Yes" or "No". All LLMs exhibit a yes or no-bias for at least one type of prompting. The yes-bias occurs mostly in models of the GPT4All-J family and never for combined SbS and CoT prompting. The no-bias occurs mostly in the LLMs of the LLaMa 2 family and only for CoT prompting or combined SbS+CoT prompting.
The yes-bias in LLMs trained with human preference finetuning might result from the fact that these LLMs tend to be more helpful and agreeable with the user and are therefore more likely to answer questions with "Yes". However, this does not apply to models trained without human preference finetuning.
The no-bias in the LLMs of the LLaMa 2 family and GPT4All-J PPO, might be caused by an over-sensitivity for the CoT-examples, as it only occurs when CoT prompting is used.

The observed models can be separated into two categories:
First, models that profit from the use of in-context learning via prompts (LLaMa2, LLaMa2-Chat and GPT4All-J DPO). Here, the use of in-context learning was helpful as it induced behavior in the models that increased the model's capability to answer the ToM questions correctly. Each type of prompting induces also specific types of errors that cause incorrect answers. However, the advantages outweigh the disadvantages.
Second, models that do not profit from the use of in-context learning via prompts (GPT4All-J v1.3

and GPT4All-J PPO). Here the use of in-context learning induced a behavior that increased the model's susceptibility for errors. Therefore, the models was less capable of answering the ToM questions correctly. The disadvantages predominate.

In all models that were not able to increase their performance by using in-context learning, the base prompting yielded the best performance.

Contrary to the results of Moghaddam&Honey[14] and the results for LLaMa2-Chat, GPT4All-J PPO was not able to increase its performance for in-context learning, although it was trained with RLHF. However, this does not necessarily mean that PPO finetuning is generally not able to increase the in-context learning capabilities of GPT4All-J v1.3.
The lack of performance increase for in-context learning in GPT4All-J PPO could also be caused by model collapse[25] during training. In the context of LLMs, model collapse describes a phenomenon in which a model fails to generate diverse and meaningful outputs. Instead, the model generates repetitive or similar outputs regardless of the input. An indicator of this could be the fashion in which GPT4All-J PPO answers the ToM questions. Each answer generated by GPT4All-J PPO begins with the binary response "Yes" or "No", followed by a very brief explanation, no matter which type of prompting is used. Model collapse is mostly caused by an overfit on the training data or in the case of reinforcement learning by reward hacking. Additionally, model collapse is associated with the use of synthetically generated datasets. Given the fact that the used dataset was synthetically generated and the unstable nature of reinforcement learning in general and PPO in particular, it is highly likely that GPT4All-J PPO suffers from model collapse.
It can be assumed that some adaptions in the training parameters, the training data, or the used reward model could result in a PPO-trained version of GPT4All-J that is capable of making use of in-context learning.
It can be concluded that PPO finetuning on GPT4All-J was done improperly and unsuccessful. Therefore, the results of GPT4All-J PPO should not be seen as representative of proper and successful PPO finetuning.

In all models that were able to increase their performance by using in-context learning, the combined use of SbS and CoT prompting increased the performance the most. An advantage of using both CoT and SbS prompting is the fact that they do not interfere with each other. This means the induced behavior for both methods does not contradict each other. This should be considered when combining different types of in-context prompting.

The results for GPT4All-J DPO show, that not only RLHF[19][2] but also other types of human preference finetuning are capable of enabling models to exploit in-context learning to answer questions about ToM scenarios. Additionally, the results for LLaMa 2 show, that even models trained without any type of human preference finetuning explicitly are capable of exploiting in-context learning to answer questions about ToM-scenarios. Therefore, human preference finetuning does not seem to be a requirement for profiting from in-context learning.

Similar to much bigger LLMs, like Davinci-3 or GPT-3.5-Turbo, some smaller LLMs, like LLaMa2 Chat or GPT4All-J DPO, are only capable of answering questions about ToM scenarios when using appropriate prompting. Although bigger LLMs generally perform better than smaller LLMs, model size alone seems to be a subordinate factor in performance. The most important factor is the capability of the LLM to make use of in-context prompts. For example, LLaMa2-Chat ($\sim$7B parameters) with combined SbS and CoT prompting achieves a better performance than Davinci-3 and GPT-3.5-Turbo ($\sim$175B parameters) without in-context prompting[14].

Moghaddam&Honey[14] argue that single-word completions and multiple choice answers are not capable of capturing the complex behavior that might be required to solve ToM reasoning tasks. According to Moghaddam&Honey[14] the long-form answers generated during their and this experiment have two benefits:

1. **A more fair evaluation of the model output** LLMs might generate the correct reasoning but might arrive at the wrong overall conclusion. Additionally, the model might have some incomplete knowledge of a situation, but not enough knowledge to arrive at the correct conclusion. For long-form answers, it would be possible to take the correct reasoning and partial knowledge into account during evaluation.
2. **Unlocking reasoning abilities** LLMs might be able to achieve better reasoning abilities when provided with the possibility to generate an elaborate response that can involve

reasoning steps. Moghaddam&Honey[14] claim that this is partially the reason for the increase in performance in their experiment.

It is true that long-form answers give more insights into the behavior of the model and enable the model to generate more elaborate answers.
However, both benefits come with a drawback:

1. **Evaluation becomes more difficult** According to the rules for evaluation in this experiment and the experiment of Moghaddam&Honey[14], a response is evaluated as correct, if a model generates a list of possible answers that also contain the correct answer. On the other hand, inconclusive answers are evaluated as incorrect. The correct answer for all ToM-questions are either "Yes" or "No". In the case of a model generating a list of possible answers, it is very likely that the model also generates the correct answer. The additionally provided explanations are mostly a repetition of the scenario or parts of the scenario and are mostly identical between all possible answers in the list. Reasoning steps, that could falsify the answer are rarely used. Evaluating answers of this type as correct would falsely attribute ToM capabilities to evasive behavior. Therefore, all model responses that involved a list of possible answers were labeled as inconclusive in this experiment. The difference in performance between the LLMs observed in this experiment and the LLMs used by Moghaddam&Honey[14] might be caused by a different interpretation of what constitutes a correct or incorrect answer.
2. **Lack of reasoning steps** Although long-form answers enable the use of reasoning steps, they do not guarantee their use. The observed models in this experiment rarely used any form of inductive reasoning to answer the ToM-question. Without the use of reasoning steps, it remains unclear if the observed models truly possess ToM reasoning capabilities. The lack ToM reasoning capabilities raise the question if the models possess any ToM capabilities or if the models simply detect an underlying pattern in the ToM question to solve it correctly.

The increased difficulty in evaluating and the lack of desired behavior partially undermine the positive aspects of long-form answers. Especially the lack of ToM reasoning raises the question if the conducted experiment is a viable method to assess the ToM capabilities of the observed LLMs.

5.2 Open Questions

- **Do the LLMs have ToM reasoning capabilities?**
  All provided ToM-questions can be solved by inductively inferring information from the scenario or the mental states of involved characters. Ideally, these intermediate reasoning steps lead to the correct answer to the question. However, in all models and for all prompting types the generated answers involve mostly a repetition or a summary of the provided scenario and no reasoning steps at all. The models rarely try to inductively infer information from the provided scenario or use any form of inductive reasoning. A simple repetition or summary of the scenario followed by a conclusion, which involves no form of inductive assumption, does not resemble a reasoning process. Therefore, this experiment provides little evidence that the observed models are truly capable of performing inferential inductive reasoning and therefore also ToM reasoning. However, the lack of ToM reasoning capabilities does not necessarily mean a general lack of ToM capabilities. The performance of some models and the lack of ToM reasoning could suggest that ToM reasoning capabilities are not a necessary and only a sufficient precursor for ToM capabilities. Since the LLMs can evade the use of inferential reasoning when generating long-form answers, the experiment might not be suitable for assessing the ToM reasoning capabilities of LLMs.
- **Do the LLMs have general ToM capabilities?**
  The lack of any inferential reasoning steps could indicate that the observed LLMs lack general ToM capabilities. However, ToM reasoning capabilities might not be a necessary precursor for ToM capabilities and the used LLMs might still possess ToM capabilities. Given the quite simple nature of the ToM-questions, it can be assumed that most humans would be able to solve the ToM-questions intuitively and not by applying inferential reasoning steps explicitly. Therefore, it might also be possible that the observed LLMs are capable of solving the ToM-questions in a way that does not involve the use of inductive reasoning. However, this leads to the next question.

- **Is the detected underlying pattern a confounding factor?**
  Some of the observed models manage to solve the ToM-tasks without the use of ToM reasoning. This suggests that the models are capable of detecting an underlying pattern in the ToM-task to answer the question correctly. It is possible that this capability of detecting patterns to solve ToM-tasks is a for of ToM itself. However, it is also possible that the underlying pattern, which is detected by the models, represents a confounding factor that was not adjusted. Therefore, it is hard to determine if the observed models have any ToM capabilities or if the high performance is simply caused by a confounding factor. This high level of uncertainty, caused by the lack of ToM reasoning, undermines the whole purpose of this experiment.
- **Does the experiment assess Theory-of-Mind?**
  The observed LLMs might be able to answer the ToM questions correctly by detecting an underlying pattern in the ToM-tasks. This pattern might be detectable by humans too. This means that the experiment does not necessarily assess the ToM capabilities of LLMs or humans. The performed experiment might therefore not be adequate to assess the ToM capabilities of LLMs and humans.

The lack of use of ToM reasoning steps makes it questionable if the LLMs truly possess any ToM capabilities since the presence of ToM reasoning capabilities would be a sufficient precursor of ToM capabilities in general. The lack of ToM reasoning can imply two possibilities:

1. The observed LLMs have ToM capabilities, which do not require the use of ToM reasoning, and the experiment is able to assess these ToM capabilities.
2. The observed LLMs have no ToM capabilities but are able to answer the ToM-tasks by abusing an underlying pattern. Therefore the experiment would not be able to assess ToM capabilities.

These questions could be further investigated by performing further experiments on the observed LLMs. Testing the ToM capabilities of the observed LLMs on single-word completion or multiple-choice Theory-of-Mind benchmarks. A correlation between performance on these benchmarks and the conducted experiment could indicate that the experiment is capable of assessing ToM and that the models truly have some ToM capabilities.
The ToM tasks of the experiment could also be changed to single-word completion tasks, in which intermediate reasoning steps are already predefined. The tasks of the model would be to correctly fill in the blanks. This way it could be verified if the observed models are capable of ToM reasoning. However, high performance on this single-word completion task would be meaningless if the models never uses ToM reasoning in practice.

6 Conclusion

It was shown that small LLMs ($\sim$7B) that were successfully trained with human preference methods were able to increase their performance in answering ToM questions by making use of in-context prompting. Both, chain-of-thought reasoning and step-by-step thinking were able to increase the performance, while the combined use yielded the highest increase. The increase was also observed for a model that was trained on human preferences without RLHF (GPT4All-J DPO) and a model trained without human preference finetuning (LLaMa2).

The ToM questions were answered without the use of ToM reasoning. This raises the question if the models have ToM capabilities, which do not require the use of ToM reasoning, or if the models abused an underlying pattern, to answer the ToM-questions correctly. Given this ambiguity, it is hard to conclude the true ToM capabilities of the observed LLMs.

The experiment was able to show the benefits of in-context prompting to solve complicated tasks but was not able to assess the Theory-of-Mind capabilities of the observed models. Therefore, further investigation is necessary to determine the ToM capabilities of the observed models.

References

[1] Alex Havrilla. synthetic-instruct-gptj-pairwise (revision cc92d8d), 2023.

[2] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

[3] F. Cuzzolin, A. Morelli, B. Cirstea, and B. J. Sahakian. Knowing me, knowing you: theory of mind in ai. *Psychological medicine*, 50(7):1057–1061, 2020.

[4] databricks. Dolly 15K, May 2023. URL: https://huggingface.co/datasets/databricks/databricks-dolly-15k.

[5] T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer. 8-bit optimizers via block-wise quantization, 2022.

[6] D. Dodell-Feder, J. Koster-Hale, M. Bedny, and R. Saxe. fMRI item analysis in a theory of mind task. *neuroimage*, 55(2):705–712, 2011.

[7] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, K. McDonell, N. Muennighoff, J. Phang, L. Reynolds, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, Sept. 2021. URL: https://github.com/EleutherAI/lm-evaluation-harness.

[8] C. Heyes. Animal mindreading: what's the problem? *Psychonomic bulletin & review*, 22:313–327, 2015.

[9] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[11] J. Huang and K. C.-C. Chang. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*, 2022.

[12] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[13] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[14] S. R. Moghaddam and C. J. Honey. Boosting theory-of-mind performance in large language models via prompting. *arXiv preprint arXiv:2304.11490*, 2023.

[15] NomicAI. GPT4ALL-J, Apr. 2023. URL: https://huggingface.co/nomic-ai/gpt4all-j.

[16] NomicAI. GPT4ALL-J dataset, Apr. 2023. URL: https://huggingface.co/datasets/nomic-ai/gpt4all-j-prompt-generations.

[17] OpenAI. Model overview, 2023. URL: https://platform.openai.com/docs/models/overview.

[18] OpenAI. GPT-4 technical report, 2023.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[20] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[21] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[22] RyokoAI. ShareGPT52K, Apr. 2023. URL: https://huggingface.co/datasets/RyokoAI/ShareGPT52K.

[23] M. Sap, R. LeBras, D. Fried, and Y. Choi. Neural theory-of-mind? on the limits of social intelligence in large lms. *arXiv preprint arXiv:2210.13312*, 2022.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[25] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot, and R. Anderson. The curse of recursion: Training on generated data makes models forget, 2023.

[26] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient founda*arXiv preprint arXiv:2302.13971*, 2023.

[27] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[28] T. Ullman. Large language models fail on trivial alterations to theory-of-mind tasks. *arXiv preprint arXiv:2302.08399*, 2023.

[29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[30] C. Westby and L. Robinson. A developmental perspective for promoting theory of mind. *Topics in language disorders*, 34(4):362–382, 2014.

7 Supplement

7.1 Detailed Results

Model	Prompting	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	Σ
GPT4All-J v1.3	Base	1	20	19	0	16	8	0	0	15	18	20	19	20	0	8	0	164
	SbS	0	5	15	2	8	6	2	0	5	9	19	6	2	5	13	3	100
	CoT	7	13	14	1	4	17	2	11	7	11	7	12	12	7	11	10	146
	SbS+CoT	3	14	11	0	2	20	17	4	10	13	16	9	7	1	9	7	143
GPT4All-J PPO	Base	5	2	20	15	8	20	5	5	14	19	18	12	13	20	20	2	198
	SbS	3	15	18	0	7	0	4	1	15	7	15	7	13	0	6	0	111
	CoT	20	3	9	7	0	16	9	20	1	19	20	4	0	20	14	9	171
	SbS+CoT	13	7	15	6	4	11	8	8	18	14	14	8	2	6	14	10	158
GPT4All-J DPO	Base	0	20	20	2	6	0	0	0	0	19	0	0	20	0	0	0	87
	SbS	0	1	20	0	5	2	5	3	14	15	10	14	19	0	3	0	111
	CoT	0	15	20	0	19	7	0	0	8	20	0	14	20	19	15	0	157
	SbS+CoT	10	18	19	2	15	17	7	18	14	18	13	3	5	9	5	18	191
LLaMa2	Base	9	5	9	4	5	7	4	0	4	6	10	7	8	14	11	7	110
	SbS	0	8	18	5	12	2	0	0	9	11	14	12	15	15	12	4	137
	CoT	20	5	4	14	3	19	15	15	16	13	2	5	4	13	20	11	179
	SbS+CoT	18	4	3	11	7	19	18	15	8	18	8	0	5	15	19	17	185
LLaMa2-Chat	Base	8	11	7	2	5	1	2	0	20	3	8	6	9	4	0	2	88
	SbS	20	5	17	11	16	20	6	7	17	20	7	1	4	17	11	5	184
	CoT	20	2	14	20	1	17	15	16	6	13	2	10	19	20	20	20	215
	SbS+CoT	18	18	20	16	2	20	17	19	11	14	19	10	14	20	18	12	248

Table 3: Overview over the correct answers of each model for each type of prompting on each question.

7.2 Statistical Tests

7.2.1 GPT4All-J v1.3

- Performance difference: (Paired T-test, $N = 16$)
  - Base vs SbS: $p = 0.0442^*$
    ($\mu_{\text{Base}} = 10.25, SEM_{\text{Base}} = 2.21, \mu_{\text{SbS}} = 6.25, SEM_{\text{SbS}} = 1.36$)
  - Base vs CoT: $p = 0.5885$
    ($\mu_{\text{Base}} = 10.25, SEM_{\text{Base}} = 2.21, \mu_{\text{CoT}} = 9.13, SEM_{\text{CoT}} = 1.10$)
  - Base vs CoT+SbS: $p = 0.5503$
    ($\mu_{\text{Base}} = 10.25, SEM_{\text{Base}} = 2.21, \mu_{\text{CoT+SbS}} = 8.94, SEM_{\text{CoT+SbS}} = 1.50$)
- Yes/No-bias: (Unpaired T-test, $N = 8$)
  - Base: $p = 0.0001^*$
    ($\mu_{\text{Yes}} = 18.38, SEM_{\text{Yes}} = 0.68, \mu_{\text{No}} = 2.13, SEM_{\text{No}} = 1.29$)
  - SbS: $p = 0.0793^*$
    ($\mu_{\text{Yes}} = 8.63, SEM_{\text{Yes}} = 2.01, \mu_{\text{No}} = 3.88, SEM_{\text{No}} = 1.51$)
  - CoT: $p = 0.4449$
    ($\mu_{\text{Yes}} = 10.00, SEM_{\text{Yes}} = 1.25, \mu_{\text{No}} = 8.25, SEM_{\text{No}} = 1.84$)
  - SbS+Cot: $p = 0.4015$
    ($\mu_{\text{Yes}} = 10.25, SEM_{\text{Yes}} = 1.56, \mu_{\text{No}} = 7.63, SEM_{\text{No}} = 2.60$)

7.2.2 LLaMa2

- Performance difference: (Paired T-test, $N = 16$)
  - Base vs SbS: $p = 0.1927$
    ($\mu_{\text{Base}} = 6.88, SEM_{\text{Base}} = 0.84, \mu_{\text{SbS}} = 8.56, SEM_{\text{SbS}} = 1.50$)
  - Base vs CoT: $p = 0.0337^*$
    ($\mu_{\text{Base}} = 6.88, SEM_{\text{Base}} = 0.84, \mu_{\text{CoT}} = 11.19, SEM_{\text{CoT}} = 1.60$)
  - Base vs CoT+SbS: $p = 0.0195^*$
    ($\mu_{\text{Base}} = 6.88, SEM_{\text{Base}} = 0.84, \mu_{\text{CoT+SbS}} = 11.56, SEM_{\text{CoT+SbS}} = 1.64$)
- Yes/No-bias: (Unpaired T-test, $N = 8$)
  - Base: $p = 0.8871$
    ($\mu_{\text{Yes}} = 6.75, SEM_{\text{Yes}} = 0.75, \mu_{\text{No}} = 7.00, SEM_{\text{No}} = 1.56$)
  - SbS: $p = 0.0058^*$
    ($\mu_{\text{Yes}} = 12.38, SEM_{\text{Yes}} = 1.15, \mu_{\text{No}} = 4.75, SEM_{\text{No}} = 2.04$)

- CoT: $p = 0.0007^*$
  ($\mu_{\text{Yes}} = 6.50, SEM_{\text{Yes}} = 1.80, \mu_{\text{No}} = 15.88, SEM_{\text{No}} = 1.20$)
- SbS+Cot: $p = 0.0004^*$
  ($\mu_{\text{Yes}} = 6.63, SEM_{\text{Yes}} = 1.89, \mu_{\text{No}} = 16.50, SEM_{\text{No}} = 0.96$)

7.2.3 LLaMa2 Chat

- Performance difference: (Paired T-test, $N = 16$)
  - Base vs SbS: $p = 0.0096^*$
    ($\mu_{\text{Base}} = 5.50, SEM_{\text{Base}} = 1.28, \mu_{\text{SbS}} = 11.50, SEM_{\text{SbS}} = 1.65$)
  - Base vs CoT: $p = 0.0095^*$
    ($\mu_{\text{Base}} = 5.50, SEM_{\text{Base}} = 1.28, \mu_{\text{CoT}} = 13.44, SEM_{\text{CoT}} = 1.77$)
  - Base vs CoT+SbS: $p = 0.0001^*$
    ($\mu_{\text{Base}} = 5.50, SEM_{\text{Base}} = 1.28, \mu_{\text{CoT+SbS}} = 15.50, SEM_{\text{CoT+SbS}} = 1.22$)
- Yes/No-bias: (Unpaired T-test, $N = 8$)
  - Base: $p = 0.0089^*$
    ($\mu_{\text{Yes}} = 8.63, SEM_{\text{Yes}} = 1.84, \mu_{\text{No}} = 2.38, SEM_{\text{No}} = 0.92$)
  - SbS: $p = 0.7179$
    ($\mu_{\text{Yes}} = 10.88, SEM_{\text{Yes}} = 2.60, \mu_{\text{No}} = 12.13, SEM_{\text{No}} = 2.17$)
  - CoT: $p = 0.0011^*$
    ($\mu_{\text{Yes}} = 8.38, SEM_{\text{Yes}} = 2.35, \mu_{\text{No}} = 18.50, SEM_{\text{No}} = 0.76$)
  - SbS+Cot: $p = 0.1017$
    ($\mu_{\text{Yes}} = 13.50, SEM_{\text{Yes}} = 2.09, \mu_{\text{No}} = 17.50, SEM_{\text{No}} = 0.93$)

7.2.4 GPT4All-J PPO

- Performance difference: (Paired T-test, $N = 16$)
  - Base vs SbS: $p = 0.0240^*$
    ($\mu_{\text{Base}} = 12.38, SEM_{\text{Base}} = 1.73, \mu_{\text{SbS}} = 6.94, SEM_{\text{SbS}} = 1.59$)
  - Base vs CoT: $p = 0.4580$
    ($\mu_{\text{Base}} = 12.38, SEM_{\text{Base}} = 1.73, \mu_{\text{CoT}} = 10.69, SEM_{\text{CoT}} = 1.94$)
  - Base vs CoT+SbS: $p = 0.1639$
    ($\mu_{\text{Base}} = 12.38, SEM_{\text{Base}} = 1.73, \mu_{\text{CoT+SbS}} = 9.88, SEM_{\text{CoT+SbS}} = 1.12$)
- Yes/No-bias: (Unpaired T-test, $N = 8$)
  - Base: $p = 0.6291$
    ($\mu_{\text{Yes}} = 13.25, SEM_{\text{Yes}} = 2.14, \mu_{\text{No}} = 11.50, SEM_{\text{No}} = 2.82$)
  - SbS: $p = 0.0001^*$
    ($\mu_{\text{Yes}} = 12.13, SEM_{\text{Yes}} = 1.57, \mu_{\text{No}} = 1.75, SEM_{\text{No}} = 0.82$)
  - CoT: $p = 0.0536$
    ($\mu_{\text{Yes}} = 7.00, SEM_{\text{Yes}} = 2.92, \mu_{\text{No}} = 14.38, SEM_{\text{No}} = 1.94$)
  - SbS+Cot: $p = 0.7497$
    ($\mu_{\text{Yes}} = 10.25, SEM_{\text{Yes}} = 2.04, \mu_{\text{No}} = 9.50, SEM_{\text{No}} = 1.07$)

7.2.5 GPT4All-J DPO

- Performance difference: (Paired T-test, $N = 16$)
  - Base vs SbS: $p = 0.4471$
    ($\mu_{\text{Base}} = 5.44, SEM_{\text{Base}} = 2.17, \mu_{\text{SbS}} = 6.94, SEM_{\text{SbS}} = 1.80$)
  - Base vs CoT: $p = 0.0290^*$
    ($\mu_{\text{Base}} = 5.44, SEM_{\text{Base}} = 2.17, \mu_{\text{CoT}} = 9.81, SEM_{\text{CoT}} = 2.18$)
  - Base vs CoT+SbS: $p = 0.0109^*$
    ($\mu_{\text{Base}} = 5.44, SEM_{\text{Base}} = 2.17, \mu_{\text{CoT+SbS}} = 12.06, SEM_{\text{CoT+SbS}} = 1.53$)
- Yes/No-bias: (Unpaired T-test, $N = 8$)
  - Base: $p = 0.0108^*$
    ($\mu_{\text{Yes}} = 10.63, SEM_{\text{Yes}} = 3.52, \mu_{\text{No}} = 0.25, SEM_{\text{No}} = 0.25$)

- SbS: $p = 0.0006^*$
  ($\mu_{\text{Yes}} = 12.25, SEM_{\text{Yes}} = 2.33, \mu_{\text{No}} = 1.63, SEM_{\text{No}} = 0.68$)
- CoT: $p = 0.0256^*$
  ($\mu_{\text{Yes}} = 14.50, SEM_{\text{Yes}} = 2.55, \mu_{\text{No}} = 5.13, SEM_{\text{No}} = 2.75$)
- SbS+Cot: $p = 0.4514$
  ($\mu_{\text{Yes}} = 13.13, SEM_{\text{Yes}} = 2.13, \mu_{\text{No}} = 10.75, SEM_{\text{No}} = 2.20$)