

# **Master Computer Science**

BABA IS LLM: Large Language Models Play Baba is You

Name:Fien van Wetten<br/>s2269651Date:[24/03/2025]Specialisation:Artificial Intelligence1st supervisor:Prof. dr. A. Plaat<br/>Dr. M.J. van Duijn

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS) Leiden University Niels Bohrweg 1 2333 CA Leiden The Netherlands

#### Abstract

Artificial intelligence (AI) has long been tested through games, which serve as a benchmark for reasoning and problem-solving. This paper explores the capabilities of Large Language Models (LLMs) in playing the 2D puzzle game *Baba is You*, a game in which players manipulate rules by rearranging text blocks that define object properties and behaviours. Unlike traditional games with fixed rules, *Baba is You* requires dynamic rule changes, making it a compelling challenge for LLMs.

This research investigates whether LLMs can understand and play *Baba is You*, focusing on their ability to manipulate rules and solve puzzles. Two approaches are investigated: prompt-based learning and fine-tuning. Six LLMs — GPT-4o, Gemini Flash 1.5, Mixtral 8x7B, Mistral 7B, OLMo 7B and OLMo 13B — are evaluated using different prompts, including simple, rule-extended and action-extended prompts. In addition, the Mistral and OLMo models are fine-tuned using textual and structural data extracted from the game.

Experimental results show that while larger models such as GPT-40 perform better in reasoning and puzzle solving, smaller models struggle to recognise game mechanics and apply rule changes. Fine-tuning improves on analysing the levels, but does not significantly improve solution formulation. The results highlight the potential and limitations of LLMs in dynamic rule-based environments and provide insights into their applicability to complex problem-solving tasks.

#### Acknowledgements

This Master thesis was written as a final project for the computer science Master at the Leiden Institute of Advanced Computer Science (LIACS), and was supervised by Prof. dr. A. Plaat and Dr. M.J. van Duijn. This work was performed using the compute resources from the Academic Leiden Interdisciplinary Cluster Environment (ALICE) provided by Leiden University. The code developed for this research, along with the fine-tuned LLMs and the custom implementation of *Baba is You*, is available in the referenced repositories [22, 27, 44].

# Contents

1	Introduction		
	1.1	$\label{eq:Research} \mbox{Research question}  \ldots  \ldots  \ldots  \ldots  \ldots  \ldots  2$	
	1.2	Thesis overview $\ldots \ldots 2$	
2	Related work		
	2.1	Prompt-based learning	
	2.2	Fine-tuning	
	2.3	Techniques for updating pretrained LLM weights $\ldots \ldots \ldots$	
		2.3.1 Full Fine-tuning	
		2.3.2 Parameter-efficient Fine-tuning (PEFT)	
	2.4	Puzzling with LLMs	
3	Enviror	nment	
	3.1	ASCII representation	
4	Metho	<b>d</b>	
	4.1	LLM models $\ldots \ldots \ldots$	
	4.2	Prompt-based learning	
	4.3	Evaluation of Reasoning	
	4.4	Levels	
	4.5	Fine-tuning	
		4.5.1 Parameter settings	
5	Results	23	
	5.1	Prompt-based learning	
		5.1.1 Simple prompt (1)	
		5.1.2 Rule-extended prompt (2) and Action-extended prompt (3) . 23	
	5.2	Fine-tuning	
		5.2.1 Mistral 7B	
		5.2.2 OLMo 7B	
6	Discuss	sion & Conclusion $\ldots \ldots 33$	
	6.1	Limitations & Further work	
References			

# 1 Introduction

Artificial Intelligence (AI) has a long history in games. It has been a benchmark for reasoning, decision-making, and problem-solving. From early games like chess, to complex dynamic environments [3, 5, 28, 33–35], games provide a controlled but challenging testbed for AI research. This thesis investigates the use of large language models (LLMs) in the context of the 2D puzzle game *Baba is You* [37]. Unlike conventional games, *Baba is You* allows players to manipulate the rules of the game environment by rearranging text blocks that define object properties and behaviours. Success in the game requires players to dynamically adapt their strategies by analyzing existing rules, breaking them, or creating new ones to achieve the desired outcome.

LLMs have attracted significant attention because of their capabilities in natural language processing (NLP). They excel in tasks across various domains including text generation, machine translation, conversational agents and code generation [23]. The transformer architecture revolutionized NLP by introducing the self-attention mechanism [40]. This mechanism improved the contextual understanding of the model compared to earlier architectures such as recurrent neural networks (RNNs) and long short-term memory networks (LSTM). LLMs are trained on massive text data using unsupervised or semi-supervised learning techniques. Where the model learns to generate or predict missing tokens based on the surrounding context. To improve the output of LLMs different techniques are used, such as fine-tuning [45], reinforcement learning with human feedback (RLHF) [8] and prompt-based learning [17]. Fine-tuning adapts a pre-trained model to specific tasks or domains. RLHF helps with creating better outputs by incorporating human-generated evaluations to refine the model's responses. Lastly, prompt-based learning can be used to optimize the model's output by crafting good input prompts.

LLMs have shown promise in handling various tasks that involve language understanding and generation, but their use extends beyond typical language processing tasks. LLMs are also used as agents in games [1, 38]. Baba is You is particularly interesting for LLMs due to its language manipulation mechanics. Its dynamic rule changes make the game unique. Firstly, in most games rules are fixed, but here they constantly change. Secondly, a core mechanic of Baba is You is breaking, modifying, and creating rules by rearranging words in the game world. This requires understanding how language shapes the mechanics in the game. The model then has to use this understanding to solve puzzles. Since LLMs are trained on large amounts of text, they recognizing patterns in language bit according to Mirchandani et al. [21] LLMs are not just language models but general pattern models, meaning that their capabilities go beyond mere language pattern recognition. The game Baba is You goes beyond simple pattern recognition. To succeed, an LLM must understand how rule changes affect the game state and apply that understanding. Some solutions require unconventional thinking, where the player (or AI) must redefine what is possible within the logic of the game. This makes Baba is You a compelling test for LLMs. Not only does it challenge their ability to follow instructions, but also their ability to learn, apply, and manipulate context-based rules, as well as for solve problems creatively.

While LLMs perform well on some reasoning tasks [18, 47], few studies examine their effectiveness in games like *Baba is You*. A study by Nathan Cloos et al. [9] found that state-of-the-art LLMs struggle with generalizing rule manipulations. This suggests a need for further exploration of fine-tuning and prompting techniques.

In this thesis, we tested the two types of learning. We first test how LLMs do out of the box on *Baba is you* with prompt-based learning. Secondly, we fine-tune two different LLMs. For prompt-based learning, we show the result of six different LLMs: GPT-4o [13], Gemini-Flash 1.5 [36], OLMo 2 13B [26], OLMo 2 7B [26], Mixtral 8x7B [16] and Mistral 7B [15]. For each LLM, three different prompts were tested. We fine-tuned the Mistral 7B and OLMo 7B LLMs with textual data from the game *Baba is You*. Our choice of GPT-4o is based on its status as one of the current state-of-the-art LLMs as well as its branding as a better reasoning model, while Gemini-Flash 1.5 is not branded as a reasoning model, it ranks highly as a promising alternative. We selected Mistral and OLMo because they are relatively easy to fine-tune and, despite their smaller size, have shown strong performance compared to other models in their category.

## 1.1 Research question

This research investigates the ability of LLMs to play *Baba is You* assessing their capacity to understand and manipulate in-game rules. Level solving refers to identifying the problem within a level and generating a correct solution based on the given rules, while rule manipulation is a tool that allows players to break existing rules or create new rules. By analyzing their performance across different puzzle types and prompting techniques, this study provides insights into the limitations and potential of language-based AI in complex problem-solving environments. Our main problem statement is:

Is an LLM able to understand and play the game Baba is You?

We operationalize this into the following research questions:

**RQ1:** How does the choice of pre-trained LLMs affect the performance on level-solving tasks in *Baba is You*? .

**RQ2:** To what extent can an LLM create new rule manipulations and apply them in the game *Baba is You* to solve the puzzles?

**RQ3:** How does fine-tuning with textual and structural data from *Baba is You* affect the performance of an LLM in learning to play the game compared to models without such fine-tuning?

### 1.2 Thesis overview

The thesis is structured as follows: Section 2 provides the background information. Section 3, describes the game mechanics of the game *Baba is You*. Section 4 outlines the methodology used. Section 5 presents the experiments and their outcome. Finally, Section 6 concludes the thesis.

# 2 Related work

### 2.1 Prompt-based learning

We will now explain why it is interesting to study a reasoning game, by looking at step by step LLM reasoning. With the advent of LLMs, a new type of learning has emerged: Prompt-based Learning [17]. This type of learning occurs at inference time, which means that the model parameters are not changed during training. The general idea behind prompt-based learning is to provide the LLM with a task-relevant prompt accompanied by a question. The prompt consists of a task instruction, which is a description of the task the model should perform. Next, there are examples, which are input-output pairs that demonstrate the task. They assist the model in recognizing the pattern or relationship between inputs and expected outcomes. The prompt does not always include examples, depending on the type of learning being used. Finally, there is a query prompt, which typically includes a new input or query for the model to solve. Depending on the number of examples provided in the prompt, three distinct learning types emerge: Zero-shot learning, one-shot learning, and few-shot learning [4].

Zero-shot prompt-based learning refers to a model's ability to perform a task without seeing any examples of it during inference. In this case, the model generates an output based solely on the task description and query provided by the prompt. One-shot prompt-based learning gives the model a single example of the task, in addition to the task description and query. Few-shot prompt-based learning, as the name implies, involves providing the model a few samples (usually 2-5) in the prompt, as well as the task description and inquiry. [4].





After this, the idea of having the model generate intermediate steps, work through these steps and then answer the final question is raised. To unlock these reasoning abilities, letting LLMs generate intermediate steps, or a chain of thought (CoT), can be achieved in several ways.

The first attempt was through supervised learning [25] after this few-shot CoT prompt was tried [43]. The general idea is that the prompt is created by including a series of demonstrations or examples that consist of three main components: the input, the chain of thought, and the output. Here is how the prompt is constructed: The prompt begins by presenting a task or

question (such as a mathematical word problem, a common sense question, or a symbolic reasoning task). This is the initial input to the model to be solved. After the input, instead of providing the answer directly, a sequence of natural language reasoning steps is introduced. These steps break down the process required to arrive at the final answer. The reasoning is clearly explained step by step, similar to how a human would think through the problem. The final part of each example is the correct output, which represents the final answer to the task after going through the intermediate reasoning steps (Figure 1). The paper concludes that CoT improves performance in solving complex reasoning tasks on sufficiently large language models.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: Let's think step by step.
(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.

Figure 2: Example of Zero-shot CoT prompting [19, Fig. 1].

Koijma et al. [19] propose a zero-shot CoT template for reasoning. The difference with Wei et al. [43] is that it does not require a few examples in the prompt, they unlock the reasoning step by adding the "Let's think step by step" sentence at the end of each prompt see Figure 2. This prompting method involves two steps: the reasoning extraction and the answer extraction. The reasoning extraction exists of the input question and the "Let's thinks step by step" sentence. The answer extraction exists of the prompt sentence created in the reasoning extraction, the output of the reasoning extraction prompt and the sentence to trigger the answer. This prompt is self-augmented because it contains the sentence that is generated by the same model.

Wang et al. [42] introduce a plan and solve prompting (PS) see Figure 3. A new zero-shot CoT prompting method has been developed which enables LLM systems to formulate a plan for solving a given problem and generate the intermediate reasoning process before making a final prediction for the input problem. Like Zero-shot-CoT, PS Prompting involves two steps as well. The proposed prompt template begins with an inference in step 1 to create the reasoning process and solution to a problem. In the second stage, it collects the responses to be evaluated using the answer extraction prompt. The prompt consist of a question followed by "Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step." They extend this sentence with more detailed instructions to reduce errors in the reasoning step.

Another approach is self-ask [29] (Figure 4). This builds on one- or few-shot CoT but instead of using examples of the Chain-of-Thought the prompt has to explicitly state the next follow-up question it wants to ask before answering it. Self-ask uses also one- or few-shot prompt that shows the question and how to formulate follow up questions and answers and ends with the question that wants to be answered together with "Are follow up questions needed here:". The model starts with giving follow-up questions and answers until it thinks it has enough information to answer the original question and gives the final answer.



Figure 3: Example with Plan-and-Solve (PS) prompting [42, Fig. 2].



Figure 4: Example prompt with self-ask [29, Fig. 2].

# 2.2 Fine-tuning

Another approach to learning a pre-trained LLM to perform a new task is fine-tuning, where the parameters of the model are adjusted [14]. Fine-tuning is used to train LLMs to specialise in specific tasks or domains. LLMs are initially trained on large amounts of general data, resulting in a lack of precision in specific tasks; fine-tuning bridges this gap by updating the model's parameters using smaller domain-specific data sets. This method relies on the existing knowledge of the model to improve performance on specific tasks using less data and computational resources. Fine-tuning involves transferring the patterns and features learned by the pre-trained model to new tasks, improving performance and minimising the amount of training data required. One of the most commonly used techniques for fine-tuning an LLM is supervised fine-tuning (SFT), which uses labelled data to train the model for the target task.

Instruction-tuning is a form of SFT that involves feeding natural language instructions to the LLM. The LLM learns to interpret these instructions, allowing it to perform specific tasks without requiring huge amounts of labelled data for each task. The data consists of *(Instruction, Output)* pairs where the *Instruction* is a human instruction and the *Output* is the desired response by the LLM for that instruction [48].



Figure 5: Different PEFT methods [11, Fig. 4].

# 2.3 Techniques for updating pretrained LLM weights

### 2.3.1 Full Fine-tuning

Full tuning involves updating all the parameters in the model. This technique can be costly and time consuming, as the pre-trained model often contains billions of parameters that need to be updated. This fine-tuning method, seen in models such as OpenAI's Generative Pre-trained Transformer (GPT) [31], involves minimal addition of task-specific parameters. Instead, it adapts the model to new tasks by adjusting all pre-trained parameters through direct fine-tuning on the target tasks.

#### 2.3.2 Parameter-efficient Fine-tuning (PEFT)

Parameter-efficient fine-tuning (PEFT) is an alternative to full fine-tuning. PEFT is more resource and time efficient, since it focuses on adjusting a minimal set of parameters to enhance performance compared to full fine-tuning on downstream tasks [11].

Reparametrized fine-tuning is a category of PEFT (Figure 5c) that introduces new low-rank trainable parameters during training, which are later merged with the original model. A notable method in this category is Low-Rank Adaptation (LoRA) [12], an approach that aims to improve the efficiency of fine-tuning by reducing the number of parameters that need to be updated. Instead of modifying the entire model, LoRA approximates the difference between pre-trained and task-specific weights using a low-rank matrix. This technique not only minimises memory and computational requirements, but also results in a more compact set of fine-tuned weights.

The efficiency of LoRA comes from its use of low-rank decomposition, where large weight updates are replaced by the multiplication of two smaller matrices. By updating only the smaller matrices during fine-tuning, LoRA significantly reduces the computational load.

### 2.4 Puzzling with LLMs

The rise of AI agents achieving dominance in gaming begins in the 1990s with Deep Blue [5], as demonstrated in the match against Garry Kasparov. IBM's Deep Blue made history by defeating world chess champion Garry Kasparov in a match under standard time controls, marking the first time a computer surpassed a reigning champion in chess. This achievement showcased the potential of brute-force search combined with domain expertise.

As AI matured, research shifted the focus on solving entire games Schaeffer, J., et al. [32] solved the game of checkers. Demonstrating that with optimal play the game always ends in a draw. This accomplishment required an exhaustive search of the game's state space and served as a milestone in deterministic game-solving.

After this there was a shift in AI agents playing games. Machine and reinforcement learning was used to tackle increasingly complex games. DeepMind's AlphaGo [34] defeated world champion Lee Sedol in Go by combining deep neural networks with Monte Carlo Tree Search (MCTS). This was followed by DeepMind's AlphaZero [35], which mastered Chess, Shogi and Go without any domain-specific knowledge. It relied solely on self-play and reinforcement learning. This demonstrated the power of general-purpose algorithms to achieve superhuman performance across games. Another milestone was DeepMind's MuZero [33]. An evolution of AlphaZero. MuZero mastered chess, shogi and Atari video games without being explicitly programmed with the rules of these games. Instead, it learned both a model of the environment and optimal strategies through reinforcement learning. This demonstrated the potential of model-based learning to solve problems in unknown or partially understood domains.

With the introduction of LLMs a new kind of AI emerged in the field of AI agents playing games. This development has opened up new possibilities using LLMs as agents in various games. An example is ChessGPT [10], a substantial game and language dataset for chess has been created, upon which two models, ChessCLIP and ChessGPT, have been introduced. Furthermore, an evaluation strategy has been devised to assess the models' capabilities in terms of modeling, value judgment and policy.

In a study by Li et al. [20], the ability of LLMs to play Minesweeper was evaluated. The study compared two ways of inputting information – tabular and coordinate representations – to determine how they affected LLM performance. While GPT-3.5 demonstrated a basic grasp of the game, techniques such as few-shot prompting only improved performance marginally. By contrast, GPT-4 was better at identifying mines but was not always able to complete a level. This finding underscores the efficacy of Minesweeper as a reliable metric for assessing the problem-solving capabilities of LLMs.

Noever and Burdick [24] introduce an approach to the solution of puzzles such as mazes, Sudoku and the Rubik's Cube. This method employs GPT-2 to generate solutions to these puzzles, a technique which is distinct from both traditional search-based and heuristic-driven approaches. The method involves training a language model on a database of solved puzzles, thereby enabling it to generate plausible solutions without the need for explicit human guidance. This approach demonstrates the potential of generative models to address complex problems in combinatorial spaces, highlighting a shift from search-based methods to text-based reasoning.

The study by Xu et al. [46] propose a framework, using tuning-free LLM agents to play the game Werewolf, which is a communication game. They investigate how historical experiences shape the capabilities of LLMs. Interestingly, they notice the emergence of unplanned strategic behaviors in LLMs during gameplay, including trust, confrontation, camouflage, and leadership. These studies highlight the growing potential of LMMs in solving puzzles and playing games across various domains.

# **3** Environment

The game *Baba is You* [37] is a 2D puzzle game with levels, a level is represented as a grid filled with objects and text blocks (see Figure 6). Text blocks can be used to create rules; these rules can be created from left to right or from top to bottom. The rule is active if there is at least one object, one verb, and one object or property aligned in a valid syntax (Figure 7).

Objects are elements in the game, for example BABA, WALL or ROCK, verbs are IS or HAS and properties determine the action of objects, for example PUSH, STOP, WIN or YOU. Rules in the game determine how objects in the game behave, and these rules can be created or broken during the level, except for TEXT IS PUSH, which means that text tiles can always be pushed, and BORDER IS STOP, which means that it is not possible to cross the border of the grid, these rules are always on by default without text tiles saying so. Some rules cannot be broken for the whole level if they are placed against the border or in an inaccessible corner. The level is completed when the object that is part of the IS YOU rule touches the object that is part of the IS WIN rule, or when the is you object itself is also IS WIN. An important aspect of the game is that if the IS YOU rule is broken, or if there are no IS YOU objects left in the game, the player cannot continue and should restart the level or redo an action.

Every solvable level should have a win condition and a object that can be controlled by the player. In this game an object does not really matter until there is a rule assigned to the object. If there are multiple objects of the same kind they all act the same when a rule is assigned to it. The three basic formats of the rules are:

- < Object > IS < Property >. This rule gives an object a property and determines how an object behaves in the game.
- < Object1 > IS < Object2 >. This means that all instances of object 1 transforms into object 2.
- < Object1 > IS < Object1 >. This rule sets an object to be itself and therefore override the transformation rule.

The most common properties and rules are explained in Table 1.



Figure 6: Example level with a baba and flag object. Where the baba object is controlled by the player because of the rule BABA IS YOU and the flag object has no rule assigned to it yet.



Figure 7: Same level where the rule FLAG IS WIN is created, meaning that the flag object has become the win object. To win this level baba needs to move to the tile where the flag object is located.

<object 1> IS	Transforms all instances of object 1 into object 2
<object $2>$	
<object 1> IS	Set an object to be itself and therefore becomes immutable for the
<object 1=""></object>	transformation rule.
<object> IS</object>	Makes the object the win condition anything controlled by you
WIN	touches or is the object wins.
<object> IS</object>	Objects that are set to be YOU can be controlled by you. All
YOU	objects will move simultaneously.
<object> IS</object>	Objects set to PUSH can be moved by either an object controlled
PUSH	by you or other movable objects, shifting them one tile forward
	and occupying the new tile if possible.
<object> IS</object>	Objects move one space per turn in their facing direction, reversing
MOVE	direction if they hit an object, and if an object is both YOU and
	MOVE, it moves two spaces per turn.
<object> IS</object>	Objects becomes impassible, important is that nature of the PUSH
STOP	rule overrides the blocking mechanism.
< object > IS	Objects that are set to SINK will destroy themselves as well as any
SINK	other object that comes in contact with them.
<object> IS</object>	Objects set to HOT will destroy anything set to MELT that
HOT, <object></object>	touches them. If an object is both HOT and MELT, it will destroy
IS MELT	itself.
<object> IS</object>	Objects set to KILL destroy all objects controlled by you when
KILL	they touch them, while remaining intact themselves. Note that the
	nature of the PUSH rule overrides this mechanic.

Table 1: Explanation of common rules in the game BABA IS YOU.

## 3.1 ASCII representation

For the research we used a simplified version of the game, in this version we used the jam version of *Baba is You*<sup>1</sup> and a section of the game mechanics of "Game Module" from Baba is Y'all [7]. A Python version of parts of the "Game Module" was created and adapted. The functionality of the program is outlined as follows: an input string of characters, where each character corresponds to an object or text tile in the level, forms an ASCII representation of the level, see figure 8. A sequence of moves is then given to the program, and the level is loaded, with the sequence of moves being played through the level, updating the currently active rules after each move. The game ends when the sequence of moves is completed or when the winning condition is met. The use of ASCII representation allows for a straightforward representation of the levels to an LLM.



Figure 8: ASCII representation of a level.

In the simplified version of the game, the player is able to perform four distinct actions. Firstly, the player can move the object they are controlling towards an object or text block. Secondly, the player can break a rule, which involves pushing a text block away from the rule with the object they are controlling, thereby the rule becomes inactive. Thirdly, the player can make a rule, which involves aligning text blocks with the object they are controlling so that it has a valid syntax, thereby it becomes an active rule. The final action involves the act of pushing objects or text blocks with the object under control; however, it should be noted that objects can only be pushed if they are set to PUSH by a rule. Making a rule consists of a sequence of push to and move to actions (Figure 9).

<sup>&</sup>lt;sup>1</sup>https://hempuli.itch.io/baba-is-you



(a) Level 4: A flag object is the win condition but there is currently no flag object to reach.



(c) Level 6: The flag object, which serves as the win condition, is surrounded by pushable rock objects.



(e) Level 12: The path towards the flag object (win condition) is blocked by lava because its hot and baba object is set to melt.



(b) The rule ROCK IS FLAG is created and the rock object is transformed into a flag object.



(d) Two rock objects are pushed away by the baba object to clear the path to the flag object.



(f) The rule BABA IS MELT is broken this means baba will not melt when touching the lava. This way the path toward the flag object is cleared.

Figure 9: Examples of different game mechanics using the flag as a win condition. Each sequence shows how obstacles are manipulated or rules are changed to create a path to the flag.

 $\Rightarrow$ 

 $\Rightarrow$ 

 $\Rightarrow$ 

# 4 Method

In this chapter, we describe the methodologies used in our study, including data collection, experimental setup, and analysis techniques.

### 4.1 LLM models

We use six different LLMs for this research. The first is GPT-40 [13], developed by OpenAI. It is considered a state-of-the-art model due to its strong performance on various tasks, making it a valuable candidate for this study. Secondly, GPT-40 is the beginning of a transition where a model can reason at inference time [39], which is where it differs from the other models. OpenAI has not disclosed the exact number of parameters. GPT-40 has an input context window of 128K tokens, allowing it to process large amounts of information at once.

Gemini 1.5-Flash [36], developed by Google DeepMind, is designed for cost efficiency and fast inference. Its exact number of parameters has not been disclosed. It features a long context window that supports up to 1M tokens by default. This allows it to handle extended text histories, processing over 700,000 words in a single prompt. The model is optimized for speed, prioritizing fast inference.

Mistral 7B Instruct[15], created by Mistral AI, is a model with 7 billion parameters, making it relatively compact for an LLM. The "Instruct" designation indicates fine-tuning for instruction-following tasks. Another model from Mistral AI, Mixtral 8x7B[16], as a Mixture of Experts (MoE) architecture with a total of 45 billion parameters, but only 12 billion parameters are active at a time, as it selects two out of eight experts per prompt. This design reduces computational cost while maintaining efficiency. Both models have a 32K token context window.

The OLMo models [26] (7B and 13B) were developed by the Allen Institute for AI (AI2) to support open research. OLMo 7B has 7 billion parameters and is designed for general NLP tasks, while OLMo 13B has 13 billion parameters and offers increased reasoning capabilities. We used the instruct version of both models. Al2 is the only organization among these model developers that has disclosed the exact dataset used for pretraining. Both models are trained on Dolma, an open dataset that includes a mix of web content, academic publications, code, books, and encyclopedic materials. Multiple versions of this dataset are available. The OLMo models are designed with a focus on research accessibility, interpretability, and transparency.

### 4.2 Prompt-based learning

To enable the model to understand and play the game *Baba is You*, we first construct a prompt. This prompt consists of a short description of the game mechanics, followed by definitions of the characters to interpret the given level. Next, we explain the different rules that can be created in the game. After this a question is given to solve the given grid level with at the end a PS sentence to activate CoT.

We developed three distinct types of prompts: the *simple prompt*, which contains only the above description (see Figure 10); a *rule-extended prompt*, which additionally includes the active rules of the level (see Figure 11); and an *action-extended prompt*, which further expands upon the previous version by including an explicit description of the possible actions that can be performed in the game (see Figure 12). Following the approach by Nathan Cloos et al. [9], we adopt the sequence of actions they introduced in their prompt, modify them slightly, and add an additional action to improve the push mechanism of the game.

The construction of these prompts is manual, with the process being developed through a process of trial and error. Following the submission of a prompt to ChatGPT with version GPT-40, the result output was examined to identify areas for improvement. The prompt was then adjusted and re-submitted until a satisfactory result was obtained. Subsequently, the refined prompt was submitted to ChatGPT with GPT-40 version, requesting that it be further refined. This process ultimately led to the creation of the final prompt.

Model Input
You are helping to solve a gridworld game. In Baba is You, the player can change the game rules by moving text blocks around. The grid contains object text and property text.
Text Blocks: Object Text: Words representing game objects. Property Text: Words that describe actions or properties.
Active Rules: A rule is formed when the text blocks are aligned in a valid way, either horizontally or vertically. Valid rule formats include: <object.text> IS <property_text>: Grants a property to an object. <object1.text> IS <object1.text>: Changes one object into another. <object1.text> IS <object1.text>: Makes an object immutable. The goal is to use these rules to solve the level by moving the text blocks and controlling the objects.</object1.text></object1.text></object1.text></object1.text></property_text></object.text>
The level is displayed in a 2D grid, where each character has the following meaning: Text blocks in the game which always can be pushed: <pre>cobject.text&gt;:</pre> B BABA, S = SKULL, F = FLAG, O = FLOOR, A = GRASS, L = LAVA, R = ROCK, W = WALL, K = KEKE, G = GOOP, V = LOVE
<pre><pre><pre>cproperty.text&gt;: 1 = IS, 2 = YOU, 3 = WIN, 4 = KILL, 5 = PUSH, 6 = STOP, 7 = MOVE, 8 = HOT, 9 = MELT, 0 = SINK</pre></pre></pre>
Objects in the game: <pre><object skill,<br="">f = object skill, f = object flag, a = object grass, l = object rock, w = object wall, k = object kake, g = object goop, v = object love, . = border, . = empty space</object></pre>
<pre><object.text> IS YOU: Makes the object you control in the game. You can move it and push blocks. <object.text> IS WIN: The object you need to reach or to be to win the level. <object.text> IS STNE: Makes the object impassable (you can't move through it). <object.text> IS STNE: Makes the object melt when touched by something marked as HOT. <object.text> IS WELT: Makes the object destroy any object marked as MELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS STNE: Makes the object destroy any object marked as MELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS STNE: Makes the object destroy any object marked as HELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS NUE: Makes the object destroy any object marked as HELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS STNE: Makes the object of the object on the object of the object of the object of the object the object of the object of the object of the object of the object the object of the object of the object of the object of the object the object of the object of the object of the object the object destroy itself and anything it touches when it is touched.</object.text></object.text></object.text></object.text></object.text></object.text></object.text></object.text></object.text></pre>
Question: Give a solution to the following grid level:
\n_B12F13_\n\n\n\nbf\n \nn\n\n\n
Let's first understand the problem, extract the relevant objects, text blocks and rules (explain the rules) in the level and make a plan to solve the problem. Then let's carry out the plan by giving the intermediate actions (using common sense). Solve the problem step by step and show the solution.

Figure 10: **Simple prompt**: consisting only of a short game description and definitions of the characters and rules. Followed by a question to solve a level with at the end a sentence to activate zero-shot CoT.

# Model Input

You are helping to solve a gridworld game. In Baba is You, the player can change the game rules by moving text blocks around. The grid contains object text and property text.
Text Blocks: Object Text: Words representing game objects. Property Text: Words that describe actions or properties.
Active Rules: A rule is formed when the text blocks are aligned in a valid way, either horizontally or vertically. Valid rule formats include: <object.text> IS <object2.text>: Grants a property to an object <object1.text> IS <object2.text>: Makes an object immutable. The goal is to use these rules to solve the level by moving the text blocks and controlling the objects.</object2.text></object1.text></object2.text></object.text>
The level is displayed in a 2D grid, where each character has the following meaning: Text blocks in the game which always can be pushed: <pre>cobject.text&gt;:</pre> B = BABA, S = SKULL, F = FLAG, O = FLOOR, A = GRASS, L = LAVA, R = ROCK, W = WALL, K = KEKE, G = GOOP, V = LOVE
<pre>\$property.text&gt;: 1 = IS, 2 = YOU, 3 = WIN, 4 = KILL, 5 = PUSH, 6 = STOP, 7 = MOVE, 8 = HOT, 9 = MELT, 0 = SINK</pre>
Objects in the game: <pre><cbject shal,<br="">s = object shal, f = object shal, o = object floor, a = object grass, l = object ava, r = object rock, w = object rock, w = object tock, g = object goop, v = object love, = = border, . = empty space</cbject></pre>
<pre><object.text> IS YOU: Makes the object you control in the game. You can move it and push blocks. <object.text> IS WIN: The object you need to reach or to be to win the level. <object.text> IS STOP: Makes the object impassable (you can't move through it). <object.text> IS MLT: Makes the object melt when touched by something marked as HOT. <object.text> IS MLT: Makes the object destroy any object marked as MELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS MLT: Makes the object destroy any object marked as MELT when they touch it. If the same object is both HOT and MELT it self-destructs. <object.text> IS MLT: Makes the object destroy any object more through out then touched, but it stays intact. <object.text> IS VILL: Makes the object destroy anything you control when touched, but it stays intact. <object.text> IS PUSH: Lets you push the object destroy itself and anything it touches when it is touched.</object.text></object.text></object.text></object.text></object.text></object.text></object.text></object.text></object.text></pre>
Question: Give a solution to the following grid level:
\n_B12F13_\n\n\n\nbf\n \n\n\n\n
Current active rules: BABA IS YOU. Let's first understand the problem, extract the relevant objects, text blocks and rules (explain the rules) in the level and make a plan to solve the problem. Then let's carry out the plan by giving the intermediate actions (using common sense). Solve the problem step by step and show the solution.

Figure 11: **Rule-extended prompt**: In addition to the simple prompt the active rules of the level are added in the question part.

#### Model Input

You are helping to solve a gridworld game. In Baba is You, the player can change the game rules by moving text blocks around. The grid contains object text and property text. Text Blocks: Object Text: Words representing game objects. Property Text: Words that describe actions or properties. Active Bules A rule is formed when the text blocks are aligned in a valid way, either horizontally or vertically. Valid rule formats include: The goal is to use these rules to solve the level by moving the text blocks and controlling the objects. The level is displayed in a 2D grid, where each character has the following meaning: Text blocks in the game which always can be pushed: <object\_text>:
B = BABA, S = SKULL, F = FLAG, O = FLOORA = GRASS. L = LAVA, R = ROCK, W = WALL. K = KEKE, G = GOOP V = LOVE <property\_text>: 1 = IS, 2 = YOU, 3 = WIN, 4 = KILL, 5 = PUSH, 6 = STOP, 7 = MOVE, 8 = HOT, 9 = MELT, 0 = SINKObjects in the game: <object>: b = object baba, s = object skull, f = object flag, o = object floor, a = object grass, I = object lava. r = object rock w = object wall k = object keke g = object goop, v = object love, = border,
= empty space <object\_text> IS YOU: Makes the object you control in the game. You can move it and push blocks.
<object\_text> IS WIN: The object you need to reach or to be to win the level.
<object\_text> IS STDP: Makes the object massable (you can't move through it).
<object\_text> IS MELT: Makes the object melt when touched by something marked as HOT.
<object\_text> IS HOT: Makes the object destroy any object marked as MELT when they touch it. If the same object is both HOT and MELT it self-destructs.
<object\_text> IS MOVE: Makes the object move one step in its direction every turn.</or> cobject.text> IS KILL: Makes the object destroy anything you control when touched, but it stays intact. <object.text> IS PUSH: Lets you push the object or have it pushed by other moving objects. <object.text> IS SINK: Makes the object destroy itself and anything it touches when it is touched. A solution exists of one or more actions. An action can be one of the following: Move\_To[ ]: The player-controlled object moves to a valid target in the grid: <object>, <object.text>, or <property\_text>. This action is valid only if the specified target is present in the grid. Break\_Rule[ ] Removes an active rule unless it's at the border. Effects depend on the rule: [<object\_text> IS <object\_text>]: Nemoves the property from the object. [<object\_text> IS <object\_text>]: No changes occur. [<object\_text> IS <object\_text>]: Nakes the object mutable. Make\_Rule[ ]: Creates a new rule using valid text blocks in the grid, with effects depending on the rule: Cobject.text> IS operty.text>]: Grants a property to the object. [<object.text> IS <object2.text>]: Transforms <object1> the object. [<object1.text> IS <object1.text>]: Makes <object1> immutable. This action requires all components of the rule to be present in the grid. Push[]: The player-controlled object pushes another object or text block in the grid. Effects depend on what is pushed: Text blocks <object.text>, <property.text>: Can be pushed to arrange or create rules. Objects <object>: Can be pushed towards other objects if the pushed object is set to "push." This action is valid only if all elements are present in the grid and not placed at the border. Question: Give a solution to the following grid level: \_\_\_\_\_\n\_B12..F13\_\n\_....\_\n\_....\_\n\_....\_\n\_....\_\n\_.b....f.\_\n Current active rules: BABA IS YOU. Let's first understand the problem, extract the relevant objects, text blocks and rules (explain the rules) in the level and make a plan to solve the problem. Then let's carry out the plan by giving the intermediate actions (using common sense). Solve the problem step by step and show the solution.

Figure 12: Action-extended prompt: In addition to the rule-extend prompt an action format is given to help formulating the solution steps.

### 4.3 Evaluation of Reasoning

In order to investigate how well LLMs perform in reasoning and solving *Baba is You* levels, a manual analysis was performed to examine the reasoning chains generated by LLMs while solving *Baba is You* levels. The reasoning chain can be divided into four distinct sections: the interpretation of the level, the formulation of the problem statement, the formulation of the solution for the problem, and the formulation of the actions that should be taken for the solution. The first two sections are part of the analysis of the level, while the latter two sections are part of the solution process so formulating an answer consists of four steps. Table 2 summarizes errors encountered in these steps. If a step is correct, it is marked with a c label together with the number of the step, otherwise errors are classified according to the subcategories.

The process of interpreting the level involves the identification of the objects and text blocks present within the level, as well as the rule definition of all the currently active rules within the level. An incorrect interpretation of the level can result from the hallucination of the model, incorrect definition of terms, or lack of complete information. Hallucination refers to the identification of objects, text blocks, or rules that are not present on the level. An incorrect definition may include the incorrect classification of an object or text block (e.g., "F is an object FLAG") or the failure to define a rule. Incomplete information can be defined as the absence of defining objects, text blocks, or rules that are present in the level.

The formulation of the problem statement should include a clear definition of the objective of the level, together with any obstacles that may stand in the way of achieving the goal. An incorrect formulation of the problem statement could be the transfer of errors, incomplete information, incorrect assumptions or hallucinations of the model. The transfer of errors can be defined as the propagation of errors from a preceding step to the current step, by which the current step is also affected due to the presence of errors. Incomplete information refers to missing elements in the problem statement. Wrong assumptions include, for instance, the assumption that a flag object in the game, in the absence of a corresponding rule (FLAG IS WIN), constitutes a win condition.

The formulation of the solution incorporates a solution that is accurate and appropriate for the level in question. A correct solution comprises a description of the steps that can be performed in the level and which result in the level being solved. An incorrect formulation may manifest itself in one or more of the following ways: Transfer of mistakes, hallucinations, wrong reasoning or incomplete solution. The transfer of errors can be defined as the propagation of errors from the preceding steps to the current step. Hallucination is defined as the introduction of rules that cannot be derived from the text blocks present in the level, the breaking of rules that are not present at all, or actions with objects that are not present in the level. Incorrect reasoning can be defined as: misinterpreting the syntax of the rules, meaning struggling to correctly interpret the specific format and structure of the rule blocks. Failure to consider rule interactions, where the game operates on the principle of rule manipulation, where different rules interact in complex ways, a failure to fully understand how the interactions between rules lead to constraints or unintended outcomes. Overlooking constraints means that they generate strategies that do not take into account the specific constraints of the level. Failure to deal with ambiguity, in levels where the correct solution is not immediately obvious, leads to incorrect assumptions about the state of the game or overly simplistic strategies that ignore the complexity of the level. Incomplete solution means that the formulated solution misses components to fully solve the level.

The action formulation includes the actions the player must take within the level to successfully solve the problem. An incorrect formulation can result in error transfer, incomplete actions, inappropriate format, incorrect actions, or hallucination. Error transfer can be defined as the propagation of errors from the previous steps to the current step. Incomplete actions refer to the absence of a necessary step that should be included to solve the level. The term "wrong format" is only relevant to the third prompt, as the LLM should provide actions in a specific format. "Wrong actions" refer to the generation of actions that do not solve the problem when followed. Finally, we have hallucination, which means that new objects or rules are mentioned in the generated actions that were not mentioned before.

Category	Sub-Category	Description	
Level Interpre- tation (w1)	(1) Hallucination	When information is completely out of context and not present in the levels.	
	(2) Incorrect def- initions	Incorrect classification of an object or text block or incorrect definition of a rule.	
	(3) Incomplete Information	Absence of defining objects, text blocks, or rules that are present in the level.	
Formulating Problem State- ment (w2)	(1) Transfer of errors	Wrong problem statement derived from a previous incorrect statements .	
	(2) Incomplete information	Missing elements in the problem statement.	
	(3) Wrong as- sumptions	Assuming a condition or rule applies without explicit evidence.	
	(4) Hallucination	When information is completely out of context and not present in the levels	
Formulating Solutions (w3)	(1) Transfer of errors	Wrong solution derived from a previous incorrect statements	
	(2) Hallucination	When information is completely out of context and not present in the levels.	
	(3) Wrong rea- soning	Drawing incorrect conclusions due to misinterpreting rules, neglecting con- straints, or failing to account for rule interactions and ambiguities.	
	(4) Incomplete solution	Missing steps to fully solve the level.	
Formulating Actions (w4)	(1) Transfer of er- rors	Wrong actions derived from a previous incorrect statements.	
	(2) Incomplete actions	Missing actions to complete the level.	
	(3) Wrong for- mat	The actions are in the wrong format (only prompt 3).	
(4) Wrong ac- tions		Actions proposed are not solving the level.	
	(5) Hallucination	When actions are completely out of con- text.	

Table 2: Error Categorization.

# 4.4 Levels

To evaluate the LLMs, each model and prompt format was tested in 14 different levels, each of which tests a different aspect of the game, see Figure 13. Most of these levels are demo levels of the Keke AI competition [6], except level 14. The first level is the simplest, as it does not require any rule manipulation to solve. In this level, the player must move towards the flag to win. Levels 2-4 test the ability to create a rule, where three different aspects of the game are tested. Level 5 tests the understanding of the immutable rule. Level 6 tests the understanding of pushing objects. Level 7 deviates from the typical structure by not including the IS YOU rule at the beginning. Instead, it introduces the MOVE rule, which is activated by player input. Levels 8-10 focus on the ability to break rules with different properties. Level 11 tests understanding of the SINK rule, where there are stones that can be pushed into the goop to create a path to the flag. Level 12 is about understanding the HOT and MELT rules. This level can be solved in two ways: either by breaking the BABA IS MELT rule and going for the flag, or by breaking the BABA IS MELT rule and using the bricks to establish the BABA IS WIN rule. Levels 13 and 14 test the ability to create new rules, one involving understanding how to make another object the controllable object. The other is to create a new winning condition.



Figure 13: Levels used for the evaluation of the LLM models in playing Baba is You.

These levels require some logical thinking, but are relatively easy for humans. This is due to our natural ability to reason. For example, humans can intuitively understand that breaking the WALL IS STOP rule will allow movement through the wall. In addition, it is easier for humans to determine which rules can and cannot be broken by looking at the placement of the rules in the grid.

However, LLMs encounter a challenge when confronted with the task of solving these levels. These models must not only interpret the rules and mechanics of the game from the text, but also apply them in the environment. Unlike humans, LLMs have no inherent understanding of the world. These models rely entirely on the information provided to decide how to interact with and manipulate the game state. The levels are designed to assess specific components of the game, including rule creation, transformation, immutability and logical reasoning, which are needed to determine the model's ability to play the game *Baba is You*. While these puzzles may appear simple to humans, they serve as a benchmark for evaluating the reasoning and problem-solving abilities of different LLM models.

We tested these levels on the six different models. With the Mistral and OLMo models the response was always exactly the same when trying one prompt multiple times. However this was not the case with GPT-40 and Gemini Flash 1.5. GPT-40 exhibits variability in its responses when attempting to solve a level of *Baba is You* with the same prompt. When it provides an incorrect solution, repeated attempts often result in different formulated but still incorrect answers. However, when it generates a correct solution, subsequent responses tend to be consistent with only minor rewording or reformulations while preserving the same correct strategy. In contrast, Gemini Flash 1.5 showed inconsistency, sometimes giving the correct solution and other times an incorrect one for the same prompt. To evaluate accuracy, each prompt was tested five times, and a solution was considered correct if it appeared at least three times; otherwise, it was classified as incorrect.

For the OLMo and Mistral models, we used the ALICE high-performance computing facility to generate the output. For the other two models we used their own API.

### 4.5 Fine-tuning

For fine-tuning, we combined three different datasets into a single dataset. Each dataset contains a specific type of data. The largest dataset [2] consists of various questions designed to improve the model's reasoning ability. These questions cover a range of logical and analytical challenges. The second dataset [44] contains questions specifically related to the game mechanics of *Baba is You*. It includes questions about the interactions between different game elements, the effects of specific rule changes, and the general logic of the game (Table 4). The third dataset [44] is the smallest and consists of different levels of *Baba is You*. In this dataset, the input corresponds action-extended prompt of the level description, while the output represents the expected solution that the model should generate (Table 5). Together, these three data sets form a combined data set used for fine-tuning see Table 3 for the exact size of each dataset.

Dataset	Size
CoT-logic-reasoning	10500
Questions game mechanics	289
Levels & answers	15

Table 3: The size of the three datasets used for fine-tuning.

The dataset containing questions about the game mechanics of *Baba is You* was created through the following process. Initially, a set of questions was crafted, focusing on the rules

and mechanics of the game. Then, GPT-40 was prompted to generate additional unique questions based on the ones we had already created. These generated questions and answers were reviewed and, when necessary, corrected. This iterative process allowed us to quickly build a solid dataset of questions related to the game's mechanics.

The other dataset, which consists of levels and their solutions, is entirely handwritten. As a result, this dataset is smaller, as more time was spent on creating detailed solutions for each level rather than on increasing the dataset size.

We trained Mistral 7B and OLMo 7B on the combined dataset using LoRA for parameterefficient fine-tuning.

Dataset game mechanics		
Question	Answer	
Describe the action	The 'Make_Rule[ <object_text> IS <property_text>]' action creates</property_text></object_text>	
'Make_Rule[ <object_text></object_text>	a rule that assigns the specified property to the <object_text>.</object_text>	
<property_text>]'?</property_text>	This action can only be taken if <property_text>, 'IS', and</property_text>	
	<object_text> are all present in the grid level.</object_text>	
What happens if 'BABA	If 'BABA IS YOU' and 'SKULL IS KILL' are both active, and Baba	
IS YOU' and 'SKULL IS	(the player-controlled object) touches a Skull, Baba will be	
KILL' are active, and Baba	destroyed, causing the player to lose control of their character,	
touches a Skull?	and likely fail the level unless another 'YOU' object is present.	
How does the 'MOVE'	If an object has the 'MOVE' property set by the rule ' <object_text></object_text>	
property affect an object?	IS MOVE', it will move one step in its facing direction on each	
	turn, either blocking or potentially colliding with other objects	
	based on its movement path.	
What does 'BABA IS WIN'	If the rule 'BABA IS WIN' is active, then Baba (the object	
do?	associated with 'BABA') becomes the winning condition. The	
	player could win by controlling Baba (if 'BABA IS YOU' is active)	
	to instantly win the level or reach the Baba object with the	
	controlling object.	

Table 4: Examples of questions about the game mechanics of *Baba is You*.

#### 4.5.1 Parameter settings

For fine-tuning Mistral 7B and OLMo 7B we used the high performance computing facility ALICE. For fine-tuning we used one GPU node on the A100 MIG 4g.40GB GPU.

For Mistral 7B we use a learning rate of 2e-5, which is a common value for fine-tuning transformer models. The per\_device\_train\_batch\_size is set to 4 and the gradient\_accumulation\_step is set to 8, giving an effective batch size of 32. The num\_train\_epochs is set to 4, which means that the entire data set is processed 4 times during training. To enable mixed-precision training, where some computations use 16-bit float instead of 32-bit, we set fp16 to True; this setting reduces the amount of memory used on the GPU.

For OLMo 7B we use a learning rate of 3e-5, which is slightly higher than for Mistral 7B. We set the parameter auto\_find\_batch\_size to True which means that the batch size is automatically adjusted to the available GPU memory and gradient\_accumulation\_step is set to 6. The num\_train\_epochs is set to 7, which means that the entire dataset is processed 7 times during training. Again we used mixed precision training, so fp16 is set to True. We also set lr\_scheduler\_type to cosine, which means that it uses a cosine decay learning rate schedule that starts high and gradually decreases over time.

Dataset level & answer			
ASCII Representation	Active Rules	Level Description	Text Solution
ASCII Representation	Active Rules BABA IS YOU SKULL IS KILL	Level Description Current active rules are BABA IS YOU (B12) meanig we can control the baba object (b). SKULL IS KILL (S14) meaning skull objects (s) destroys all objects controlled by you when they touch them, while remaining intact themselves. The level has two skull objects and two baba objects. Also there are unused text blocks IS and WIN against the right border. Currently there is no win are divisor wet	<pre>'Text Solution Problem: Currently there is no win condition so we should make one with the available text blocks IS and WIN. Solution: - Break_Rule[SKULL IS KILL], push IS text block away Move_To[SKULL], move Baba toward the SKULL text block Push[SKULL], push SKULL text block to IS Make_Rule[SKULL IS WIN], set skull as win condition Move_To[s], move Baba to skull te win</pre>
 _vv12_ fF_ Rb.1_ _B123_ S.S14	LOVE IS YOU BABA IS YOU FLAG IS WIN SKULL IS KILL	<pre>condition yet. The current active rules are LOVE IS YOU (V12) meaning we can control and move around the love object (v). BABA IS YOU (B12) meaning we can can control and move around the baba object (b). FLAG IS WIN makes the flag object (f) the win condition anything controlled by you touches or is the object wins. SKULL IS KILL (S14) meaning skull objects (s) destroys all objects controlled by you when they touch them, while remaining intact themselves but are not present in the current grid. The level has one love object (1), one flag object (f), one baba object (b) and unused text blocks RUCK (R) and SKULL (S1.</pre>	to win. Problem: There are no obstacles in the way to solve this level. Solution: Move_To[f], move with baba or love object towards the flag to win the level.
 bssss 	BABA IS YOU FLAG IS WIN ROCK IS FLOOR SKULL IS KILL	The current active rules are BABA IS YOU (B12) meaning we can control and move around the baba object (b). FIAG IS WIN (F13) makes the flag object (f) the win condition anything controlled by you touches or is the object wins. SKULL IS KILL (S14) meaning skull objects (s) destroys all objects controlled by you when they touch them, while remaining intact themselves. ROCK IS FLOOR (R10) meaning all rock objects (r) transform in floor objects (o). The level has two baba objects (b), two flag objects (f), two rock objects (r), multipe skull objects (s) in a row and one floor object (o). The level has the following lose text tiles, BABA (B), GRASS (A), ROCK (R), WALL (W) and IS (1).	Problem: there are no obstacles in the way to solve this level. Solution: Move_To[f], move with a baba object towards one of the flags objects to win the level.

Table 5: Examples of the levels and answers in the level & answer dataset. The ASCII level will be inserted in the action-extended prompt together with active rules. The corresponding answers consist of the level description together with the text solution.

# 5 Results

In this chapter, we present the findings of our study, analyzing the outcomes based on predefined metrics.

### 5.1 Prompt-based learning

### 5.1.1 Simple prompt (1)

For the simple prompt, the biggest challenge for all models was their difficulty in correctly identifying the active rules. GPT-40 performed better because it had less difficulty identifying objects and text blocks in the grid (Figure 14, prompt 2). It was able to identify the correct active rule in the first step more often, but still struggled to formulate all the active rules especially the rules that are placed vertical. If the initial rules are not correctly identified, it becomes much more difficult to determine the correct problem statement and solution. Both the OLMo and Mistral models struggled to identify the correct objects and text blocks in the grid. This made it extremely difficult for these models to generate the correct rules, which was a critical step that they failed to bridge. As a result, the output for the simple prompt was largely incorrect due to misinterpretation of the rules, which is why the results for this prompt are not displayed.

#### 5.1.2 Rule-extended prompt (2) and Action-extended prompt (3)

**GPT-4**o and **Gemini Flash 1.5** Overall, GPT-4 performs relatively well when it comes to solving *Baba is You* levels, demonstrating a strong ability to understand the grid and formulate the correct problem statement. This is evident in the results presented in Figure 14, where GPT-4o has fewer errors in interpreting the grid compared to other models. Additionally, it shows reasonable performance in generating solutions for levels, as shown in Figure 15. This suggests that GPT-4o can be useful for providing solutions to specific levels, potentially assisting players in progressing through the game.

However, its ability to formulate actions directly on the grid itself appears to be less reliable. Often, the actions it suggests do not align with the solutions provided in the previous step, which can lead to inconsistencies when attempting to execute them.

One specific area where GPT-4 faces challenges is in identifying the rules that can be modified or broken within the game. In certain levels, such as levels 13 and 14, both of which feature obstacles blocking the path to the objective. In both levels the rule to remove the obstacle can not be broken. But GPT-40 gives in both levels as solution to break this rule and walk towards the flag. This highlights a particular difficulty for GPT-40 in determining which rules can and cannot be broken due to placement in the grid.

These observations suggest that while GPT-40 is quite capable in many aspects of solving *Baba is You* levels, there are certain areas where its performance could be improved, particularly when it comes to actions within the grid and understanding the placement constraints of rules.

For GPT-4o, the addition of an action format in the third prompt resulted in a small improvement over the rule-extended prompt. The model demonstrated an increased tendency to formulate correct solutions, showcasing a better understanding of the game's mechanics. However, despite this improvement, GPT-4o did not always strictly adhere to the specified action format.



Figure 14: Frequency of errors per step and subcategory in the reasoning chains generated by the models when solving the *Baba is You* levels. The findings reveal that GPT-40 has the least number of errors when formulating the reasoning chain for the levels. Furthermore, it is demonstrated that OLMo 7B generates the most errors in the reasoning output for the levels, primarily when defining the text blocks and objects in the grid. Finally, it is notable that all models encounter difficulties in formulating actions on the grid itself.

Gemini Flash 1.5 demonstrates a slightly worse performance compared to GPT-40 in some aspects, particularly in terms of identifying the correct objects and text blocks within a grid. This issue often results in a higher frequency of errors, which, in turn, affects the overall problem-solving process. In most cases, these errors were not a major obstacle when it came to formulating the problem statement. However, it is important to note that the problem statement was often incomplete, typically only mentioning the objective but neglecting to describe the obstacles that need to be overcome. On one occasion, the problem statement was entirely omitted.

Another noteworthy observation is the increase in hallucinations within the steps when utilizing the third prompt. The hallucination in the first step was not that big of a problem because it did not transfer into the other steps. But there was also more hallucination in formulating the solution.

Interestingly, level 12, when using prompt 2, produced an unconventional solution. In this case, the solution suggested breaking the rule "BABA IS MELT" and creating a new rule "LAVA IS MELT." This modification caused the lava to disappear, which was a novel approach to the puzzle. Furthermore, the output advised breaking this new rule and creating the rule "BABA IS WIN," which led to a successful solution. This solution is particularly intriguing



Figure 15: Correctness frequency per step in the reasoning chain generated by the LLM models. It is observed that GPT-40 has generated the most correct steps in the reasoning chain for the levels. Furthermore, it is noted that GPT-40 and Gemini Flash 1.5 benefit from the action-extended prompt, while the smaller models encounter difficulties irrespective of prompt structuring.

because it is not the most straightforward approach, demonstrating a creative way to solve the puzzle.

For action formulation Gemini Flash 1.5 shares similar challenges with GPT-40 when it comes to generating the appropriate actions needed to successfully complete a level.

A notable difference between the rule-extended and action-extended prompt is that Gemini 1.5 Flash showed difficulty in breaking rules when using the rule-extended prompt, rarely suggesting it. However, with the action-extended prompt, rule-breaking was suggested more frequently.

Both of these models have better overall performance when using the action-extended prompt with fewer errors and more steps formulated correctly; see Figures 16 and 17. In Table 6 are some examples of error snippets of the reasoning chain of GPT-40 and Gemini Flash 1.5.

**OLMo and Mistral** We also carried out experiments with different models from OLMo and Mistral to evaluate their performance on the fourteen levels of *Baba is You*. Table 7 shows examples of error snippets of the reasoning chain of these models. Compared to GPT-40 and Gemini Flash 1.5, these models exhibited a significantly higher number of errors, particularly in terms of correctly identifying objects and text blocks within the game grid (Figure 14).

Starting with OLMo-7B, this model struggled considerably in identifying the correct objects and text blocks. It frequently hallucinated during this step, which had a negative impact on the formulation of the problem statement. The inability to accurately identify key elements on the grid resulted in repeated errors and inconsistencies in the output. In addition to these issues, OLMo-7B was particularly prone to generating hallucinations throughout the process.

LLM Model	Error Snippets
GPT-40	Level 13: Goal: To win the level, BABA (controlled by the player) must reach the FLAG (f). However, the WALL blocks the direct path. Strategy: Modify the rules to bypass or remove the obstacle created by the WALL: Remove or break the "WALL IS STOP" rule. Create an accessible path for BABA to reach the FLAG.
	Level 4: Identify Barriers: BABA (controlled by the player) must reach the FLAG, but there is no direct path. ROCK (r) is blocking BABA's movement, and it has no properties assigned. Use Rules to Clear the Path: Assign ROCK IS PUSH by moving the text blocks R, 1, and 5 (if present) to form the rule. This will allow BABA to push the ROCK (r) out of the way. Reach the FLAG: With the ROCK out of the way, BABA can move directly to the FLAG.
Gemini 1.5 Flash	Level 4:**Objects:** We have Baba (b), a Rock (r), and Flags (f). **Text Blocks:** BABA (B), IS (1), YOU (2), FLAG (F), and WIN (3). **Current Rules:** BABA IS YOU (you control Baba) and FLAG IS WIN (reaching a flag wins the level). **Plan** Currently, the "FLAG IS WIN" rule is broken because the IS (1) and WIN (3) are not next to the FLAG (F). We need to move the IS and WIN next to the FLAG to make the rule active again.
	Level 5: Plan The goal is to reach the Flag. However, the initial rules make all Rocks into Flags, which can be confusing and potentially block the path. We need to change the rules to make the Flag a win condition again and allow Baba to move freely.
	Level 13:**Solution** 1. **Breake_Rule[GOOP IS SINK]:** Remove the rule "GOOP IS SINK." This will prevent the Goop from destroying itself. 2. **Make_Rule[ROCK IS HOT]:** Create the rule "ROCK IS HOT." This will make the Rocks hot. 3. **Make_Rule[GOOP IS MELT]:** Create the rule "GOOP IS MELT." This will make the Goop melt when it touches the hot Rocks.

Table 6: Examples of error snippets in the reasoning chains of GPT-40 and Gemini 1.5 Flash LLM models, where the red-colored text indicates incorrect reasoning. These errors highlight challenges such as hallucinations, incorrect assumptions, and overlooking key constraints.

OLMo-13B, while showing a slight improvement in object and text block identification, still encountered significant challenges. Although the performance of this model was marginally better than that of OLMo-7B, errors remained frequent and detrimental to the overall problem-solving process. Notably, both OLMo-7B and OLMo-13B exhibited difficulties in understanding the distinction between objects and text blocks. This misunderstanding often led them to treat the "WIN" text block as the primary goal, rather than the intended objective. As a result, the solutions generated by these models often appeared confused or misdirected.

Another notable issue was that both OLMo models failed to follow the instructions provided in the action-extended prompt, specifically the requirement to formulate the actions in the correct format.

In general, the solutions generated by both OLMo-7B and OLMo-13B gave the impression that these models did not fully grasp the mechanics of the game. It was often out of context concerning the game mechanics. Their inability to accurately identify and interpret key elements of the puzzle, combined with frequent hallucinations and wrong reasoning, resulted in suboptimal

LLM Model	Error Snippets
Mistral 8x7B	Level 1: Step 3 Devise a plan - We need to move the BABA object to the FLAG object. - To do this, we need to change the rule BABA IS YOU to BABA IS MOVE. - Then, we need to push the ROCK object to the WALL object to create a path for the BABA object to reach the FLAG object.
	Level 3: Step 1: Understand the problem and extract the relevant objects, text blocks, and rules. Objects: BABA, FLOOR, GRASS, WALL, SKULL Text Blocks: 1, 2, 3 Rules: BABA IS YOU Step 2: Devise a plan to solve the problem. 1. Make BABA IS YOU to BABA IS MOVE. 2. Move BABA to the position of the SKULL. 3. Make SKULL IS YOU.
	Level 4: Step 2: Extract the relevant objects, text blocks and rules - Objects: BABA, SKULL, FLAG, FLOOR, GRASS, ROCK, WALL, KEKE, GOOP, LOVE, and empty space. - Text blocks: BABA, IS, YOU, FLAG, IS, WIN. - Rules: BABA IS YOU, FLAG IS WIN. Step 3: Devise a plan - Make BABA the object you control. - Make FLAG the object you need to reach to win the level.
Mistral 7B	Level 1: The goal is to move the BABA to the FLAG to win the level. Plan: 1. Move the BABA to the flag by pushing the blocks in the way. 2. Make the BABA move towards the flag. 3. Make the BABA push the flag towards itself.
	Level 3: Plan: 1. Move BABA to the 1 text block. 2. Change BABA into the flag (f) using the 1 text block. 3. Move the flag to the exit (the empty space next to the 3 text block).
	Level 9: The goal is to move the flag to the exit (position B12) without getting killed by the skull.
OLMo 13B	Level 3: **Objective:** Reach the WIN block to win the level.
	Level 4:**Plan:** 1. **Make Baba movable:** Ensure Baba has the "MOVE" property.
	<pre>Level 9: 1. **Neutralize SKULL:** - We cannot directly push SKULL because it will kill BABA We need to use text blocks to change SKULL's properties **Make Rule:** SKULL IS STOP. This will make SKULL impassable and safe for BABA.</pre>
OLMo 7B	Level 1: ### Plan 1. **Identify the Goal:** Move BABA to the FLAG (F13). 2. **Current Obstacles:** BABA starts at B12, and there are walls (W) and empty spaces (.) blocking the path.
	Level 2: 2. **Initial Setup:** - BABA is currently next to a WALL (W) and a KEKE (K). - There's a FLAG (F) adjacent to BABA.
	Level 9: ### Plan: 1. **Move BABA away from SKULL:** To avoid immediate death, move BABA away from SKULL. 2. **Create a path to FLAG:** Use the environment to create a path to FLAG. 3. **Utilize GOOP:** GOOP can be used to block SKULL or create a bridge

Table 7: Examples of error snippets in the reasoning chains of OLMo and Mistral LLM models, where the red-colored text indicates incorrect reasoning. These errors highlight challenges such as hallucinations, incorrect assumptions, and wrong reasoning steps.



Figure 16: Correct steps per model across the 14 *Baba is You* levels with the ruleextended prompt. The rule-extended prompt, which provides the active rules present in the level, improves performance across models but still highlights major differences in reasoning capabilities. GPT-40 outperforms other models, demonstrating stronger multistep problem-solving skills. While Gemini 1.5 Flash show partial success, its performance remains inconsistent. The results suggest that simply providing active rules helps but does not bridge the gap in logical reasoning ability between smaller models and more advanced LLMs like GPT-40.

performance. Overall, while these models exhibited some capability, they were far less effective in solving the *Baba is You* puzzles compared to GPT-40 and Gemini Flash 1.5 (Figure 16 and 17).

The Mistral models demonstrate slightly better performance compared to the OLMo models, particularly in defining objects and text blocks within a grid see Figure 14. However, their overall ability to provide well-structured and effective solutions remains limited. While they exhibit some improvements over OLMo models in certain areas, they still struggle with generating high-quality solutions.

A notable observation regarding the Mixtral 8x7B model is its approach to problem-solving when given the action-extended prompt. This model consistently omitted descriptions of the text blocks and rules, as well as the formulation of the problem statement. Instead of outlining the problem before attempting to solve it, Mixtral 8x7B directly proceeded to a solution formulation. This often led to incomplete or suboptimal solutions, as the model did not establish a clear understanding of the problem space before attempting to resolve it.

Another recurring issue observed in Mixtral 8x7B's solutions is its handling of rule-breaking as a mechanic. The model does not seem to recognize that breaking a rule can be a valid action to remove it. Instead, it tends to suggest creating an alternative rule rather than utilizing a rule-breaking move when explicitly provided as an option. This suggests a fundamental misunderstanding of the flexibility of the rule manipulation within the game.

Additionally, a repeated pattern in Mixtral 8x7B's responses is its disregard for active rules. The model frequently claims that certain rules do not yet exist and must still be created, even when they are already active in the level. This indicates a potential issue with tracking the



Figure 17: Correct steps per model across the 14 *Baba is You* levels with the actionextended prompt. The action-extended prompt, which provides additional details about possible actions, leads to notable improvements for some models, particularly Gemini 1.5 Flash and GPT-40. However, GPT-40 remains the strongest performer, consistently solving more steps across all levels. While some smaller models show slight improvements, their overall performance remains limited, suggesting that improving prompts alone is not sufficient to overcome their reasoning limitations. These results highlight the importance of both prompt design and underlying model capability in tackling complex rule-based reasoning tasks.

current state of the rules within the problem environment.

A notable aspect of Mistral 7B is that it often interprets the rules as a single text block rather than distinct elements. It also frequently suggests moving to the rules themselves to win the level, indicating that it does not fully grasp the separation between objects and rules as independent entities.

Both Mistral 7B and Mixtral 8x7B perform well in following the instruction to format action formulations in a specific manner. However, despite their ability to adhere to the required format, the actions they propose often lack effectiveness or logical coherence. A notable thing for level 13 was that both models suggested creating the rule "KEKE IS BABA," which, in theory, could be a valid approach. However, in the given context, there is no available "IS" block that could be used to construct this rule without first breaking the "IS YOU" rule. This suggests that the models do not fully account for the constraints of the environment when generating solutions.

Another notable flaw in their reasoning process is the way they approach movement mechanics. In some of their solutions, both models incorrectly assume that "BABA IS MOVE" must be established before Baba can be moved. This misunderstanding reflects an incomplete grasp of the mechanics in the game.

With the action-extended prompt, the Mistral models more frequently suggested breaking or forming rules, while in the rule-extended prompt, it rarely mentioned breaking rules mostly suggesting to form an other rule. The action-extended prompt also led to solutions that were more readable and better aligned with the game mechanics, though still mostly incorrect. In contrast, the rule-extended prompt frequently mentioned "use" this rule to let something happen but using a rule is not really an action that can be done in the grid.



Figure 18: Frequency of errors per step and subcategory in the reasoning chains generated by the models when solving the *Baba is You* levels. The findings reveal that both fine-tuned models have less errors in the analyzing part of the level in the reasoning chain after fine-tuning compared to the original model.

Overall, while the Mistral models show some improvements over OLMo models in terms of object and text block definition, they still exhibit significant weaknesses in solution formulation. Their inability to fully grasp rule-breaking mechanics, track active rules, and account for problem constraints limits their effectiveness in generating optimal solutions.

With these models, the action-extended prompt has no clear improvement on the models performance. So, it is different for each model how they respond on the inclusion of explicit rules and structured action formats in prompts. While GPT-40 and Gemini 1.5 Flash benefit from additional structure, OLMo and Mistral do not clearly benefit from these extra instructions.

Across all models, levels 4 and 5 proved particularly challenging when using the rule-extended prompt (Figure 16). Both levels include the active rule FLAG IS WIN, which led most models to incorrectly assume that a flag object must already exist in the grid. However, in these levels, the correct solution involves creating or breaking a rule to introduce the flag object, a step that none of the models consistently recognized.

This pattern suggests that models struggle to distinguish between active rules and their actual objects in the game. They often interpret an active rule as implying the immediate

presence of the associated object, rather than understanding that additional steps may be required.

Another common misinterpretation observed was in level 5, this level contained the rules FLAG IS ROCK and ROCK IS ROCK. Some models incorrectly concluded that the flag was inside the rock itself, leading them to believe that touching the rock would satisfy the FLAG IS WIN condition. This suggests a difficulty in distinguishing between identity transformations (X IS Y) and persistent object properties, leading to incorrect assumptions about win conditions. These errors highlight the limitations of current LLMs in handling dynamic rule interactions and game logic.



Figure 19: Correctness frequency per step in the reasoning chain generated by the LLM models. It is observed that both fine-tuned models have more correct steps after fine-tuning. For Mistral 7B there is an improvement in classification of objects and text blocks in the grid. For OLMo 7B there is an improvement in the problem statement formulation.

## 5.2 Fine-tuning

### 5.2.1 Mistral 7B

After fine-tuning Mistral 7B on the combined dataset there is an improvement on classification of the objects and text blocks in the grid see Figure 19. There was less misclassification and incomplete information during this step (Figure 18) and more often is was formulated correctly. However, unfortunately this did not lead to a clear improvement of formulating the problem statement and solution of a level compared to the original model. The formulation of the problem statement was often incomplete or there were wrong assumptions made about objects in the grid. These error often transferred into the solution formulating step which resulted that most of the time the solution was also incorrect. The biggest improvement is on level 1 where it formulated the first two steps correctly see Figure 20. However, there were also some levels where the original model was closer to the correct answer than the fine-tuned model.

#### 5.2.2 OLMo 7B

After fine-tuning OLMo 7B on the created dataset, we observe an improvement in the formulation of the problem statement (Figure 19) and a reduction in classification errors for objects and text blocks (Figure 18). However, the model still struggles with correctly distinguishing between them. Furthermore, it continues to face challenges in the solution process, often failing to differentiate objects from text blocks and occasionally misidentifying the "WIN" text block as the primary target. The generated solutions suggest that the fine-tuned model still has difficulty grasping the game mechanics, rarely proposing actions such as breaking or creating rules. Additionally, it sometimes treats text blocks as the objects you control. Thus, fine-tuning did not yield significant improvements in solution formulation.

Fine-tuning the models with textual data from the game *Baba is You* led to improvements in level analysis for both models. In the case of Mistral 7B, there was an improvement in classifying text blocks and objects, while for OLMo 7B, the problem statement formulation showed better results. However, for both models, there was no clear improvement in the solving process, as the generated solutions and actions still contained many errors.



Figure 20: Correct steps per model across the 14 Baba is You levels with the actionextended prompt. It is observed that the fine-tuned models have more correct steps across the levels but it is still not enough to fully solve the levels.

# 6 Discussion & Conclusion

We have seen multiple LLMs try to solve different levels of *Baba is You*. We will now answer the proposed research questions.

**RQ1:** How does the choice of pre-trained LLMs affect the performance on levelsolving tasks in *Baba is You*? .

The choice of pre-trained LLM affects the performance in solving *Baba is You* levels. GPT-40 and Gemini Flash 1.5 have better performance in object identification, problem formulation and solution. However, they still show inconsistencies in formulating actions in the grid. They also struggle with recognising which rules can be broken or modified.

Mistral models (7B and Mixtral 8x7B) perform slightly better than OLMo models in object recognition but fail to generate coherent solutions due to difficulties in tracking active rules and understanding rule-breaking mechanics. OLMo models (7B and 13B) perform the worst, frequently hallucinating objects and failing to grasp fundamental game mechanics.

Overall, larger, more advanced models like GPT-40 and Gemini 1.5-Flash benefit from structured prompts, whereas smaller models (Mistral and OLMo) struggle regardless of prompt structuring.

**RQ2:** To what extent can an LLM create new rule manipulations and apply them in the game *Baba is You* to solve the puzzles?

LLMs show varying abilities in creating and applying rule manipulations. GPT-4o and Gemini 1.5-Flash occasionally generate novel rule-based solutions, such as modifying "BABA IS MELT" to "LAVA IS MELT" to remove obstacles. However, they sometimes suggest breaking unbreakable rules or fail to recognize constraints imposed by the game grid.

Mistral models struggle with rule-breaking as a valid mechanic, often opting to create new rules rather than removing existing ones. OLMo models show little understanding of rule manipulation, frequently misinterpreting game mechanics.

While some LLMs can generate (creative) rule manipulations at certain levels, their effectiveness is inconsistent.

**RQ3:** How does fine-tuning with textual and structural data from *Baba is You* affect the performance of an LLM in learning to play the game compared to models without such fine-tuning?

Fine-tuning with textual and structural data from *Baba is You* led to improvements in specific areas for both Mistral 7B and OLMo 7B, but did not result in a clear overall enhancement in solving the game. Mistral 7B showed improvements in classifying objects and text blocks, reducing misclassifications and incomplete information. However, this did not translate into a significant improvement in problem statement formulation or solution generation, with errors in assumptions persisting and negatively impacting the solution process.

OLMo 7B, on the other hand, showed improvement in problem statement formulation and a reduction in classification errors, yet it continued to struggle with distinguishing between objects and text blocks. The model sometimes misidentified key game elements, such as treating the "WIN" text block as a primary target or failing to propose valid actions like breaking or creating

rules. As a result, its ability to generate correct solutions remained limited.

Overall, fine-tuning with *Baba is You* data enhanced level analysis but did not significantly improve the models' ability to solve game levels. Both models still exhibited fundamental misunderstandings of game mechanics, suggesting that fine-tuning with the used data was not enough to improve the reasoning of these models.

To answer the question of whether an LLM is able to understand and play the game Baba is You, we can say that there is still room for improvement. Although GPT-40 comes quite far in its chain of reasoning to solve the levels, we have to give the active rules and an action format in the prompt, otherwise it is much less effective in giving correct answers. Also, all LLMs performed very poorly in formulating the action to be taken directly on the grid. This suggests that they do not fully understand how to interpret the grid as a two-dimensional space. It also explains why GPT-40 has difficulty recognising which rules can be broken and which cannot. It therefore gives the most straight forward solution without taking these constraints into account. Also, sometimes in a chain of reasoning it is questionable whether the models really understand the mechanism of braking and creating a rule, that you create or break a rule by pushing text blocks with the object you are controlling. So we can conclude that high-end models such as GPT-40 and Gemini 1.5 Flash give promising results in solving Baba is You levels, but they still sometimes lack the ability to fully interpret the two-dimensional space. This leads to incorrect solutions. For the smaller models OLMo and Mistral, with or without fine-tuning, it is currently too difficult to understand and play the game. They are not able to fully interpret the levels and also give the impression that they do not fully understand the game mechanics. So whether an LLM is able to understand and play the game Baba is You: there are promising results with larger models, but there is still a lot of room for improvement.

### 6.1 Limitations & Further work

This research used a PS [41] sentence which is a form of zero-shot Chain-of-Thought (CoT) prompting approach, but there are numerous ways to construct prompts that can affect model performance. Alternative prompting techniques, such as few-shot CoT [43], step-by-step prompting or self-ask [30] could potentially yield different results and should be explored in further work. In addition, the dataset used for fine-tuning was relatively small, particularly in terms of training data for solution of levels. Only 15 examples were included in the mixed dataset for this research. In addition, the game mechanics questions dataset contained 298 questions, which, while useful, may not be sufficient to capture the full complexity of the game's logic. Expanding the data set with a larger and more diverse collection of *Baba is You* content, such as more solution examples, varied puzzle structures, and broader mechanics-related questions, could improve the generalisation and performance of the model across different levels.

The evaluation was conducted on only 14 different levels, which is informative but a limited sample size. These levels were relatively easy for human players, meaning that the results may not fully capture the model's performance on more complex puzzles. Expanding the evaluation set to include a wider range of difficulty levels could provide deeper insights into the LLM's ability to solve *Baba is You* puzzles. Furthermore, while the fine-tuning was performed using the available dataset, it is possible that a more optimal configuration of hyperparameters could improve performance. Exploring different fine-tuning strategies, parameter settings (such as varying the learning rate, batch size, or number of training epochs), and larger training datasets could lead to improved model reasoning and problem-solving abilities.

The methodology in this research relied on classifying errors and correct steps at different stages of the reasoning process and analyzing their frequency. Although this approach provides quantitative information on model performance, error classification was performed manually, making it vulnerable to subjective bias. In addition, human evaluators can interpret mistakes differently. To improve objectivity, future work could establish automated techniques to aid in error classification.

Further work could also investigate the performance of other language models, as new versions are released frequently. Testing different models could provide a broader perspective on how architectural improvements affect puzzle solving performance. Another possible direction is to experiment with different prompt inputs. It is also possible to explore the use of images instead of text-based level descriptions. This could provide an alternative way for models to process game states. Comparing text-based and image-based input may provide further insights into how models interpret and solve these types of puzzles.

# Bibliography

- Meta Fundamental AI Research Diplomacy Team (FAIR)<sup>†</sup> et al. "Human-level play in the game of Diplomacy by combining language models with strategic reasoning". In: Science 378.6624 (2022), pp. 1067–1074.
- Isaiah Bjorklund. cot-logic-reasoning. 2025. URL: https://huggingface.co/ datasets/isaiahbjork/cot-logic-reasoning.
- [3] Noam Brown and Tuomas Sandholm. "Superhuman AI for heads-up no-limit poker: Libratus beats top professionals". en. In: *Science* 359.6374 (Jan. 2018), pp. 418–424. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.aao1733. URL: https://www.science.org/doi/10.1126/science.aao1733 (visited on 01/28/2025).
- [4] Tom B Brown. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).
- [5] Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. "Deep blue". In: Artificial intelligence 134.1-2 (2002), pp. 57–83.
- [6] M Charity and Julian Togelius. "Keke AI Competition: Solving puzzle levels in a dynamically changing mechanic space". In: 2022 IEEE Conference on Games (CoG). ISSN: 2325-4289. Aug. 2022, pp. 570-575. DOI: 10.1109/CoG51982.2022.9893650. URL: https://ieeexplore.ieee.org/document/9893650/?arnumber=9893650.
- Megan Charity, Ahmed Khalifa, and Julian Togelius. "Baba is Y'all: Collaborative Mixed-Initiative Level Design". In: 2020 IEEE Conference on Games (CoG). 2020, pp. 542–549. DOI: 10.1109/CoG47356.2020.9231807.
- [8] Shreyas Chaudhari et al. RLHF Deciphered: A Critical Analysis of Reinforcement Learning from Human Feedback for LLMs. en. arXiv:2404.08555 [cs]. Apr. 2024. DOI: 10.48550/arXiv.2404.08555. URL: http://arxiv.org/abs/2404.08555 (visited on 02/03/2025).
- [9] Nathan Cloos et al. "Baba is ai: Break the rules to beat the benchmark". In: *arXiv* preprint arXiv:2407.13729 (2024).
- [10] Xidong Feng et al. "Chessgpt: Bridging policy learning and language modeling". In: Advances in Neural Information Processing Systems 36 (2024).
- [11] Zeyu Han et al. Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey. en. arXiv:2403.14608 [cs]. Sept. 2024. URL: http://arxiv.org/abs/2403. 14608 (visited on 11/12/2024).
- [12] Edward J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. en. arXiv:2106.09685 [cs]. Oct. 2021. URL: http://arxiv.org/abs/2106.09685 (visited on 11/06/2024).

- [13] Aaron Hurst et al. "Gpt-4o system card". In: arXiv preprint arXiv:2410.21276 (2024).
- [14] Cheonsu Jeong. "Fine-tuning and utilization methods of domain-specific llms". In: arXiv preprint arXiv:2401.02981 (2024).
- [15] Albert Q. Jiang et al. Mistral 7B. en. arXiv:2310.06825 [cs]. Oct. 2023. DOI: 10.
   48550/arXiv.2310.06825. URL: http://arxiv.org/abs/2310.06825 (visited on 02/03/2025).
- [16] Albert Q. Jiang et al. Mixtral of Experts. en. arXiv:2401.04088 [cs]. Jan. 2024. DOI: 10.48550/arXiv.2401.04088. URL: http://arxiv.org/abs/2401.04088 (visited on 02/03/2025).
- [17] Uday Kamath et al. "Prompt-based Learning". In: Large Language Models: A Deep Dive: Bridging Theory and Practice. Cham: Springer Nature Switzerland, 2024, pp. 83–133. ISBN: 978-3-031-65647-7. DOI: 10.1007/978-3-031-65647-7\_3. URL: https://doi.org/10.1007/978-3-031-65647-7\_3.
- [18] Liwei Kang et al. "On the Empirical Complexity of Reasoning and Planning in LLMs". In: Findings of the Association for Computational Linguistics: EMNLP 2024. Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 2897–2936. DOI: 10.18653/v1/2024.findings-emnlp.164. URL: https://aclanthology.org/ 2024.findings-emnlp.164/.
- [19] Takeshi Kojima et al. "Large language models are zero-shot reasoners". In: Advances in neural information processing systems 35 (2022), pp. 22199–22213.
- [20] Yinghao Li, Haorui Wang, and Chao Zhang. "Assessing Logical Puzzle Solving in Large Language Models: Insights from a Minesweeper Case Study". en. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Mexico City, Mexico: Association for Computational Linguistics, 2024, pp. 59–81. DOI: 10.18653/v1/2024.naacl-long.4. URL: https://aclanthology.org/2024. naacl-long.4 (visited on 01/07/2025).
- [21] Suvir Mirchandani et al. Large Language Models as General Pattern Machines.
   en. arXiv:2307.04721 [cs]. Oct. 2023. DOI: 10.48550/arXiv.2307.04721. URL: http://arxiv.org/abs/2307.04721 (visited on 01/30/2025).
- [22] Mistral\_7B\_instruct-baba. 2025. URL: https://huggingface.co/Migthytwig/ Mistral\_7B\_instruct-baba.
- [23] Humza Naveed et al. A Comprehensive Overview of Large Language Models. en. arXiv:2307.06435 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2307.06435. URL: http: //arxiv.org/abs/2307.06435 (visited on 02/03/2025).
- [24] David A. Noever and Ryerson Burdick. Puzzle Solving without Search or Human Knowledge: An Unnatural Language Approach. 2021. URL: https://api. semanticscholar.org/CorpusID:237431487.
- [25] Maxwell Nye et al. "Show your work: Scratchpads for intermediate computation with language models". In: *arXiv preprint arXiv:2112.00114* (2021).

- [26] Team OLMo et al. 2 OLMo 2 Furious. en. arXiv:2501.00656 [cs]. Jan. 2025. DOI: 10.48550/arXiv.2501.00656. URL: http://arxiv.org/abs/2501.00656 (visited on 02/03/2025).
- [27] *olmo\_7B\_instruct-baba.* 2025. URL: https://huggingface.co/Migthytwig/olmo\_7B\_instruct-baba.
- [28] OpenAI et al. Dota 2 with Large Scale Deep Reinforcement Learning. en. arXiv:1912.06680
   [cs]. Dec. 2019. DOI: 10.48550/arXiv.1912.06680. URL: http://arxiv.org/abs/ 1912.06680 (visited on 01/28/2025).
- [29] Ofir Press et al. "Measuring and narrowing the compositionality gap in language models". In: *arXiv preprint arXiv:2210.03350* (2022).
- [30] Ofir Press et al. Measuring and Narrowing the Compositionality Gap in Language Models. en. arXiv:2210.03350 [cs]. Oct. 2023. URL: http://arxiv.org/abs/2210.
   03350 (visited on 09/19/2024).
- [31] Alec Radford et al. Improving Language Understanding by Generative Pre-Training. en. 2018.
- [32] Jonathan Schaeffer et al. "Checkers is solved". In: science 317.5844 (2007), pp. 1518– 1522.
- [33] Julian Schrittwieser et al. "Mastering atari, go, chess and shogi by planning with a learned model". In: *Nature* 588.7839 (2020), pp. 604–609.
- [34] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.
- [35] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144.
- [36] Gemini Team et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. en. arXiv:2403.05530 [cs]. Dec. 2024. DOI: 10.48550/arXiv. 2403.05530. URL: http://arxiv.org/abs/2403.05530 (visited on 02/03/2025).
- [37] Arvi Teikari. "Baba is you". In: Game [PC]. (March 2019). Hempuli Oy, Finland (2019).
- [38] Oguzhan Topsakal, Colby Jacob Edell, and Jackson Bailey Harper. "Evaluating large language models with grid-based game competitions: an extensible LLM benchmark and leaderboard". In: *arXiv preprint arXiv:2407.07796* (2024).
- [39] Karthik Valmeekam et al. Planning in Strawberry Fields: Evaluating and Improving the Planning and Scheduling Capabilities of LRM o1. en. arXiv:2410.02162 [cs]. Oct. 2024. DOI: 10.48550/arXiv.2410.02162. URL: http://arxiv.org/abs/2410. 02162 (visited on 03/21/2025).
- [40] Ashish Vaswani et al. Attention Is All You Need. en. arXiv:1706.03762 [cs]. Aug. 2023.
   DOI: 10.48550/arXiv.1706.03762. URL: http://arxiv.org/abs/1706.03762
   (visited on 02/03/2025).
- [41] Lei Wang et al. Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models. en. arXiv:2305.04091 [cs]. May 2023. URL: http://arxiv.org/abs/2305.04091 (visited on 10/15/2024).

- [42] Lei Wang et al. "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models". In: *arXiv preprint arXiv:2305.04091* (2023).
- [43] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: Advances in neural information processing systems 35 (2022), pp. 24824– 24837.
- [44] F.T. van Wetten. LLM\_babaisyou. https://github.com/fien99/LLM\_babaisyou. 2025.
- [45] Lingling Xu et al. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. en. arXiv:2312.12148 [cs]. Dec. 2023. DOI: 10.48550/arXiv.2312.12148. URL: http://arxiv.org/abs/2312.12148 (visited on 02/03/2025).
- [46] Yuzhuang Xu et al. Exploring Large Language Models for Communication Games: An Empirical Study on Werewolf. en. arXiv:2309.04658 [cs]. May 2024. DOI: 10. 48550/arXiv.2309.04658. URL: http://arxiv.org/abs/2309.04658 (visited on 01/07/2025).
- [47] Dharunish Yugeswardeenoo, Kevin Zhu, and Sean O'Brien. "Question-Analysis Prompting Improves LLM Performance in Reasoning Tasks". In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop). Ed. by Xiyan Fu and Eve Fleisig. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 402–413. DOI: 10.18653/ v1/2024.acl-srw.45. URL: https://aclanthology.org/2024.acl-srw.45/.
- Shengyu Zhang et al. Instruction Tuning for Large Language Models: A Survey. en. arXiv:2308.10792 [cs]. Mar. 2024. URL: http://arxiv.org/abs/2308.10792 (visited on 04/02/2024).