RISCORE: Enhancing In-Context Riddle Solving in Language Models through Context-Reconstructed Example Augmentation

Ioannis Panagiotopoulos, Giorgos Filandrianos, Maria Lymperaiou, Giorgos Stamou

School of Electrical and Computer Engineering, AILS Laboratory

National Technical University of Athens

yiannispn@gmail.com, {geofila, marialymp}@islab.ntua.gr, gstam@cs.ntua.gr

Abstract

Riddle-solving requires advanced reasoning skills, pushing Large Language Models (LLMs) to engage in abstract thinking and creative problem-solving, often revealing limitations in their cognitive abilities. In this paper, we examine the riddle-solving capabilities of LLMs using a multiple-choice format, exploring how different prompting techniques impact performance on riddles that demand diverse reasoning skills. To enhance results, we introduce RISCORE (**RI**ddle **S**olving with **CO**ntext **RE**contruciton), a novel fully automated prompting method that generates and utilizes contextually reconstructed sentence-based puzzles in conjunction with the original examples to create few-shot exemplars. Our experiments demonstrate that RISCORE significantly improves the performance of language models in both vertical and lateral thinking tasks, surpassing traditional exemplar selection strategies across a variety of few-shot settings [1].

1 Introduction

Reasoning in Natural Language Processing has been a field of increasing interest, especially since the surge of Large Language Models (LLMs), which showcase several reasoning shortcomings (Yu et al., 2024; Qiao et al., 2023). The logical gap between humans and LLMs' advanced reasoning is further exposed when probing the puzzle solving capabilities of such models (Giadikiaroglou et al., 2024), denoting the necessity of intricate datasets (Lin et al., 2021; Szomiu and Groza, 2021; Zhang and Wan, 2022) that test and enhance complex model reasoning. In terms of advanced reasoning skills, lateral thinking datasets such as BrainTeaser (Jiang et al., 2023b) stress the creative reasoning capabilities of models, requiring them to defy obvious logical associations and approach a more abstract way of thinking.

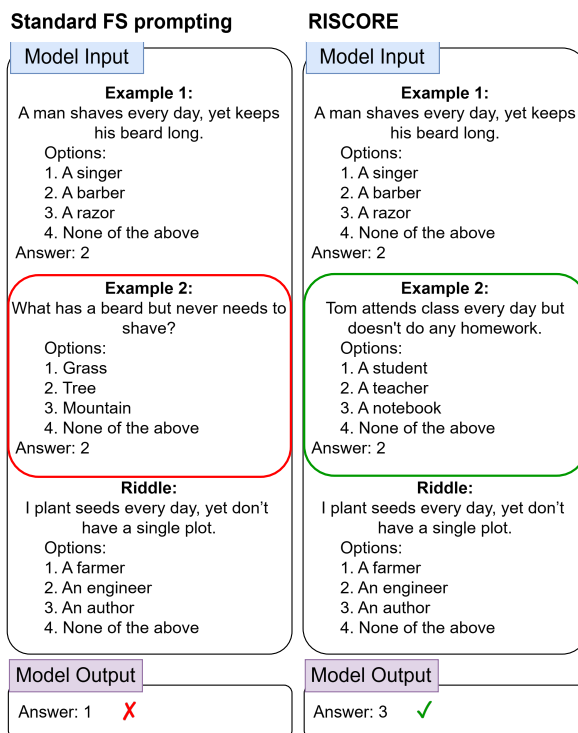[1]Code for our experiments is available at: https://github.com/GiannisPana/RISCORE

Figure 1: Standard **FS** prompting vs RISCORE prompting: by encouraging reasoning-based selection of exemplars (riddle in green) in place of semantic similarity selection (riddle in red), the model unlocks the reasoning pattern, guided towards the correct answer.

At the same time, prompting is a widespread technique of extracting reasoning patterns from LLMs (Qiao et al., 2023). Especially Chain-of-Thought (CoT) prompting (Wei et al., 2023), as naturally designed to trigger the extraction of reasoning chains, is indeed proven to be one successful and viable prompting strategy in practice, even though its functionality is only applicable on larger models. Other promising techniques, such as in-context learning, are highly capable when the right data samples are utilized as exemplars, but fail to properly drive LLMs otherwise (Dong et al., 2024). While similarity-driven exemplar selection serves as a well-established rule of thumb (Liu et al., 2022;

Qin et al., 2024; Dong et al., 2024), the notion of similarity becomes more obscure when concerning reasoning patterns: for example, the sentences *"A man shaves everyday, yet keeps his beard long"* and *"Tom attends class every day but doesn't do any homework"* (Jiang et al., 2023b) in fact represent the exact same reasoning pattern, despite being rather distant semantically.

Driven by this observation, we assume that by "stripping" a sentence from linguistic features, while focusing on pure reasoning patterns prevalent in sentences, we can provide the model with the critical information to unlock the desired underlying reasoning pattern. Sentences that represent the same reasoning pattern are provided as "context reconstruction" counterparts in BrainTeaser data (Jiang et al., 2023b) or can be automatically generated using our proposed method. Therefore, incorporating such reconstructed instances as few-shot exemplars, together with their original counterparts, offers an alternative few-shot strategy that prioritizes reasoning similarity and analogy. An example of the method is provided in Figure 2. To this end, we contribute to the following:

- We experimentally verify that providing a riddle along with its context reconstruction can enhance performance on both lateral and vertical thinking problems.

- We propose RISCORE (Example provided in Figure 1), a novel prompting method designed to enhance the in-context riddle-solving capabilities of LLMs. As a supplement to this, we introduce an algorithm for generating contextually reconstructed multiple-choice riddles.

- We compare RISCORE against a wide array of popular prompting techniques, highlighting its effectiveness across several prompting alternatives and models of varying sizes.

2 Related work

Reasoning with Language Models Multiple reasoning senses have been studied over the years using language models, including commonsense (Sap et al., 2020), arithmetic (Luo et al., 2024), abductive (Zhao et al., 2023), inductive (Han et al., 2024), deductive (Sanyal et al., 2022), analogical reasoning (Sultan and Shahaf, 2022) and others. Linear thinking processes that exploit rules and
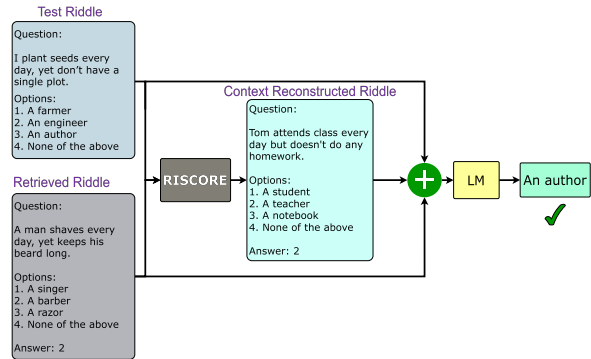

Figure 2: An overview of RISCORE, where the reconstructed instances, along with their original counterparts, are incorporated as exemplars in the few-shot setting to enhance the model's riddle solving ability.

logic, termed as "vertical thinking", have been previously explored within popular datasets (Lin et al., 2021; Bisk et al., 2020), unveiling interesting reasoning patterns of language models. On the other hand, creative thinking has been widely underexplored and often purposefully excluded from reasoning benchmarks (Speer et al., 2017; Sap et al., 2019), leading to a significant gap in literature, especially given the emergent capabilities that larger models present (Wei et al., 2022). Puzzle solving closely lies to creative reasoning (Giadikiaroglou et al., 2024; Lin et al., 2021; Zhang and Wan, 2022), since out-of-the-box thinking is mainly required. Going one step further, overriding default presuppositions and associations occurring when reasoning leads to "lateral thinking" processes, leading to more tricky puzzles, as demonstrated in the Brain-Teaser dataset (Jiang et al., 2023b) for the first time. In our work, we focus on probing vertical and lateral puzzle-solving reasoning abilities of language models via prompting.

LLMs and prompting Discovering reasoning patterns in LLMs is often performed via various types of prompting (Qiao et al., 2023). Zero-shot strategies harness simple but intuitive instructions, such as "Let's think step-by-step" that successfully improve LLM reasoning (Kojima et al., 2023); however, the search space of such magic prompts seems indefinite. Querying for intermediate reasoning steps was framed in the seminal Chain-of-Thought (CoT) prompting (Wei et al., 2023), proving that it is sufficient to elicit reasoning in larger models. Nevertheless, the placement of demonstrations in few-shot prompting faces several challenges (Dong et al., 2024); specifically, the way of selecting these
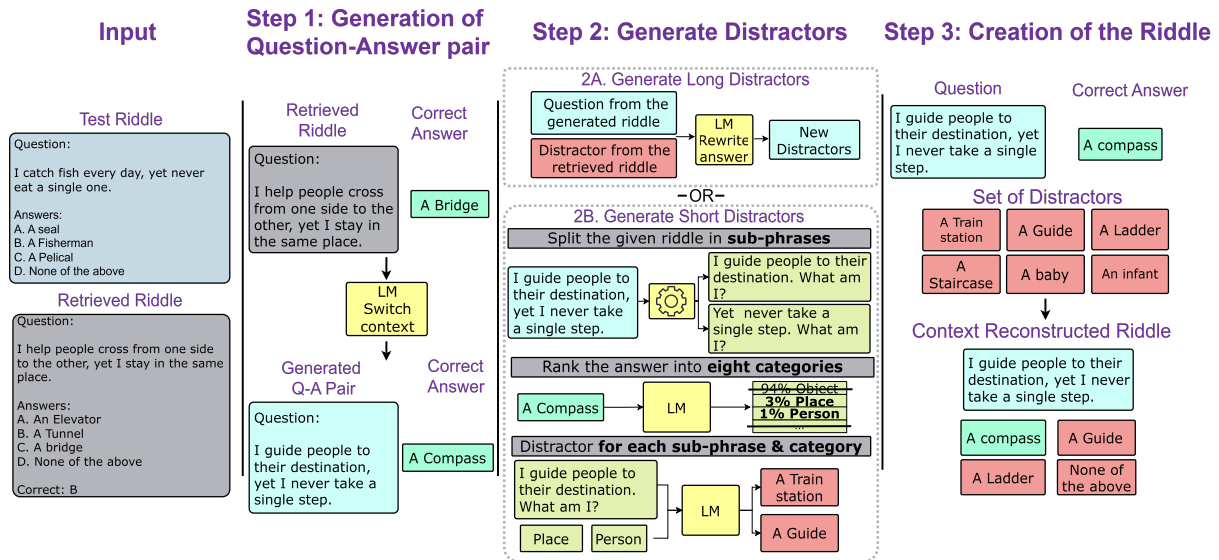
Figure 3: An overview of the automated method for generating a context-reconstructed riddle.

exemplars themselves is prone to instabilities (Lu et al., 2022; Min et al., 2022), proving the non-negotiable need for developing optimal exemplar selection techniques. Similarity-based retrieval is established as a default selection technique (Liu et al., 2022; Qin et al., 2024; Dong et al., 2024), with related enhancement optimizing the ordering of selected exemplars (Wu et al., 2023). With a focus on reasoning tasks, complexity of reasoning paths (Fu et al., 2023) or diversity of exemplars (Zhang et al., 2022) have already proven their merits, shifting the weight toward uncovering the patterns hidden in the data themselves that are able to advance reasoning, rather than similarity-driven placement based on semantics. Inspired by the above, we craft exemplars that promote the hidden reasoning patterns rather than the semantics of related data samples, maintaining simple similarity-based retrieval for their placement; by outperforming other prompting techniques, we demonstrate that highlighting those reasoning patterns is adequate for advanced LLM reasoning, without further few-shot engineering.

3 Method

3.1 RISCORE prompting

Consider the following two riddles: *R1: "A man shaves every day, yet keeps his beard long"* and *R2: "What has a beard but never needs to shave?"*. Although these riddles are semantically similar, follow the same structure, and refer to the same objects, their reasoning processes differ. In *R1*,

where the answer is "A barber," the word "beard" is used in the context of human grooming and personal appearance. In *R2*, where the answer is "A tree," "beard" is used in the botanical or natural context, referring to the "beard" of certain trees, such as the "oak". Thus, when using these riddles as context for a new riddle—such as *"I plant seeds every day, yet don't have a single plot"*—the answer will depend on the interpretation of the phrase "plant seeds", based on its contextual framing. In a creative context, the correct answer is "An author". Authors "plant seeds" of ideas through their writing but do not have a literal "plot", unless it refers to the abstract concept of a story plot, which adds layers of interpretation. On the other hand, "A farmer" is not the correct answer because a farmer typically works with physical plots of land to plant seeds, which contradicts the phrase "yet don't have a single plot."

In this example, however, instead of using *R2*, a contextually reconstructed version of *R1—R3: "Tom attends class every day but doesn't do any homework"*—would provide a clearer reasoning process for the model. Although *R2* is semantically closer to *R1*, its use of "beard" in a natural or botanical context introduces a different reasoning pathway, illustrating how the contextual framing of a riddle can clarify or obscure its intended answer.

Building on the previous example, we propose the RISCORE prompting method, designed to enhance the in-context riddle-solving abilities of LLMs. The RISCORE method supplements each exemplar in **FS** learning with a contextually recon-

structed version of itself. This preserves the desired reasoning process while only altering its context. This approach enables the model to delineate a clear and coherent reasoning trajectory, which it can then follow to effectively solve new riddles.

As illustrated in Figure 2, our approach extends existing few-shot (FS) methods (Dong et al., 2024; Wang et al., 2024; Sultan and Shahaf, 2022), with RISCORE not involved in the exemplar selection process[2]. Specifically, the goal of our method is to **augment FS samples** with automatically generated context-reconstructed examples (detailed in Section 3.2). The inclusion of these context-adapted examples has, in most cases, demonstrated greater impact in improving model performance, even outperforming the use of real examples extracted directly from the dataset (see Section G).

3.2 Method for generating context reconstructions

In this section, we present our approach for generating high-quality contextually reconstructed riddles to serve as few-shot exemplars for a model in a Multiple-Choice Question Answering (MQA) format. These exemplars, combined with the original pairs, aim to enhance the model's performance on lateral and vertical thinking tasks. Our method builds on the semi-automated pipeline proposed in the BrainTeaser paper (Jiang et al., 2023b), but we advance this framework by fully harnessing the capabilities of LLMs to automate the process. An overview of the automated method for generating a context-reconstructed riddle is provided in Figure 3.

Step 1: Generation of Question-Answer pair The first step involves generating one contextually reconstructed Question-Answer pair per selected instance, temporarily ignoring distractors. *Distractors* constitute the incorrect answer options presented alongside the correct answer in multiple-choice settings, designed to appear plausible and test the depth of understanding or reasoning of the model. To achieve this, we provide the LLM with the riddle, the correct answer, and a system prompt that outlines the task. The prompt instructs the model to analyze the given Question-Answer pair, understand the riddle, identify the reasoning process that links the question to the answer, and then generate a similar riddle that follows the same rea-

soning process, along with a corresponding correct answer. Since the model is provided with both the riddle and the correct answer, understanding the riddle becomes easier than solving it independently (Sultan and Shahaf, 2022). To further refine this approach, we apply the method in both Zero-shot (**ZS**) and Few-shot (**FS**) settings, using pre-existing pairs of original and contextually reconstructed questions from BrainTeaser. The prompts used in this process can be found in App. A.2. After generating the Question-Answer pair, a filtering procedure was conducted to ensure the quality of the instance and its alignment with the riddles in the dataset. This process involves applying rules concerning both the structure of the riddle and the answer. These rules are dataset-specific and can be adjusted to suit different datasets, ensuring adaptability across various contexts. Further analysis regarding this step can be found in App. A.1.

Step 2: Generation of the distractors This process involves generating incorrect answers to complement the multiple-choice options. While this task may initially appear straightforward, it presents several significant challenges. First, the distractors must not only match the number of original options, but also be guaranteed to be incorrect when compared to the correct answer. Additionally, they should not deviate excessively from the correct answer, as significant divergence could undermine the validity and challenge of the riddle. It is also essential to account for cases where the correct answer is "None of the above", as observed in the BrainTeaser dataset, ensuring that the other three alternatives are wrong enough.

The length of the answers also plays a crucial role in this process. For instance, BrainTeaser includes mostly answers with more than four or five words, while RiddleSense provides typically single-word answers. These variations in length introduce additional challenges in crafting effective distractors. To address this, we propose two distinct methods for generating distractors: one for long and another for short distractors, respectively.

Generation of Long Distractors In this approach, the original distractors of each riddle are given to an LLM alongside the new Question generated in Step 1. For each distractor (processed one at a time, except for "None of the above"), the model is prompted to modify its context by integrating it with the context of the new riddle. Regardless of the quality of this integration, the distractor is

guaranteed to be incorrect, though lower-quality incorporation may slightly reduce its overall effectiveness. If the original riddle's answer differs from "None of the above", but the new riddle's answer matches it, an additional distractor must be generated. In such cases, we simply prompt the LLM to generate a new distractor based on the generated Question-Answer pair. Details of the aforementioned step can be found in App. B.1.

Generation of Short Distractors The second method for generating distractors is applied to datasets like RiddleSense, which contain single-word answers. In this scenario, modifying a single word like "water" to fit the riddle's context presents a challenge. To address this, we first classify the generated Answer of Step 1 into a set of mutually exclusive categories (details in App. B.2), which informs the generation of distractors. For instance, in the riddle "I have a beard, but I never shave. What am I?" (where the correct answer is "A tree"), the system would classify the answer under the category "Nature".

Next, to ensure the distractor is incorrect, we break the riddle into smaller phrases. We also detect the presence of interrogative words, and if the question is purely descriptive without a direct query, we append the phrase "What am I?" at the end. For the previous example, we split the riddle into two parts: "I have a beard, what am I?" and "I never shave. What am I?" These phrases are then provided as inputs to the LLM, and only answers that differ from the correct one are retained. Since the LLM has access only to a phrase, we expect the generated answer to be incorrect. However, this is not always the case. In the example above, the response to "I have a beard, what am I?" might be "An oak," which is also a valid answer for the riddle.

To address this issue, we leverage the classification of the Answer to guide distractor generation. Specifically, we provide the LLM with two closely related categories, excluding the correct answer's category (e.g., "person" and "place"). The model is then prompted, through a system-user instruction, to generate a valid answer within the given context and category. This approach ensures that the generated distractors are both incorrect and contextually relevant, thereby preserving the difficulty of the riddle. After applying a filtering stage to ensure the distractors are distinct, we use WordNet (Miller, 1992) to augment them if their quantity is insuffi-

cient. Further details can be found in App. B.2. Additionally, for both methods, the same filtering procedure applied to the correct answer is also used to ensure the quality of the distractors.

Step 3: Creation of the Riddle In the final step of this pipeline, we use the Question-Answer pair, randomly shuffle the distractor set, select the required number of distractors, and place the correct answer in a randomly assigned position, so that a possible position bias is eliminated. More details regarding the Prompting process for our method can be found in App. C.

4 Experiments

We evaluate various LLMs using different datasets with our method and compare their performance against multiple baselines.

4.1 Datasets

For our experiments, we utilize two different datasets, namely BrainTeaser and RiddleSense, addressing vertical and lateral reasoning respectively. A comprehensive analysis of the datasets used in our experiments can be found in App D.

We select BrainTeaser primarily for two key reasons. First, BrainTeaser provides high-quality, manually crafted context reconstructions, which establish an upper bound on the model's performance when these are incorporated into the input. This upper bound also serves as a benchmark for evaluating the quality of riddles generated by the automated method. Second, the dataset consists of riddles that test the lateral thinking abilities of the models, a reasoning process that has proven as a challenging cognitive process for LLMs (Giadikiaroglou et al., 2024). Additionally, experiments using RiddleSense are conducted, emphasizing on the vertical thinking abilities of models, while verifying our method's broad applicability across different types of riddles. Given that hand-crafted context reconstructions are not available for RiddleSense, the automated method was utilized without ground-truth reconstructed instances in this set of experiments.

4.2 Baselines

In our study, we use various prompting techniques as baselines, including zero-shot (**ZS**), few-shot (**FS**), and Chain of Thought (**CoT**) methods. A detailed analysis of each technique is provided in App. E.

Chain-of-thought (CoT ZS) In this experiment, we prompt the model to answer the multiple-choice riddle in a step-by-step manner, following the method proposed in Wei et al. (2023), without demonstrating any examples. This approach allows us to assess the models' riddle-solving abilities in a zero-shot setting.

Few-Shot (FS) To assess the influence of example-based learning, we incorporate **FS** prompting with varying numbers of examples: 2, 4, and 8 shots. We limit our tests to 8 shots since it has been proven that reasoning abilities of LLMs are deteriorating with longer inputs (Levy et al., 2024). Two distinct strategies were used for selecting these examples: random selection, referred to as **Rand**, and selecting riddles with minimal semantic distance from the test riddle using Zhang et al. (2024)[3] as the semantic similarity model, referred to as **Sim**.

Few-shot with CoT Explanations (CoT FS) A straightforward way to aid the model in understanding an exemplar is by providing a supplementary explanation. In this experiment, we explore this approach by offering explanations alongside the few-shot exemplars. This allows us to evaluate whether these crafted explanations can improve the models' ability to comprehend and follow the required reasoning process. The explanations are generated using a semi-automated process. First, for each exemplar in the **FS** setting, we use ChatGPT[4] to generate an explanation for the riddle based on the correct answer, following the method proposed in Wei et al. (2023). These explanations were then manually curated to ensure alignment with human interpretations of the reasoning process required for the riddle. Due to the labor-intensive process of annotating the explanations, this experiment is conducted using only the 2, 4, and 8-shot settings with a randomly selected sampling strategy.

4.3 RISCORE

RISCORE is applied as follows. For each example used in the **FS** setting, the corresponding contextually reconstructed riddle is appended to the input. For a fair comparison, the number of shots refers to the total number of examples included in the prompt, regardless of whether they are new riddles

retrieved from the dataset or contextually reconstructed examples. Thus, for example RISCORE with 4 shots has the same 2 examples used in the 2-shot **FS** technique augmented with the 2 contextual reconstructed samples of them; ultimately, RISCORE keeps only half of the examples drawn from the original dataset, while the remainder consists of reconstructed examples. Therefore, the 2, 4, and 8-shot RISCORE settings correspond to 1, 2, and 4 exemplars drawn from the dataset respectively, with the rest of the demonstrations being their context reconstructions.

We primarily utilize Llama3-8B and Llama3-70B to generate both Question-Answer pairs and distractors. The Question-Answer pairs are produced in both **ZS** and **FS** settings. More details regarding these techniques can be found in App. E.

Lastly, in our experiments, we differentiate between RISCORE$_m$, which utilizes manually created reconstructions (when available, as in existing BrainTeaset context reconstructions), and RISCORE, which employs our fully automated method for generating reconstructions. The prompt structure for the system-user interaction in this method is the same as in the **FS** method outlined in App. E. The sole distinction is in the examples provided.

4.4 Models

To this end, we test various models of different scales on their reasoning abilities. Specifically, we examine Llama3 with 8 and 70 billion parameters respectively (AI@Meta, 2024), Mistral-7B and Mistral-8x7B (Jiang et al., 2023a), as well as Qwen2-7B (Yang et al., 2024). This diverse set of models allows us to investigate the impact of contextually reconstructed examples on models with varying sizes and parameterizations. By treating each model as a black-box (we merely prompt them and gather their responses), RISCORE adapts seamlessly to both open-source and proprietary models, demonstrating its versatility across a wide range of LLMs. Further details regarding the model selection process and technical specifications can be found in App. F.

5 Results

In this section, we present and analyze the results of our experiments. Additional results utilizing different metrics and hyperparameter settings are available in App. G.

---

[3] https://huggingface.co/Alibaba-NLP/gte-large-en-v1.5

[4] Specifically, we used version gpt-3.5-turbo-0125.

Method	N.	Llama3-70B	Mistral-8x7B	Llama3-8B	Mistral-7B	Qwen2-7B
CoT ZS	0	0.725	0.550	0.633	0.450	0.458
Randomly Selected Shots						
CoT FS	2	0.758	0.617	0.633	0.475	0.608
	4	0.683	0.583	0.608	0.508	0.650
	8	0.708	0.642	0.658	0.508	0.667
FS Rand	2	0.775	0.617	0.633	0.517	0.642
	4	0.808	0.683	0.642	0.483	0.608
	8	0.775	0.617	0.675	0.483	0.642
RISCORE$_m$	2	0.783	0.625	0.667	0.458	0.608
	4	0.758	0.617	0.675	0.517	0.625
	8	0.800	0.650	0.667	0.400	0.592
Semantically Similar Shots						
FS Sim	2	<u>0.825</u>	<u>0.692</u>	0.700	0.517	0.600
	4	0.792	0.683	0.717	0.458	0.633
	8	0.783	0.667	<u>0.767</u>	<u>0.533</u>	<u>0.650</u>
RISCORE$_m$	2	0.783	0.675	**0.767**	0.483	0.667
	4	**0.833**	0.708	0.742	**0.567**	0.642
	8	0.808	**0.708**	0.758	0.550	**0.667**

Table 1: Model performance for *BrainTeaser* using baselines and RISCORE$_m$ prompting. The best **FS** results are <u>underlined</u>, while best overall results per model are highlighted in **bold**. More detailed results can be found in Tables 6 and 7.

5.1 BrainTeaser results

Table 1 presents the performance outcomes of various prompting techniques applied to BrainTeaser data. A key observation is the underperformance of the **CoT FS** method relative to **FS** techniques, even when examples are supplemented with manually crafted explanations. This underperformance persists regardless of the model size. As expected (Dong et al., 2024), exemplar selection based on semantic similarity (**FS Sim**) consistently improves results when chosen over random selection (**FS Rand**), highlighting the value of semantic relevance to shot selection. This is a significant reference point for RISCORE, since a suitable initial exemplar selection is critical in properly guiding model reasoning when enhanced with contextually reconstructed examples.

Importantly, RISCORE$_m$ consistently outperforms **FS Sim** across all 4-shot and 8-shot configurations, as shown in Table 1. For instance, when comparing models with the same number of examples—such as 4-shot **FS Sim** versus 2+2-shot RISCORE$_m$, or 8-shot **FS Sim** versus 4+4-shot

Method	N.	Llama3-70B	Mistral-8x7B	Llama3-8B	Mistral-7B	Qwen2-7B
Llama3-70B ZS for QA & Llama3-8B distractors						
RISCORE	2	0.792	0.667	0.625	0.492	**<u>0.625</u>**
	4	**<u>0.792</u>**	0.642	0.675	**<u>0.467</u>**	0.625
	8	**<u>0.808</u>**	**<u>0.683</u>**	0.700	0.475	0.642
Llama3-70B FS for QA & Llama3-8B distractors						
RISCORE	2	0.750	0.675	0.683	0.475	**<u>0.625</u>**
	4	**<u>0.792</u>**	0.650	0.658	**<u>0.558</u>**	**<u>0.658</u>**
	8	**<u>0.808</u>**	**<u>0.675</u>**	0.742	0.517	**<u>0.658</u>**
Llama3-70B FS for QA & Llama3-70B distractors						
RISCORE	2	0.783	0.667	0.683	0.500	**<u>0.617</u>**
	4	**<u>0.792</u>**	0.642	0.667	**<u>0.508</u>**	0.617
	8	0.767	**<u>0.683</u>**	0.700	0.500	0.617

Table 2: Model performance for *BrainTeaser* using RISCORE prompting. Similarity-based selection was employed for choosing all the exemplars. Results that surpass the **FS** method with semantically similar examples (**FS Sim**, check Table 1), using the same number of shots, are **<u>underlined</u>**. More detailed results can be found in Tables 7 and 8.

RISCORE$_m$—RISCORE$_m$ demonstrates superior performance. The advantages of RISCORE$_m$ are particularly evident in its ability to maintain robust reasoning.

To better understand this advantage, we can examine the ability of RISCORE$_m$ to mitigate noise from suboptimal shot selections, a notable issue in the **FS Sim** results. For example, with the Llama3-70B model, the 2-shot **FS Sim** achieves a performance score of 0.825. However, adding two more semantically similar examples reduces the score to 0.792. In contrast, the 2+2-shot RISCORE$_m$ achieves a score of 0.833—a 4% improvement over **FS Sim** with the same number of examples (vs 4-shot). This trend is consistent across other models. Similarly, with the smaller Mistral-7B model, the 2-shot **FS Sim** score is 0.517, but adding two more examples reduces it to 0.458. In contrast, the 2+2-shot RISCORE$_m$, using contextually reconstructed examples, achieves a score of 0.567, outperforming **FS Sim** under identical shot conditions (vs 4-shot). These results indicate that RISCORE$_m$ effectively mitigates the noise introduced by suboptimal shot selections.

Using the automated method The manually curated RISCORE$_m$ results serve as a benchmark, showcasing the potential upper bound of performance gains achievable with high-quality, handpicked riddles and distractors. Despite this, the automated RISCORE method consistently enhances performance by effectively leveraging reconstructed contexts, as shown in Table 2. While the improvements using the automated RISCORE are understandably smaller than those achieved with manually curated examples (check RISCORE$_m$ results of Table 1), they are nonetheless consistent and significant across all models tested. For example, Llama3-70B demonstrates significant improvement when transitioning from **FS Sim** with 8 semantically similar shots (0.783) to the RISCORE-augmented setting with 8 shots (0.808), which incorporates the original examples and their automated reconstructions. This trend is evident across all relatively smaller models, with noticeable performance gains. For Qwen2-7B, a 2-shot and 4-shot RISCORE setup results in a 2.5% improvement compared to the same-shot **FS Sim** setting, while for Mistral-7B, the improvement is even more pronounced—rising by up to 10% from a baseline of 0.458 under certain conditions in the 4-shot setting.

These results underscore the value of context reconstruction, particularly when the number of shots is limited, demonstrating that automated reconstruction can effectively complement semantically chosen examples. Interestingly, the results reveal that in most cases, RISCORE's approach of combining N/2 semantically selected examples with their automated reconstructions outperforms the mere placement of the same number of examples retrieved from the dataset (indicated by the underlined results). However, there are instances where **FS Sim** outperforms RISCORE, primarily due to the selection constraints of RISCORE. In cases where the semantic similarity algorithm identifies an optimal example ranked lower than N/2 in a full **FS** setup, RISCORE may miss these shots due to its focus on N/2 examples plus reconstructions. This limitation indicates that RISCORE's performance can be constrained by the quality of initial semantic similarity rankings and the resulting shot selection.

5.2 RiddleSense

In this dataset, RISCORE can only be applied to automatically generated examples because the dataset's format lacks context reconstructions for its questions.

Table 3 presents the results of the baseline techniques using various models in the RiddleSense dataset. Once again, the results confirm that the few-shot technique, utilizing semantically similar exemplars for in-context learning, consistently delivers the best performance across all tested models.

In Table 4, we present the RiddleSense results using the proposed method for context reconstruction of each input. A clear trend emerges when comparing a simple 8-shot exemplar selection based on semantic similarity with the 8-shot setting of our method, where the top 4 semantically similar examples are augmented with our generated contextual pairs. Notably, the results show that our method consistently outperforms the standard 8-shot exemplar approach, demonstrating a significant improvement in model performance across various instances. An example of this trend is observed with the Llama3-8B model, where our method scores 0.708, approximately 2% higher than the few-shot setting based on semantic similarity, which achieves a score of 0.681. In the case of Mistral-8x7B, RISCORE avhieves a score of 0.700, a 2.5% gain over **FS Sim**'s 0.675, further demonstrating its consistent advantage. The same pattern is evident when comparing the two 4-shot settings. We achieve similar or marginally better accuracy using a total of just 4 examples—two original and two generated contextual reconstructions. This underlines the effectiveness of integrating contextually reconstructed pairs in enhancing model accuracy. We do not achieve a significant performance boost; however, we attain similar or marginally better results while relying on less grounded knowledge. This demonstrates the efficiency of our method, as it maintains comparable performance with fewer, yet strategically selected, exemplars, shifting the focus towards reasoning relevance of exemplars rather than quantity.

5.3 Quality of context reconstructed riddles

To generate contextually reconstructed riddles, we use Llama3 models with 8 billion and 70 billion parameters in both **FS** and **ZS** settings. We find that the Llama3-8B model struggles to produce high-quality Question-Answer pairs for the BrainTeaser dataset and is therefore not suitable for RISCORE. This difficulty likely arises from the BrainTeaser dataset's demand for lateral thinking, which is particularly challenging for smaller reasoners. The

Method	N.	Llama3-70B	Mistral-8x7B	Llama3-8B	Mistral-7B	Qwen2-7B
CoT ZS	0	0.775	0.675	0.619	0.589	0.608
Randomly Selected Shots						
CoT FS	2	0.789	0.692	0.625	0.594	0.667
	4	0.783	0.686	0.672	0.603	0.656
	8	0.783	0.697	0.658	0.597	0.625
FS Rand	2	0.769	0.706	0.672	0.586	0.689
	4	0.772	0.719	0.639	0.586	0.683
	8	0.800	0.711	0.672	0.586	0.700
Semantically Similar Shots						
FS Sim	2	0.792	**0.714**	0.706	0.608	0.722
	4	**0.817**	0.692	**0.711**	**0.633**	0.714
	8	0.800	0.675	0.681	0.611	**0.731**

Table 3: Model Performance for *RiddleSense* using baseline techniques. The best results overall are in **bold**. More detailed results can be found in Table 9. Note that no RISCORE$_m$ numbers are reported, since *RiddleSense* does not contain any ground truth reconstructed riddles.

Method	N.	Llama3-70B	Mistral-8x7B	Llama3-8B	Mistral-7B	Qwen2-7B
Llama3-70B fewshot for QA & Llama3-70B distractors						
RISCORE	2	**0.792**	0.672	0.692	0.600	0.697
	4	0.783	0.689	**0.722**	0.600	**0.717**
	8	0.789	**0.700**	**0.708**	0.597	**0.731**
Llama3-70B fewshot for QA & Llama3-8B distractors						
RISCORE	2	0.786	**0.719**	0.681	0.603	0.681
	4	0.789	0.686	0.686	0.606	0.697
	8	0.775	**0.689**	**0.706**	**0.617**	0.719
Llama3-8B zeroshot for QA & Llama3-8B distractors						
RISCORE	2	**0.792**	0.681	0.689	0.589	0.694
	4	0.778	**0.714**	0.700	0.600	0.683
	8	**0.806**	**0.689**	**0.686**	**0.614**	0.689

Table 4: Model performance for *RiddleSense* using RISCORE prompting. Similarity-based selection was employed for choosing all the exemplars. Results that surpass the FS method with semantically similar examples, using the same number of shots, are **underlined**. More detailed results can be found in Table 10.

Question-Answer pairs are essential, and if their quality is insufficient—as observed when leveraging Llama3-8B on BrainTeaser—high-quality distractors cannot compensate for this deficiency on their own. To address this issue, our preprocessing and filtering process ensures that only high-quality contextual examples are retained, maintaining the effectiveness of the method without being compromised by low-quality generations. However, for vertical thinking riddles, the smaller model effectively generates riddles even in the **ZS** setting, which, when used in the **FS** setting, can lead to increased performance compared to real examples drawn from the dataset.

6 Conclusion

We explore the riddle-solving capabilities of LLMs through multiple-choice formats, investigating how different prompts affect riddle performance requiring diverse reasoning. Our findings suggest that in few-shot settings, augmenting exemplars with contextually reconstructed examples improves performance. We introduce a novel prompting method, RISCORE, validated using the BrainTeaser dataset featuring manually crafted contextual riddles. Inspired by these results, we provide a method for automatically generating contextual reconstructions for multiple-choice riddles, demonstrating that RISCORE enhances LLMs' abilities in lateral and vertical thinking tasks.

Limitations

This method has some limitations. For instance, the placement of exemplars is based on semantic similarity, although cues unrelated semantically might be more crucial for reasoning, such as reasoning path similarity or diversity (Zhang et al., 2022). To address this, future work will employ more advanced methods for selecting exemplars for few-shot settings, like those suggested (Sultan and Shahaf, 2022). Additionally, the current study, having tested only two datasets, limits the broader impact of the conclusions since outcomes might vary with other datasets. Future steps will include testing the RISCORE prompting method with additional datasets across various tasks (Zhang and Wan, 2022). Finally, our experimentation is exclusively focused on the English language, therefore we cannot conclude whether our findings are applicable on other languages.

Ethical Considerations

There are no considerable risks associated with our work.

Acknowledgments

References

AI@Meta. 2024. Llama 3 model card.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Cognitive Science Laboratory, Princeton University. 2006. Wordnet - a lexical database for the english language. http://wordnet.princeton.edu/, last accessed on March 8th, 2006.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2023. Complexity-based prompting for multi-step reasoning. *Preprint*, arXiv:2210.00720.

Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. 2024. Puzzle solving using reasoning of large language models: A survey. *Preprint*, arXiv:2402.11291.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Simon Jerome Han, Keith J. Ransom, Andrew Perfors, and Charles Kemp. 2024. Inductive reasoning in humans and large language models. *Cognitive Systems Research*, 83:101155.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yifan Jiang, Filip Ilievski, Kaixin Ma, and Zhivar Sourati. 2023b. BRAINTEASER: Lateral thinking puzzles for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14317–14332, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *Preprint*, arXiv:2402.14848.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Preprint*, arXiv:1910.13461.

Bill Yuchen Lin, Ziyi Wu, Yichi Yang, Dong-Ho Lee, and Xiang Ren. 2021. RiddleSense: Reasoning about riddle questions featuring linguistic creativity and commonsense knowledge. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1504–1515, Online. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision. *Preprint*, arXiv:2406.06592.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang,

and Huajun Chen. 2023. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5368–5393, Toronto, Canada. Association for Computational Linguistics.

Chengwei Qin, Aston Zhang, Chen Chen, Anirudh Dagar, and Wenming Ye. 2024. In-context learning with iterative demonstration selection. *Preprint*, arXiv:2310.09881.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Soumya Sanyal, Harman Singh, and Xiang Ren. 2022. FaiRR: Faithful and robust deductive reasoning over natural language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1075–1093, Dublin, Ireland. Association for Computational Linguistics.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. Atomic: an atlas of machine commonsense for if-then reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Maarten Sap, Vered Shwartz, Antoine Bosselut, Yejin Choi, and Dan Roth. 2020. Commonsense reasoning for natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 27–33, Online. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: an open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Oren Sultan and Dafna Shahaf. 2022. Life is a circus and we are the clowns: Automatically finding analogies between situations and processes. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3547–3562, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Roxana Szomiu and Adrian Groza. 2021. A puzzle-based dataset for natural language inference. *arXiv preprint arXiv:2112.05742*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Liang Wang, Nan Yang, and Furu Wei. 2024. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta. Association for Computational Linguistics.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1423–1436, Toronto, Canada. Association for Computational Linguistics.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. 2024. Natural language reasoning, a survey. *ACM Comput. Surv.* Just Accepted.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang,

Wenjie Li, and Min Zhang. 2024. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *Preprint*, arXiv:2407.19669.

Yunxiang Zhang and Xiaojun Wan. 2022. Birdqa: A bilingual dataset for question answering on tricky riddles. *Preprint*, arXiv:2109.11087.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *Preprint*, arXiv:2210.03493.

Wenting Zhao, Justin Chiu, Claire Cardie, and Alexander Rush. 2023. Abductive commonsense reasoning exploiting mutually exclusive explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14883–14896, Toronto, Canada. Association for Computational Linguistics.

A   Question-Answer Pairs Generation

A.1   Process details

This section of our pipeline involves several steps, and subsequent steps after the initial one vary based on the dataset format.

Initially, we need to create contextual reconstructed Question-Answer pairs, temporarily disregarding distractors for both datasets. To generate these Question-Answer pairs, we can leverage the capabilities of large language models (LLMs) in two ways. First, we use a system-user prompt template to provide specific instructions to the model, guiding it through the process of producing a new Question-Answer pair. This approach maintains the underlying commonsense premise of the original question while altering both the question and the answer to fit a new situational context. This method operates in a zero-shot setting, meaning the model generates the new pairs without prior examples or training specific to the task. The second approach is quite similar but leverages the pre-existing pairs of original and contextually reconstructed questions from BrainTeaser. In this few-shot setting, we use the same instructions as in the zero-shot approach, but we enhance the model's understanding by providing in-context examples. Specifically, we include a pair consisting of an original Question-Answer and its corresponding contextually reconstructed version. This helps the model better grasp the task at hand. It is important to note that these few-shot examples are of high quality, given their manual construction and careful curation. In the few-shot approach, the provided examples are static. It is important to highlight

that, in BrainTeaser, when selecting examples for few-shot exemplars from the training dataset, we ensure that the examples chosen are not selected for reconstruction. This avoids any overlap between the few-shot exemplars and the examples for which we are requesting a context change, ensuring that the given pair examples are distinct from the requested ones. The corresponding system and user prompts regarding the above tasks can be found in App. A.2.

Models utilized   In this approach, we experiment with two models to assess the impact of model size on the quality of contextual reconstructions and the overall performance across both datasets. We use a relatively small model, Llama3-8B (AI@Meta, 2024), and a larger model from the same family, Llama3-70B (AI@Meta, 2024). These models were selected due to their strong performance across a diverse range of tasks, outperforming other models of comparable size. By incorporating both a smaller model, Llama3-8B, and a larger model, Llama3-70B, we aim to systematically investigate the effect of model size on the quality of contextual reconstructions and its overall impact on performance. This exploration allows us to gain deeper insights into how model capacity influences the effectiveness of the generated examples across different datasets.

Quality Assurance for Generated Text   After generating the contextual reconstructed Question-Answer pairs for all initially selected examples from the training set, we perform a small but necessary quality control to ensure that the generated pairs adhere to several high-level criteria. For the BrainTeaser dataset, the filtering process is more flexible, as the questions and answers can vary in length. The only constraint is that the questions must have a minimum length of 7 words, while the answers can be as long as necessary. As a result, the number of generated pairs filtered out is relatively low, around 2%, ensuring that the majority of reconstructions meet the quality criteria. In contrast, the RiddleSense dataset requires stricter limitations due to its typically shorter answers. For this dataset, questions must be at least 6 words in length, while answers must not exceed 7 words, reflecting the dataset's concise nature. As a result of these stricter rules, approximately 10% of the generated reconstructions for the RiddleSense dataset are filtered out for not meeting the required standards. It is important to note that the mini-

mum question length requirement is implemented to ensure that the generated questions maintain a sufficient level of complexity and depth. Questions with fewer than 6 or 7 words are unlikely to provide the necessary detail to establish a well-structured and coherent riddle. This limitation helps preserve the quality and integrity of the riddles by ensuring that they are adequately framed to challenge and engage the model effectively.

A.2 Prompts

First, we will present the **ZS** prompt configuration used for generating the contextually reconstructed Question-Answer pairs.

System Prompt:
You are an expert in context reconstruction. Your task is to receive a question along with its correct answer and adapt them to a new scenario while maintaining the misleading commonsense premise.

Please follow these steps:

- First, you will receive an unsolved riddle along with five answer options. Analyze the given setting and identify the connection between the question and its correct answer.

- Modify the original question and correct answer to fit a different situational context, ensuring that the underlying logic and relationship between them are preserved.

- Ensure that both the new question and the new correct answer are distinct from the originals.

User Prompt:

Question: ```{QUESTION}```

Correct answer: ```{ANSWER}```

Next, with minor modifications, we introduce the **FS** prompt configuration that follows the same logic but incorporates in-context examples to guide the model's responses.

System Prompt:
You are an expert in context reconstruction. Your task is to receive a question along with its correct answer and adapt them to a new scenario while maintaining the misleading commonsense premise.

Please follow these steps:

First, review an example provided with its context reconstruction, which illustrates the type of transformation you will need to perform.

Next, you will receive an unsolved riddle along with five answer options. Analyze the given setting and identify the connection between the question and its correct answer.

Modify the original question and correct answer to fit a different situational context, ensuring that the underlying

logic and relationship between them are preserved.

Ensure that both the new question and the new correct answer are distinct from the originals.

User Prompt:
{EXAMPLES}
Adapt the following riddle - answer pair while taking into consideration the examples above regarding context reconstruction:

Question: ```{QUESTION}```

Correct answer: ```{ANSWER}```

In the above setting, it is understood that the values of the **EXAMPLES** represent pairs of answered Question-Answer examples: the original and its contextually reconstructed counterpart, both sourced from the BrainTeaser dataset.

B Distractor Creation

B.1 BrainTeaser

In this process, we leverage the capabilities of large language models (LLMs) to understand and rephrase context in order to generate distractors. We use two different pipelines to produce at least three distractors, ensuring coverage of various potential situations. ① The first approach involves prompting the model with a system-user prompt that instructs it to analyze the given Question-Answer pair. The model is tasked with understanding the riddle, identifying the reasoning process that links the question to the answer, and then suggesting a distractor based on the more challenging or deceptive aspects of the concept. This approach yields one of the three required distractors. ② In the second approach, the process is more intricate and can result in distractors of questionable quality. We prompt the model with a system-user prompt, providing it with the reconstructed question without its answer, along with the original question's incorrect distractors, after removing the option "None of the above". The model is then tasked with modifying the concept of the given distractors by incorporating elements from the setting described in the question. Importantly, the correct answer is not provided to ensure that the generated distractors are distinctly different from the correct answer and effectively capture a varied interpretation of the question's context. Certainly, there are instances where the model generates sub-optimal contexts for the distractors provided. Despite these cases, the distractors remain incorrect and serve

their purpose, though they may not always be sufficiently challenging. Our experiments indicate that when the model is given both the original and the contextual reconstructed examples, the minor issue of lower-quality distractors does not significantly impact overall performance. Now, we have also created two new distractors that are somewhat contextually relevant to the setting, adding an additional layer of coherence to the generated options. This ensures that the distractors are not only incorrect but also related to the underlying premise of the riddle, enhancing the overall quality of the multiple-choice options. To create the final dataset, we randomly select two of the three generated distractors and shuffle them with the correct answer in random order. Finally, we append the option "None of the above" as the last choice. In cases where "None of the above" is the correct answer, we use all three generated distractors. With this approach, our dataset is prepared and ready for use.

Prompts used First, we will provide the system-user prompt used to task the model with understanding the riddle, identifying the reasoning process linking the question to the answer, and then suggesting a distractor based on the more challenging or deceptive aspects of the concept.

System Prompt:
Your task is to act as a concept grasper. You will be given a riddle and its correct answer.

Your goal is to understand the connection between the riddle and the correct answer, focusing on the tricky parts. Based on these tricky aspects, propose a plausible wrong answer that someone might give.

The wrong answer should be short, concise, and limited to one sentence.

- Riddle:

- Correct Answer:

Response format:

- Wrong Answer:

User Prompt:
```
- Riddle: {QUESTION}

- Correct Answer: {ANSWER}
```

Now we will provide the system-user prompt used for the second method.

System Prompt:
You will be given a sentence without context and then provided with a specific context.

Your task is to rewrite the sentence so that it aligns with the given context, while keeping it as close as possible to the original meaning.

The purpose is to adapt the sentence to the context, not to answer any questions related to the context.

- Sentence (out of context):

- Context:

Response format:

- Sentence:

User Prompt:
```
- Sentence (out of context): {ORI_CHOICE}

- Context:{QUESTION}
```

Here, the value of **ORI_CHOICE** refers to the distractors from the original instance, excluding the "None of the above" option. We create prompts for the model with each of these distractors individually.

B.2 RiddleSense

In this setting, the approach is fundamentally different. The answers and distractors in the original dataset are primarily one-word responses, while the questions feature detailed settings with punctuation, conjunctions, and more complex structures. To handle this, we also generate distractors using two distinct pipelines tailored to this format.

First pipeline The first pipeline involves a more granular approach by splitting the contextual reconstructed question into subphrases based on punctuation or conjunctions. If this yields fewer than three distinct subphrases, we further split the sentence at the position of the word "and" to create additional segments. We also detect the presence of interrogative words, and if the question is purely descriptive without any direct question, we append the phrase "What am I?" at the end. This is not chosen arbitrarily but follows the standard structure of many riddles in this dataset, where "What am I?" is the common question leading to single-word answers. Now that we have broken the riddle into sub-phrases and appended the appropriate question, we can prompt the model to generate a wrong answer for each sub-phrase concatenated with the question. This method ensures that the distractors align with different parts of the riddle's setting. However, since some sub-phrases may contain key ideas central to the riddle, the model may

still generate answers too similar to the correct one. To address this issue, we incorporate an additional intermediate step utilizing zero-shot classification. Specifically, we use the *facebook/bart-large-mnli* model [5] (Lewis et al., 2019) provided by Hugging Face. This model classifies the correct answer into one of eight general categories: *'food', 'person', 'object', 'animal', 'nature', 'time', 'place', 'concept'*. These categories are chosen to be mutually exclusive to avoid overlap. In our approach, we first predict the category of the correct answer using this model. We then leverage this classification to guide the generation of distractors. For each sub-phrase concatenated with the question, we provide the LLM with the two most similar categories (excluding the correct answer's category). The model is prompted with a system-user instruction to produce a correct answer of the given setting in the given category. This method ensures that the generated distractors are not only incorrect but also contextually aligned with the riddle, thus maintaining the challenge for the model. After applying filtering to ensure the distractors are distinct and relevant, we have produced several distractors. The LLMs utilized in this pipeline include the previously mentioned Llama3-8B (AI@Meta, 2024) and its larger counterpart, Llama3-70B (AI@Meta, 2024).

Second pipeline In cases where the distractors produced by the above pipeline are insufficient, we use WordNet (Cognitive Science Laboratory, Princeton University, 2006) to augment our distractor set. For each generated distractor, or if necessary, for the distractors from the original question, we retrieve synonyms and hyponyms from Word-Net. These additional terms are then included as potential distractors. After compiling these distractors, we randomly select four of them and add the correct answer in a random order to complete the set. This approach ensures that we have a diverse and comprehensive set of distractors for each question. To ensure the quality of our distractors, we impose a limitation that at least two of the required four distractors must be generated using the first approach. This is due to the fact that the distractors generated through WordNet augmentation tend to be of inferior quality compared to those produced directly by the model. If this requirement is not met, we skip the particular train set instance for producing a contextual reconstruction. This approach helps maintain the overall quality and relevance of

---
[5]facebook/bart-large-mnli

the distractors in our dataset.

Prompts used We will present the system-user prompt configuration used in the first pipeline. In this approach, for each sub-phrase concatenated with the question, we provide the model with two categories, excluding the correct answer's category. For each category, a separate system-user prompt is issued to instruct the model to generate a correct answer that fits the given setting within the specified category.

System Prompt:
Task: Provide a concise, relevant answer to the given question within the specified category.

Constraints:
- The answer should not exceed three words.
- Follow the exact format provided below.

Response Format:

Answer: ...

User Prompt:
```
Question:  ```{QUESTION}```

Category: {CATEGORY}
```

C Pairs Prompting

We have successfully created the desired contextual reconstructions for both datasets. However, an issue remains: some of the originally selected examples do not have corresponding reconstructed examples due to the quality control filtering process.

To address this issue, we first use the original most semantically similar examples as in-context learning exemplars, appending their corresponding automatically generated contextual pairs. If these exemplars are not sufficient for the required settings of two, four, or eight (i.e., **RISCORE**), we employ a more structured approach rather than adding examples randomly. ① We begin by generating embeddings for the set of original examples and their contextual reconstructions that have not already been included in the current exemplars. Using cosine similarity, we then identify the most similar examples from this set. Importantly, these similar examples might not be part of the original training set but could be among the reconstructed examples. ② We select the most similar examples and pair each with its corresponding pair, ensuring that we add either the original or the reconstructed

example as needed. This process is repeated until the number of exemplars meets the required quantity.

D Dataset Description

In this work we are using two different datasets of multiple QA format. ① The first one is the Brain-Teaser dataset, which consists of two sub-tasks regarding both lateral thinking challenges. In addition to the original puzzles, the dataset includes adversarial subsets created by manually modifying the original BrainTeaser while preserving their reasoning paths. The original data were perturbed in two ways: First, there is a *semantic reconstruction* of each original question, ensuring that the answers and distractors remain unchanged. Second, the original data undergoes *context reconstruction*, where the reasoning path is preserved, but the brain teaser is rephrased to describe a new situational context. Each question offers four possible choices, with the last option always being "None of the above". It is important to note that our experiments are focused exclusively on the sentence puzzle (SP) setting. ② The second dataset used in our experiments is RiddleSense. This dataset features riddle-style questions that require sophisticated commonsense reasoning and a strong grasp of figurative language to answer accurately. It is structured as a multiple-choice question answering task, with riddles that test the model's capacity to navigate and interpret nuanced commonsense scenarios. The reasoning challenges posed by RiddleSense are closely aligned with those found in BrainTeaser. RiddleSense offers one correct answer and four distractors for each riddle. This contrasts with BrainTeaser, which includes a "None of the above" option as the final choice, thereby introducing a more challenging setting.

D.1 Data statistics

Format of Datasets The BrainTeaser dataset is divided into three splits: train, development, and a hidden test set used for evaluation. Although RiddleSense follows a similar format, we did not have direct access to its hidden test set. As a result, we utilized the development data split of RiddleSense for our experiments.

Prepossessing and filtering Due to data leakage across the two datasets, we were required to exclude semantically similar examples from the test split utilized. This leakage is understandable, as

the sources for English riddles are limited, and the number of unique riddles is finite. After identifying semantically similar riddles, we retained only one instance within the datasets, prioritizing Brain-Teaser due to the higher quality of its questions and distractors, which were manually crafted. Additionally, hardware limitations made processing the full RiddleSense unfeasible. To manage this, we split its test set into two halves. To ensure the fairness and accuracy of this division, we evaluated various techniques on both halves. Following the completion of these procedures, we present the dataset statistics in the table below. Statistics regarding the datasets are provided in Table 5. More information regarding this process can be found in D.3.

Dataset	Train	Dev	Test
BrainTeaser - SP	$507_{(169x3)}$	$120_{(40x3)}$	$120_{(40x3)}$
RiddleSense (initial)		1021	
RiddleSense (filtered)	3510	720	—
RiddleSense (sampled 50%)		360	

Table 5: Data statistics

D.2 Evaluation metrics

Given that our task is formatted as a multiple-choice question-answering problem, the evaluation metric employed will be accuracy. This metric will provide a measure of the proportion of correctly answered questions out of the total number of questions presented. Although accuracy is a straightforward metric, it is effective in providing useful insights and serves as a reliable measure for evaluation in this context.

① For the BrainTeaser task, we not only track the overall accuracy but also monitor the accuracy for each specific type of instance, including original, semantic, and context reconstructions. This detailed tracking is feasible because the data in each set are balanced, with each original instance having corresponding semantic and context variations. This approach allows us to evaluate the model's proficiency in handling the same reasoning path under different conditions. The total accuracy is then calculated as the mean of these three individual metrics. We also track group-based accuracy, where a "group" refers to either two questions (original and semantically reconstructed) or three questions (original, semantically reconstructed and contextually reconstructed). This metric assesses the model's performance when all instances within

a group are answered correctly. Group-based accuracy provides a broader perspective on the model's ability to handle these lateral thinking challenges across different types of question reconstructions. This format of evaluation metrics was both provided and requested by the creators of the dataset in an open Kaggle competition, and we are adopting it for our analysis.

② Unfortunately, the simple data row format in RiddleSense restricts our ability to track more detailed metrics comprehensively, limiting us to evaluating only the overall accuracy for this dataset.

D.3 Dataset preprossessing

Our goal in this process is to ensure that no question from the RiddleSense test set is present in the combined train, dev, and test datasets of BrainTeaser. To achieve this, we are using the Sentence Transformers library [6] (Reimers and Gurevych, 2019) with the *gte-large-en-v1.5* [7] (Zhang et al., 2024) model to generate text embeddings and perform semantic similarity comparisons. We convert each question from the entire BrainTeaser puzzle into embeddings that represent the semantic meaning of each question. We then calculate the cosine similarity between these embeddings and those in the RiddleSense test set to identify duplicates or highly similar questions. A lower threshold is applied to define similarity, meaning that any question with a cosine similarity above this threshold is considered a near-duplicate and excluded from the test set. After conducting several experiments, we determined that a cosine similarity threshold of 0.9 was optimal. This threshold is a delicate balance, as our questions are riddle-style and often rely on abstract or metaphorical language, making it challenging to clearly assess whether two questions are similar enough to warrant exclusion. Since riddles are not always straightforward in their wording, the threshold needed to be fine-tuned to avoid excluding questions that are conceptually distinct but might have overlapping language, ensuring a fair evaluation without overly aggressive filtering. This means that any question from the RiddleSense test set that had a similarity score of 0.9 or higher to a question in the BrainTeaser datasets was removed to avoid redundancy. As previously mentioned, because we prioritize BrainTeaser questions due to their superior quality from manual annotation, we decided to remove any similar questions from the

RiddleSense test set. This approach helps preserve the integrity of the evaluation process and ensures that the test set contains only unique and distinct questions, minimizing potential data leakage and improving the robustness of our results. Additionally, by removing these instances, we can perform more experiments on the remaining unique questions, optimizing our use of hardware resources and potentially yielding more comprehensive insights.

E Prompting details

Zero-shot Chain of Thought Prompting Technique This is the most straightforward prompting technique. In this approach, each instance from the test set of each dataset is presented to the model using a specific system-user prompt format. This technique operates in a zero-shot setting, meaning the model is not given any prior examples or training specific to the task at hand. Instead, it relies solely on the system prompt and user prompt to generate responses based on its pre-existing knowledge. The chaoin of thought character is a result of the usage of the phrase *Let's think step by step* in the end of each system prompt. The Chain of Thought (CoT) character is achieved by including the phrase *"Let's think step by step"* at the end of each system prompt. This approach aligns with the findings of (Kojima et al., 2023), which demonstrate that large language models can perform effectively as zero-shot reasoners by incorporating this specific phrase before presenting an unanswered test instance. The following are the system-user prompts that were used:

System Prompt:
You will encounter a riddle that requires analytical thinking and reasoning to solve.
A riddle is a question or statement intentionally phrased so as to require ingenuity in ascertaining its answer or meaning, typically presented as a game.

Different ideas can be used in these riddles:
1. Riddles often employ misdirection, leading you away from the actual solution.
2. They include elements with double meanings, requiring a keen eye for words with dual interpretations.
3. Metaphorical wordplay adds another layer, urging you to decipher figurative language.
4. Look out for exaggeration, as riddles may present overly dramatic details to divert your attention.
5. Common phrases and sayings may hide within the puzzle, demanding familiarity.
6. Associations and irony play a crucial role, introducing unexpected connections.
7. Numerical puzzles can also be part of the mystery, requiring you to decode their significance.
8. Elemental imagery, drawn from nature, might hold key descriptors.

---

9. Rhyming and sound clues can add a poetic dimension. 10. Word Puzzles: Pay attention to anagrams, acrostics, and other wordplay elements.
11. Also, it is important to note you should decode the upcoming riddle using everyday logic and creativity.

Approach the riddle with everyday logic and creativity, avoiding supernatural explanations.
You will be given an unsolved riddle and five options to choose the answer amongst them. Let's think step by step.

User Prompt:

```
Riddle: ```
{RIDDLE}
```

Options:
[option 1]: ```{OPTION_1}```
[option 2]: ```{OPTION_2}```
[option 3]: ```{OPTION_3}```
[option 4]: ```{OPTION_4}```
[option 5]: ```{OPTION_5}```
```

Where the input parameters are self-explanatory. It is worth mentioning that for BrainTeaser, where the multiple-choice options are four, the prompt is adjusted accordingly.

**Few-shot Prompting Techniques** In this approach, we provide a specific number of exemplars from each dataset's training set before asking the model to solve a multiple-choice question from the test set. These exemplars, which include their correct answers, serve as in-context learning examples. By presenting these examples, we aim to guide the model and improve its performance on the subsequent unanswered tasks. This method leverages the provided context to enhance the model's ability to understand and solve new instances more effectively. The system prompt that is used here is the one of the zero-shot prompting with some minor changes. The number of examples that are used are 2, 4 and 8 as mentioned in section 3.

Now, a key decision is how to select the exemplars that will be provided to the model before it attempts to answer the test set questions. The simplest and most straightforward approach is to randomly choose these answered exemplars from the training dataset. This method, referred to as **FS Rand** in our experiments, offers a baseline for comparison, as it doesn't apply any specific strategy for selecting the most relevant examples and relies purely on random sampling.

Another approach implemented in our experiments for selecting exemplars is based on semantic similarity. Specifically, we utilize the Sentence

Transformers library [8] (Reimers and Gurevych, 2019) in conjunction with the *gte-large-en-v1.5* model [9] (Zhang et al., 2024). This setup allows us to generate text embeddings and perform semantic similarity comparisons using cosine similarity between these embeddings. For each experiment, we first generate text embeddings for all instances in the training datasets. Then, for each question in the test set, we generate its embedding and calculate its cosine similarity with all training set embeddings. Based on the number of in-context learning examples needed, we select the most similar training instances to use as exemplars. This method ensures that the selected examples are semantically relevant, aiming to improve the model's performance by presenting it with contextually aligned training examples. This approach is referred to as **FS Sim** in our experiments, as it focuses on semantic similarity to guide exemplar selection for improved task performance.

The system and user prompts used for both of the above settings are outlined as follows:

**System Prompt:**
You will encounter a riddle that requires analytical thinking and reasoning to solve.
A riddle is a question or statement intentionally phrased so as to require ingenuity in ascertaining its answer or meaning, typically presented as a game.

Different ideas can be used in these riddles:
1. Riddles often employ misdirection, leading you away from the actual solution.
2. They include elements with double meanings, requiring a keen eye for words with dual interpretations.
3. Metaphorical wordplay adds another layer, urging you to decipher figurative language.
4. Look out for exaggeration, as riddles may present overly dramatic details to divert your attention.
5. Common phrases and sayings may hide within the puzzle, demanding familiarity.
6. Associations and irony play a crucial role, introducing unexpected connections.
7. Numerical puzzles can also be part of the mystery, requiring you to decode their significance.
8. Elemental imagery, drawn from nature, might hold key descriptors.
9. Rhyming and sound clues can add a poetic dimension.
10. Word Puzzles: Pay attention to anagrams, acrostics, and other wordplay elements.
11. Also, it is important to note you should decode the upcoming riddle using everyday logic and creativity.

Approach the riddle with everyday logic and creativity, avoiding supernatural explanations.

First, you'll encounter **X** examples with their answer provided similar to the riddle you will need to solve.

---

Then you will be given an unsolved riddle and **X** options to choose the answer amongst them.

**User Prompt:**

`{EXAMPLES}`

Answer the following riddle while taking into consideration the examples above. Choose the best and the most logical option from the available choices:

```
Riddle: ```
{RIDDLE}
```
```

```
Options:
[option 1]: ```{OPTION_1}```
[option 2]: ```{OPTION_2}```
[option 3]: ```{OPTION_3}```
[option 4]: ```{OPTION_4}```
[option 5]: ```{OPTION_5}```
```

Where the **EXAMPLES** refer to the few-shot examples selected for each test set instance, and the other input parameters are self-explanatory. It is worth mentioning that for BrainTeaser, where the multiple-choice options are four, the prompt is adjusted accordingly.

Chain of Thought Prompting Technique Another promising technique for enhancing model performance across various tasks is Chain of Thought (CoT) prompting. This approach involves not only providing answered examples but also including an explanation consisting of intermediate reasoning steps. By laying out the thought process step-by-step, CoT prompting significantly improves the capability of large language models to tackle complex reasoning tasks. In our task, the Chain of Thought (CoT) prompting technique is anticipated to outperform the previously mentioned methods. This expectation is supported by the performance improvements observed in the CommonsenseQA dataset (Talmor et al., 2019), a domain similar to our setting that involves both lateral and vertical thinking. The dataset demonstrated significant improvements when using a manually composed set of two, four, or eight few-shot exemplars with CoT prompting, highlighting its effectiveness in eliciting successful reasoning. In our case, these explanations were generated manually to ensure they align with human perceptions of the reasoning process required by the riddle. The explanation format we used follows the structure outlined by (Wei et al., 2023) for few-shot exemplars in full chain-of-thought prompts for CommonsenseQA. Due to the similarity in the multiple-choice question-answer format of our datasets, it served as a good foundation for developing our own approach of creating

explanations. Below, we provide the system and user prompts used in the aforementioned technique:

System Prompt:
You will encounter a riddle that requires analytical thinking and reasoning to solve.
A riddle is a question or statement intentionally phrased so as to require ingenuity in ascertaining its answer or meaning, typically presented as a game.

Different ideas can be used in these riddles:
1. Riddles often employ misdirection, leading you away from the actual solution.
2. They include elements with double meanings, requiring a keen eye for words with dual interpretations.
3. Metaphorical wordplay adds another layer, urging you to decipher figurative language.
4. Look out for exaggeration, as riddles may present overly dramatic details to divert your attention.
5. Common phrases and sayings may hide within the puzzle, demanding familiarity.
6. Associations and irony play a crucial role, introducing unexpected connections.
7. Numerical puzzles can also be part of the mystery, requiring you to decode their significance.
8. Elemental imagery, drawn from nature, might hold key descriptors.
9. Rhyming and sound clues can add a poetic dimension.
10. Word Puzzles: Pay attention to anagrams, acrostics, and other wordplay elements.
11. Also, it is important to note you should decode the upcoming riddle using everyday logic and creativity.

Approach the riddle with everyday logic and creativity, avoiding supernatural explanations.

First, you'll encounter **X** examples demonstrating analytical reasoning similar to the riddle you will need to solve.

Then you will be given an unsolved riddle and five options to choose the answer amongst them. Let's think step by step and solve the riddle based on the examples provided above.

User Prompt:

`{EXAMPLES_COT}`

Answer the following riddle while taking into consideration the examples above. Choose the best and the most logical option from the available choices:

```
Riddle: ```
{RIDDLE}
```
```

```
Options:
[option 1]: ```{OPTION_1}```
[option 2]: ```{OPTION_2}```
[option 3]: ```{OPTION_3}```
[option 4]: ```{OPTION_4}```
[option 5]: ```{OPTION_5}```
```

Where the **EXAMPLES_COT** refer to the few-shot examples with manually generated explanations selected for each test set instance, and the other input parameters are self-explanatory. It is important to note that for BrainTeaser, where

there are four multiple-choice options, the prompt is modified accordingly.

F Experimental Setup

In our experiments, we employed the Google Colab platform and Kaggle, leveraging various open-source Python packages such as Transformers, BitsAndBytes, Accelerate (Gugger et al., 2022) and Sentence-Transformers. We also utilized the Amazon Bedrock API, which allowed us to access models without being constrained by hardware limitations, providing scalable and flexible model deployment.

We specifically opted for instruction variations of the models because they aligned more closely with our task requirements. Instruction-tuned models generally offer better performance in scenarios where understanding and following specific instructions is crucial. This alignment ensures that the models are better equipped to handle the nuances of the tasks, leading to more effective and relevant outputs.

The temperature and repetition penalty values were determined through a series of exploratory experiments. To ensure consistency across our work, we systematically applied the same parameters whenever possible. For our experiments, we used a temperature of 0.5 and repetition penalties of either 1.0 or 1.15.

Llama 3 (AI@Meta, 2024) In our experiments, we chose two variations of the model: the 8B and 70B versions. The Llama3-8B[10] model was used for inference without quantization, which allowed it to deliver results with full precision. This approach was ideal when computational resources were sufficient and precision was critical. On the other hand, the Llama3-70B [11] model, due to hardware limitations, underwent quantization. This process reduced the model's size and computational needs, making it accessible despite the constraints of our hardware. While quantization might lead to some loss of precision, it was necessary to deploy a model with more parameters, which otherwise would not have been feasible.

Mistral (Jiang et al., 2023a) In our experiments, we used the Mistral-7B-Instruct-v0.2[12] and the Mixtral-8x7B-Instruct-v0.1[13] in their unquantized forms. The Mistral 7B-Instruct v0.2 was selected for its strong instruction-following abilities, maintaining full precision. Similarly, the Mixtral-8x7B-Instruct-v0.1, which integrates eight 7B models, was used unquantized to benefit from its combined performance. Our goal was to explore the effectiveness of smaller models for our riddle tasks, and we preferred these instruction-tuned variations to ensure they were well-suited to our specific requirements, leading to more effective results.

Qwen2 (Yang et al., 2024) To further explore smaller model variations, we selected the Qwen2-7B-Instruct[14] version. The Qwen2 family of models, including this 7B-Instruct variant, is known for its strong performance in reasoning tasks. The Qwen2 models are designed with advanced instruction-following capabilities and are particularly effective at complex problem-solving and logical reasoning, making them well-suited for our objectives.

G Detailed Results

In this section, we present the detailed results for both datasets across all our experimental techniques. Due to the extensive nature of the experiments, the results for each dataset are organized into subtables. We begin with the BrainTeaser dataset, where the metrics are more detailed due to the dataset's structure, followed by the results for the RiddleSense dataset. The structure is organized by method per model, with results presented in descending order based on score for each method, rather than by the number of examples. This differs from the approach outlined in Section 5.

---

[10]meta-llama/Meta-Llama-3-8B-Instruct
[11]meta-llama/Meta-Llama-3-70B-Instruct
[12]mistralai/Mistral-7B-Instruct-v0.2
[13]mistralai/Mixtral-8x7B-Instruct-v0.1

[14]Qwen/Qwen2-7B-Instruct

Model	Method	Num.Ex	Task	Temp	Rep_Pen	Original	Semantic	Context	Ori. + Sem.	Ori. + Sem. + Con.	Average
						Chain-of-Thought Zero-shot					
Meta-Llama-3-70B-Instruct	CoT_ZS (Q)	0	SP	0.5	1.15	0.725	0.775	0.675	0.675	0.550	0.725
Mixtral-8x7B-Instruct-v0.1	CoT_ZS	0	SP	0.5	1.0	0.575	0.550	0.525	0.475	0.275	0.550
Meta-Llama-3-8B-Instruct	CoT_ZS	0	SP	0.5	1.0	0.625	0.650	0.625	0.500	0.325	0.633
Mistral-7B-Instruct-v0.2	CoT_ZS	0	SP	0.5	1.0	0.375	0.475	0.500	0.300	0.250	0.450
Qwen2-7B-Instruct	CoT_ZS	0	SP	0.5	1.15	0.475	0.450	0.450	0.350	0.200	0.458
						Few-shot with CoT Explanations					
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	2	SP	0.5	1.15	0.850	0.750	0.675	0.700	0.500	0.758
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	8	SP	0.5	1.0	0.675	0.750	0.700	0.600	0.425	0.708
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	4	SP	0.5	1.15	0.700	0.650	0.700	0.575	0.450	0.683
Mixtral-8x7B-v0.1	CoT_FS	8	SP	0.5	1.0	0.650	0.675	0.600	0.575	0.375	0.642
Mixtral-8x7B-v0.1	CoT_FS	2	SP	0.5	1.0	0.625	0.625	0.600	0.525	0.350	0.617
Mixtral-8x7B-v0.1	CoT_FS	4	SP	0.5	1.0	0.575	0.600	0.575	0.475	0.350	0.583
Meta-Llama-3-8B-Instruct	CoT_FS	8	SP	0.5	1.0	0.675	0.700	0.600	0.525	0.325	0.658
Meta-Llama-3-8B-Instruct	CoT_FS	2	SP	0.5	1.0	0.725	0.625	0.550	0.550	0.350	0.633
Meta-Llama-3-8B-Instruct	CoT_FS	4	SP	0.5	1.0	0.650	0.625	0.550	0.500	0.300	0.608
Mistral-7B-Instruct-v0.2	CoT_FS	8	SP	0.5	1.0	0.525	0.550	0.450	0.375	0.300	0.508
Mistral-7B-Instruct-v0.2	CoT_FS	4	SP	0.5	1.0	0.525	0.550	0.450	0.375	0.300	0.508
Mistral-7B-Instruct-v0.2	CoT_FS	2	SP	0.5	1.0	0.525	0.425	0.475	0.300	0.225	0.475
Qwen2-7B-Instruct	CoT_FS	8	SP	0.5	1.0	0.600	0.725	0.675	0.550	0.425	0.667
Qwen2-7B-Instruct	CoT_FS	4	SP	0.5	1.0	0.650	0.675	0.625	0.550	0.450	0.650
Qwen2-7B-Instruct	CoT_FS	2	SP	0.5	1.0	0.675	0.550	0.600	0.450	0.375	0.608
						Few-shot with Random Selection					
Meta-Llama-3-70B-Instruct	FS Rand (Q)	4	SP	0.5	1.15	0.825	0.850	0.750	0.800	0.700	0.808
Meta-Llama-3-70B-Instruct	FS Rand (Q)	8	SP	0.5	1.15	0.775	0.800	0.750	0.675	0.550	0.775
Meta-Llama-3-70B-Instruct	FS Rand (Q)	2	SP	0.5	1.15	0.750	0.800	0.775	0.725	0.600	0.775
Mixtral-8x7B-Instruct-v0.1	FS Rand	4	SP	0.5	1.0	0.675	0.775	0.600	0.600	0.400	0.683
Mixtral-8x7B-Instruct-v0.1	FS Rand	2	SP	0.5	1.0	0.650	0.625	0.575	0.475	0.300	0.617
Mixtral-8x7B-Instruct-v0.1	FS Rand	8	SP	0.5	1.0	0.675	0.650	0.525	0.525	0.350	0.617
Meta-Llama-3-8B-Instruct	FS Rand	8	SP	0.5	1.0	0.750	0.625	0.650	0.575	0.400	0.675
Meta-Llama-3-8B-Instruct	FS Rand	4	SP	0.5	1.0	0.625	0.650	0.650	0.500	0.375	0.642
Meta-Llama-3-8B-Instruct	FS Rand	2	SP	0.5	1.0	0.600	0.700	0.600	0.550	0.375	0.633
Mistral-7B-Instruct-v0.2	FS Rand	2	SP	0.5	1.0	0.525	0.550	0.475	0.425	0.275	0.517
Mistral-7B-Instruct-v0.2	FS Rand	8	SP	0.5	1.0	0.450	0.575	0.425	0.400	0.300	0.483
Mistral-7B-Instruct-v0.2	FS Rand	4	SP	0.5	1.0	0.550	0.425	0.475	0.325	0.225	0.483
Qwen2-7B-Instruct	FS Rand	2	SP	0.5	1.15	0.675	0.650	0.600	0.550	0.425	0.642
Qwen2-7B-Instruct	FS Rand	8	SP	0.5	1.15	0.700	0.600	0.625	0.525	0.425	0.642
Qwen2-7B-Instruct	FS Rand	4	SP	0.5	1.15	0.675	0.575	0.575	0.525	0.400	0.608

Table 6: Model performance for *BrainTeaser* (Part 1). The presence of **(Q)** in the method column indicates that the results correspond to the quantized version of the model.

Model	Method	Num.Ex	Task	Temp	Rep_Pen	Original	Semantic	Context	Ori. + Sem.	Ori. + Sem. + Con.	Average
Few-shot with Semantic Similarity											
Meta-Llama-3-70B-Instruct	FS Sim (Q)	2	SP	0.5	1.15	0.850	0.875	0.750	0.800	0.675	0.825
Meta-Llama-3-70B-Instruct	FS Sim (Q)	4	SP	0.5	1.15	0.825	0.775	0.775	0.700	0.575	0.792
Meta-Llama-3-70B-Instruct	FS Sim (Q)	8	SP	0.5	1.15	0.775	0.825	0.750	0.700	0.600	0.783
Mixtral-8x7B-Instruct-v0.1	FS Sim	2	SP	0.5	1.0	0.700	0.700	0.675	0.575	0.450	0.692
Mixtral-8x7B-Instruct-v0.1	FS Sim	4	SP	0.5	1.0	0.750	0.650	0.650	0.575	0.425	0.683
Mixtral-8x7B-Instruct-v0.1	FS Sim	8	SP	0.5	1.0	0.750	0.625	0.625	0.525	0.375	0.667
Meta-Llama-3-8B-Instruct	FS Sim	8	SP	0.5	1.0	0.750	0.775	0.775	0.700	0.600	0.767
Meta-Llama-3-8B-Instruct	FS Sim	4	SP	0.5	1.0	0.675	0.825	0.650	0.675	0.475	0.717
Meta-Llama-3-8B-Instruct	FS Sim	2	SP	0.5	1.0	0.800	0.675	0.625	0.625	0.525	0.700
Mistral-7B-Instruct-v0.2	FS Sim	8	SP	0.5	1.0	0.550	0.600	0.450	0.500	0.325	0.533
Mistral-7B-Instruct-v0.2	FS Sim	2	SP	0.5	1.0	0.55	0.475	0.525	0.375	0.275	0.517
Mistral-7B-Instruct-v0.2	FS Sim	4	SP	0.5	1.0	0.475	0.450	0.450	0.350	0.225	0.458
Qwen2-7B-Instruct	FS Sim	8	SP	0.5	1.15	0.675	0.625	0.650	0.575	0.500	0.650
Qwen2-7B-Instruct	FS Sim	4	SP	0.5	1.15	0.650	0.600	0.650	0.550	0.450	0.633
Qwen2-7B-Instruct	FS Sim	2	SP	0.5	1.15	0.600	0.550	0.650	0.500	0.400	0.600
RISCORE$_m$ Rand											
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Rand (Q)	8	SP	0.5	1.15	0.775	0.825	0.800	0.700	0.600	0.800
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Rand (Q)	2	SP	0.5	1.15	0.850	0.775	0.725	0.725	0.650	0.783
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Rand (Q)	4	SP	0.5	1.15	0.775	0.800	0.700	0.725	0.600	0.758
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Rand	8	SP	0.5	1.0	0.700	0.625	0.625	0.600	0.450	0.650
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Rand	2	SP	0.5	1.0	0.725	0.600	0.550	0.600	0.400	0.625
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Rand	4	SP	0.5	1.0	0.675	0.650	0.525	0.550	0.325	0.617
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Rand	4	SP	0.5	1.0	0.650	0.700	0.675	0.625	0.500	0.675
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Rand	2	SP	0.5	1.0	0.700	0.725	0.575	0.525	0.375	0.667
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Rand	8	SP	0.5	1.0	0.700	0.750	0.550	0.575	0.425	0.667
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Rand	4	SP	0.5	1.0	0.575	0.475	0.500	0.425	0.300	0.517
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Rand	2	SP	0.5	1.0	0.550	0.425	0.400	0.325	0.200	0.458
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Rand	8	SP	0.5	1.0	0.350	0.450	0.400	0.275	0.200	0.400
Qwen2-7B-Instruct	RISCORE$_m$ Rand	4	SP	0.5	1.15	0.625	0.650	0.600	0.550	0.400	0.625
Qwen2-7B-Instruct	RISCORE$_m$ Rand	2	SP	0.5	1.15	0.650	0.575	0.600	0.500	0.400	0.608
Qwen2-7B-Instruct	RISCORE$_m$ Rand	8	SP	0.5	1.15	0.625	0.550	0.600	0.450	0.325	0.592
RISCORE$_m$ Sim											
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Sim (Q)	4	SP	0.5	1.15	0.850	0.850	0.800	0.800	0.650	0.833
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Sim (Q)	8	SP	0.5	1.15	0.800	0.775	0.850	0.700	0.625	0.808
Meta-Llama-3-70B-Instruct	RISCORE$_m$ Sim (Q)	2	SP	0.5	1.15	0.850	0.775	0.725	0.725	0.600	0.783
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Sim	8	SP	0.5	1.0	0.775	0.625	0.725	0.575	0.500	0.708
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Sim	4	SP	0.5	1.0	0.675	0.750	0.700	0.600	0.425	0.708
Mixtral-8x7B-Instruct-v0.1	RISCORE$_m$ Sim	2	SP	0.5	1.0	0.725	0.675	0.625	0.675	0.475	0.675
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Sim	2	SP	0.5	1.0	0.750	0.875	0.675	0.675	0.575	0.767
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Sim	8	SP	0.5	1.0	0.775	0.800	0.700	0.775	0.650	0.758
Meta-Llama-3-8B-Instruct	RISCORE$_m$ Sim	4	SP	0.5	1.0	0.825	0.700	0.700	0.675	0.525	0.742
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Sim	4	SP	0.5	1.0	0.575	0.550	0.575	0.400	0.325	0.567
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Sim	8	SP	0.5	1.0	0.575	0.575	0.500	0.450	0.325	0.550
Mistral-7B-Instruct-v0.2	RISCORE$_m$ Sim	2	SP	0.5	1.0	0.500	0.475	0.475	0.350	0.200	0.483
Qwen2-7B-Instruct	RISCORE$_m$ Sim	8	SP	0.5	1.15	0.700	0.650	0.650	0.575	0.425	0.667
Qwen2-7B-Instruct	RISCORE$_m$ Sim	2	SP	0.5	1.15	0.700	0.625	0.675	0.550	0.375	0.667
Qwen2-7B-Instruct	RISCORE$_m$ Sim	4	SP	0.5	1.15	0.750	0.625	0.550	0.575	0.425	0.642

Table 7: Model performance for *BrainTeaser* (Part 2). The presence of (**Q**) in the method column indicates that the results correspond to the quantized version of the model.

Model	Method	Num.Ex	Task	Temp	Rep_Pen	Original	Semantic	Context	Ori. + Sem.	Ori. + Sem. + Con.	Average
						RISCORE Results					
					Llama3-70B zeroshot for QA & Llama3-8B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	8	SP	0.5	1.15	0.850	0.825	0.750	0.775	0.600	0.808
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	2	SP	0.5	1.15	0.800	0.775	0.800	0.750	0.650	0.792
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	4	SP	0.5	1.15	0.825	0.750	0.800	0.675	0.600	0.792
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	SP	0.5	1.0	0.700	0.750	0.600	0.650	0.475	0.683
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	SP	0.5	1.0	0.750	0.750	0.500	0.650	0.375	0.667
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	4	SP	0.5	1.0	0.725	0.625	0.575	0.575	0.375	0.642
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	SP	0.5	1.0	0.700	0.725	0.675	0.650	0.525	0.700
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	SP	0.5	1.0	0.650	0.675	0.700	0.550	0.425	0.675
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	SP	0.5	1.0	0.700	0.600	0.575	0.525	0.350	0.625
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	SP	0.5	1.0	0.550	0.450	0.475	0.375	0.300	0.492
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	SP	0.5	1.0	0.525	0.400	0.475	0.300	0.200	0.467
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	SP	0.5	1.0	0.425	0.525	0.475	0.400	0.325	0.475
Qwen2-7B-Instruct	RISCORE Sim	8	SP	0.5	1.15	0.600	0.700	0.625	0.550	0.425	0.642
Qwen2-7B-Instruct	RISCORE Sim	2	SP	0.5	1.15	0.650	0.650	0.575	0.550	0.400	0.625
Qwen2-7B-Instruct	RISCORE Sim	4	SP	0.5	1.15	0.600	0.650	0.625	0.550	0.450	0.625
					Llama3-70B fewshot for QA & Llama3-8B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	8	SP	0.5	1.15	0.800	0.800	0.825	0.750	0.650	0.808
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	4	SP	0.5	1.15	0.775	0.800	0.800	0.700	0.600	0.792
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	2	SP	0.5	1.15	0.800	0.725	0.725	0.725	0.600	0.750
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	SP	0.5	1.0	0.725	0.725	0.575	0.675	0.475	0.675
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	SP	0.5	1.0	0.700	0.675	0.65	0.625	0.450	0.675
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	4	SP	0.5	1.0	0.700	0.675	0.575	0.625	0.400	0.650
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	SP	0.5	1.0	0.775	0.750	0.700	0.700	0.600	0.742
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	SP	0.5	1.0	0.725	0.700	0.625	0.600	0.400	0.683
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	SP	0.5	1.0	0.725	0.625	0.625	0.550	0.450	0.658
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	SP	0.5	1.0	0.625	0.575	0.475	0.475	0.350	0.558
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	SP	0.5	1.0	0.500	0.550	0.500	0.425	0.350	0.517
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	SP	0.5	1.0	0.550	0.425	0.450	0.375	0.225	0.475
Qwen2-7B-Instruct	RISCORE Sim	4	SP	0.5	1.15	0.625	0.700	0.650	0.575	0.450	0.658
Qwen2-7B-Instruct	RISCORE Sim	8	SP	0.5	1.15	0.650	0.675	0.650	0.575	0.475	0.658
Qwen2-7B-Instruct	RISCORE Sim	2	SP	0.5	1.15	0.625	0.650	0.600	0.525	0.400	0.625
					Llama3-70B fewshot for QA & Llama3-70B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	4	SP	0.5	1.15	0.875	0.775	0.725	0.750	0.600	0.792
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	2	SP	0.5	1.15	0.775	0.825	0.750	0.775	0.675	0.783
Meta-Llama-3-70B-Instruct	RISCORE Sim (Q)	8	SP	0.5	1.15	0.775	0.750	0.775	0.725	0.600	0.767
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	SP	0.5	1.0	0.700	0.725	0.625	0.600	0.450	0.683
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	SP	0.5	1.0	0.700	0.700	0.600	0.625	0.500	0.667
Mixtral-8x7B-Instruct-v0.1	RISCORE Simm	4	SP	0.5	1.0	0.725	0.650	0.550	0.600	0.425	0.642
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	SP	0.5	1.0	0.800	0.675	0.625	0.625	0.475	0.700
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	SP	0.5	1.0	0.675	0.700	0.675	0.600	0.475	0.683
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	SP	0.5	1.0	0.725	0.650	0.625	0.550	0.475	0.667
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	SP	0.5	1.0	0.575	0.500	0.450	0.375	0.300	0.508
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	SP	0.5	1.0	0.625	0.400	0.475	0.350	0.275	0.500
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	SP	0.5	1.0	0.600	0.475	0.425	0.400	0.300	0.500
Qwen2-7B-Instruct	RISCORE Sim	2	SP	0.5	1.15	0.650	0.600	0.600	0.500	0.400	0.617
Qwen2-7B-Instruct	RISCORE Sim	4	SP	0.5	1.15	0.650	0.625	0.575	0.600	0.475	0.617
Qwen2-7B-Instruct	RISCORE Sim	8	SP	0.5	1.15	0.625	0.625	0.600	0.575	0.450	0.617

Table 8: Model performance for *BrainTeaser* (Part 3). The presence of **(Q)** in the method column indicates that the results correspond to the quantized version of the model.

Model	Method	Num.Ex	Quant	Temp	Rep_Pen	Average
Chain-of-Thought Zero-shot						
Meta-Llama-3-70B-Instruct	CoT_ZS (Q)	0	4bit	0.5	1.15	0.775
Mixtral-8x7B-v0.1	CoT_ZS	0	False	0.5	1.0	0.675
Meta-Llama-3-8B-Instruct	CoT_ZS	0	False	0.5	1.0	0.619
Mistral-7B-Instruct-v0.2	CoT_ZS	0	False	0.5	1.0	0.589
Qwen2-7B-Instruct	CoT_ZS	0	False	0.5	1.15	0.608
Few-shot with CoT Explanations						
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	2	4bit	0.5	1.15	0.789
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	4	4bit	0.5	1.15	0.783
Meta-Llama-3-70B-Instruct	CoT_FS (Q)	8	4bit	0.5	1.15	0.783
Mixtral-8x7B-v0.1	CoT_FS	8	False	0.5	1.0	0.697
Mixtral-8x7B-v0.1	CoT_FS	2	False	0.5	1.0	0.692
Mixtral-8x7B-v0.1	CoT_FS	4	False	0.5	1.0	0.686
Meta-Llama-3-8B-Instruct	CoT_FS	4	False	0.5	1.0	0.672
Meta-Llama-3-8B-Instruct	CoT_FS	8	False	0.5	1.0	0.658
Meta-Llama-3-8B-Instruct	CoT_FS	2	False	0.5	1.0	0.625
Mistral-7B-Instruct-v0.2	CoT_FS	4	False	0.5	1.0	0.603
Mistral-7B-Instruct-v0.2	CoT_FS	8	False	0.5	1.0	0.597
Mistral-7B-Instruct-v0.2	CoT_FS	2	False	0.5	1.0	0.594
Qwen2-7B-Instruct	CoT_FS	2	False	0.5	1.15	0.667
Qwen2-7B-Instruct	CoT_FS	4	False	0.5	1.15	0.656
Qwen2-7B-Instruct	CoT_FS	8	False	0.5	1.15	0.625
Few-shot with Random Selection						
Meta-Llama-3-70B-Instruct	FS Rand (Q)	4	4bit	0.5	1.15	0.800
Meta-Llama-3-70B-Instruct	FS Rand (Q)	8	4bit	0.5	1.15	0.772
Meta-Llama-3-70B-Instruct	FS Rand (Q)	2	4bit	0.5	1.15	0.769
Mixtral-8x7B-v0.1	FS Rand	4	False	0.5	1.0	0.719
Mixtral-8x7B-v0.1	FS Rand	8	False	0.5	1.0	0.711
Mixtral-8x7B-v0.1	FS Rand	2	False	0.5	1.0	0.706
Meta-Llama-3-8B-Instruct	FS Rand	2	False	0.5	1.0	0.672
Meta-Llama-3-8B-Instruct	FS Rand	8	False	0.5	1.0	0.672
Meta-Llama-3-8B-Instruct	FS Rand	4	False	0.5	1.0	0.639
Mistral-7B-Instruct-v0.2	FS Rand	2	False	0.5	1.0	0.586
Mistral-7B-Instruct-v0.2	FS Rand	4	False	0.5	1.0	0.586
Mistral-7B-Instruct-v0.2	FS Rand	8	False	0.5	1.0	0.586
Qwen2-7B-Instruct	FS Rand	8	False	0.5	1.15	0.700
Qwen2-7B-Instruct	FS Rand	2	False	0.5	1.15	0.689
Qwen2-7B-Instruct	FS Rand	4	False	0.5	1.15	0.683
Few-shot with Semantic Similarity						
Meta-Llama-3-70B-Instruct	FS Sim (Q)	4	4bit	0.5	1.15	0.817
Meta-Llama-3-70B-Instruct	FS Sim (Q)	8	4bit	0.5	1.15	0.800
Meta-Llama-3-70B-Instruct	FS Sim (Q)	2	4bit	0.5	1.15	0.792
Mixtral-8x7B-v0.1	FS Sim	2	False	0.5	1.0	0.714
Mixtral-8x7B-v0.1	FS Sim	4	False	0.5	1.0	0.692
Mixtral-8x7B-v0.1	FS Sim	8	False	0.5	1.0	0.675
Meta-Llama-3-8B-Instruct	FS Sim	4	False	0.5	1.0	0.711
Meta-Llama-3-8B-Instruct	FS Sim	2	False	0.5	1.0	0.706
Meta-Llama-3-8B-Instruct	FS Sim	8	False	0.5	1.0	0.681
Mistral-7B-Instruct-v0.2	FS Sim	4	False	0.5	1.0	0.633
Mistral-7B-Instruct-v0.2	FS Sim	8	False	0.5	1.0	0.611
Mistral-7B-Instruct-v0.2	FS Sim	2	False	0.5	1.0	0.608
Qwen2-7B-Instruct	FS Sim	8	False	0.5	1.15	0.731
Qwen2-7B-Instruct	FS Sim	2	False	0.5	1.15	0.722
Qwen2-7B-Instruct	FS Sim	4	False	0.5	1.15	0.714

Table 9: Model Performance for *RiddleSense* (Part 1). The column **Quant** indicates whether the model is quantized or not.

Model	Method	Num.Ex	Quant	Temp	Rep_Pen	Average
RISCORE Results						
Llama3-70B Fewshot for QA & Llama3-70B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim	2	True	0.5	1.15	0.792
Meta-Llama-3-70B-Instruct	RISCORE Sim	8	True	0.5	1.15	0.789
Meta-Llama-3-70B-Instruct	RISCORE Sim	4	True	0.5	1.15	0.783
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	False	0.5	1.0	0.700
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	4	False	0.5	1.0	0.689
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	False	0.5	1.0	0.672
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.722
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.708
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.692
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	False	0.5	1.0	0.600
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	False	0.5	1.0	0.600
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	False	0.5	1.0	0.597
Qwen2-7B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.731
Qwen2-7B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.717
Qwen2-7B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.697
Llama3-70B Fewshot for QA & Llama3-8B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim	4	True	0.5	1.15	0.789
Meta-Llama-3-70B-Instruct	RISCORE Sim	2	True	0.5	1.15	0.786
Meta-Llama-3-70B-Instruct	RISCORE Sim	8	True	0.5	1.15	0.775
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	False	0.5	1.0	0.719
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	False	0.5	1.0	0.689
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	4	False	0.5	1.0	0.686
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.706
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.686
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.681
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	False	0.5	1.0	0.617
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	False	0.5	1.0	0.606
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	False	0.5	1.0	0.603
Qwen2-7B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.719
Qwen2-7B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.697
Qwen2-7B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.681
Llama3-8B Fewshot for QA & Llama3-8B for distractors						
Meta-Llama-3-70B-Instruct	RISCORE Sim	8	True	0.5	1.15	0.806
Meta-Llama-3-70B-Instruct	RISCORE Sim	2	True	0.5	1.15	0.792
Meta-Llama-3-70B-Instruct	RISCORE Sim	4	True	0.5	1.15	0.778
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	4	False	0.5	1.0	0.714
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	8	False	0.5	1.0	0.689
Mixtral-8x7B-Instruct-v0.1	RISCORE Sim	2	False	0.5	1.0	0.681
Meta-Llama-3-8B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.700
Meta-Llama-3-8B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.689
Meta-Llama-3-8B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.686
Mistral-7B-Instruct-v0.2	RISCORE Sim	8	False	0.5	1.0	0.614
Mistral-7B-Instruct-v0.2	RISCORE Sim	4	False	0.5	1.0	0.600
Mistral-7B-Instruct-v0.2	RISCORE Sim	2	False	0.5	1.0	0.589
Qwen2-7B-Instruct	RISCORE Sim	2	False	0.5	1.0	0.694
Qwen2-7B-Instruct	RISCORE Sim	8	False	0.5	1.0	0.689
Qwen2-7B-Instruct	RISCORE Sim	4	False	0.5	1.0	0.683

Table 10: Model Performance for *RiddleSense* (Part 2). The column **Quant** indicates whether the model is quantized or not.