Large Language Models Exhibit Limited Reasoning Ability on Coding Problems

Jinyoung Jo
Stanford University
jinyoungjo@stanford.edu

Jonah Engelmann
Santa Clara University
joengelmann@scu.edu

Sean Choi
Santa Clara University
sean.choi@scu.edu

Abstract

Claims that large language models (LLMs) have complex reasoning ability have stirred broad interests, and controversies, of academics and non-academics alike. A popular basis for such claims comes from LLMs' ability to solve coding questions, which involves understanding the question statement and providing code that solves the question. Although such abilities are remarkable feats worth praising, we argue that they come from memorization rather than reasoning. We first show that LLMs' question-solving ability degrades with increased recency of the question, likely due to the reduced amount of training data for more recent questions. Additionally, we show that an LLM often fails to solve the question when presented with rephrased but equivalent question statements, further suggesting their limited reasoning ability.

1 Introduction

The development of large language models (LLMs) has substantially advanced natural language processing, enabling various applications across multiple domains. Their remarkable performance on diverse tasks has sparked significant research interest in their reasoning abilities. While LLMs are often claimed to possess reasoning abilities, it still remains a controversy (Huang and Chang, 2023; Mondorf and Plank, 2024). Although some previous studies provide evidence that LLMs are able to reason (Shi et al., 2022; Suzgun et al., 2023; Lampinen et al., 2024; Saparov and He, 2023), others show that their ability to reason is limited (Arkoudas, 2023; Yang et al., 2023; Kambhampati et al., 2024; McCoy et al., 2023; Valmeekam et al., 2023; Razeghi et al., 2022; Hao et al., 2025; Xu et al., 2025). Some studies show evidence for both the true ability to reason and reliance on shallow heuristics (Prabhakar et al., 2024). Thus, it remains unclear whether the behavior of these systems is based on true reasoning or superficial heuristics such as memorization or recognition of surface-level patterns.

In this study, we evaluate LLMs' reasoning ability on solving coding questions, which inherently require reasoning skills. Code generation, where models assist with generating and completing code, is one of the most popular applications of LLMs (Jiang et al., 2024), with the promises of AI-generated code improving coding efficiency. While previous research has shown that LLMs can generate code, often more efficiently than humans (Coignion et al., 2024), it has rarely been investigated whether this ability stems from genuine reasoning or merely from retrieving patterns from training data. Coding questions provide an effective test case for this distinction, as they require reasoning skills, yet many widely used question sets are likely to have been included in training data. This allows us to assess whether LLMs' performance reflects true reasoning or pattern recognition. If LLMs exhibit lower accuracy on prompts that are unlikely to have appeared in their training data, while performing well on otherwise similar questions, this would suggest that their performance is primarily dependent on memory retrieval rather than reasoning.

To test this, we conducted two analyses. First, we investigated the change in model performance as a function of the time point at which the question was entered into the coding question database. If models perform better with older questions, which are more likely to have been included in the training data, compared to more recent questions with similar difficulty, the finding would suggest that the model tend to recite from memory rather than use reasoning skills. Second, we investigated the effect of prompt perturbation, by replacing one to three keywords with their synonyms and testing whether the model performance changes. If model performance on code generation is susceptible to word-level perturbation in the prompts, as often reported in previous studies (Wang et al., 2023; Qiang et al., 2024; Zhu et al., 2024; Zhuo et al., 2023), the finding would suggest that models perform well only on word sequences they have encountered during training, rather than applying reasoning skills to answer coding questions. We also examine the effect of word frequency on performance under perturbation. Our results from both analyses demonstrate that models respond differently to questions that are essentially the same but phrased differently, suggesting that LLMs have limited reasoning ability.

2 Methods

2.1 Dataset Specification

The evaluation data set consists of coding questions from a popular repository called LeetCode[1]. Each LeetCode question has a set of associated attributes: question ID, difficulty, test cases, number of submissions, and number of acceptances. Question IDs are numerical values assigned automatically by LeetCode, indicating the order in which questions were added to the database. A higher ID corresponds to a more recently added question. The difficulty level is one of Easy, Medium and Hard, and is a manually labeled identifier that provides an estimate of the difficulty of each question. Test cases are input–output pairs used to verify the correctness of the provided solution. A solution is considered correct or accepted only if it produces the expected output for every given input, thereby passing all test cases. Number of submissions is the number of attempted solutions submitted by the users, and the number of acceptances are the subset of such submissions that pass all the test cases. A sample of the question and test cases is presented in Appendix A.

After removing questions for which fewer than three test cases were available, the dataset collected for the present study consisted of 1191 questions (Easy N=309, Medium N=572, Hard N=310).

2.2 Model Specifications

The model we used for the main experiment reported below was GPT-4o (OpenAI et al., 2024), where we investigated the model accuracy as a function of question ID on the full set of 1191 questions described above. Before conducting the main experiment, we tested five models including GPT-4o on a smaller subset of 90 questions (Easy N=30, Medium N=30, Hard N=30): GPT-3.5, GPT-4o (OpenAI et al., 2024), Gemini 1.5 (Gemini Team et al., 2024), Llama 3 70B (Grattafiori et al., 2024), and StarCoder 2 15B (Li et al., 2023). This selection ensures coverage of key model variables such as the model size, whether the model is open source or closed source, and whether the model is vanilla or fine-tuned. Among these models, GPT-4o exhibited the highest accuracy (see Appendix B for a comparison between models); however, results from GPT-4o on the larger dataset and those from the other models on the smaller dataset yielded similar and consistent conclusions. We report the results from GPT-4o in Section 3 and include the results from other models in Appendix B.

2.3 Prompt Perturbation Analysis

We examined how modifying certain words within a prompt affects the output quality of an LLM, particularly GPT-4o. Specifically, we identified one to three keywords in each prompt and replaced them with each of their synonyms using two different methods, outlined below. We assessed the ability of GPT-4o to generate code for these modified prompts.

In the first method, we asked GPT-4o to identify keywords in each prompt and replace each one with an appropriate synonym, without relying on any external libraries or databases. An example of prompt perturbation using GPT-4o is presented in Appendix C. For this method, we used a subset of questions (N=750 out of 1191), for which GPT-4o passed all test cases with the original, unmodified prompt.

In the second method, we extracted a set of keywords in each prompt using keyBERT (Grootendorst, 2020), a model that provides a list of keywords, each with a score that measures how important the word is in the prompt. Next, for each selected keyword, we retrieved a list of synonyms from the WordNet database (Princeton University, 2010) via the NLTK package (Bird et al., 2009). We then manually selected the most appropriate synonym for each keyword, i.e., the one that preserved the original meaning as closely as possible. This was to test whether the results obtained using GPT-4o in the first method hold when synonyms were manually chosen. For this manual replacement, we used a smaller data set of 25 questions. The results of this manual analysis closely aligned with those from the LLM-based synonym replace-

[1]https://leetcode.com

ment. We present the results of the first method in Section 3 and those of the manual replacement in Appendix D.

In both methods described above, we performed three levels of analysis, replacing the highest scoring (the most important), the top two, and the top three keywords with their synonyms, as determined by GPT-4o or keyBERT. This allowed us to examine changes in model performance as a function of the number of keywords replaced.

Further, we obtained the word frequencies of the original keywords and their synonyms from SUBTLEXus (Brysbaert and New, 2009) to examine the effect of relative frequency between keywords and their synonyms on model performance. If a word does not appear on the SUBTLEXus frequency list, the frequency was regarded as zero.

2.4 Evaluation Metric

Model performance is evaluated on whether or not each generated output code passes all test cases. For a given model m, question p and query q, correctness of the model and question pair for the given query is defined as:

$$C_{m,p}(q) = \begin{cases} 1, & m\text{'s solution for } p \text{ generated} \\ & \text{from } q \text{ passes all test cases} \\ 0, & \text{Otherwise.} \end{cases}$$

Given $C_{m,p}(q)$, the accuracy A is defined as

$$A_m(q) = \frac{\sum_{p \in P} C_{m,p}(q)}{|P|}$$

where accuracy for a given model m for a set of question P varies for input query q.

3 Results & Discussion

3.1 Recency effect on accuracy

We first evaluated how the performance of GPT-4o changes as a function of the time point at which a question was entered into LeetCode. Figure 1 presents correctness of the model's output as a function of question ID, divided by difficulty. The overall accuracy was 0.63; not surprisingly, the accuracy was highest in Easy questions (0.77), intermediate in Medium (0.64) and lowest in the Hard category (0.48). Crucially, in Medium and Hard questions, we see an effect of question ID such that the model tends to struggle more often with higher-numbered (more recent) questions. This finding demonstrates that the model performs better with older questions, which are more likely to have been included in the training data, suggesting that the question solving ability is likely based on memorization rather than true reasoning.

3.2 Prompt perturbation effect on accuracy

Next, we evaluated changes in model performance under prompt perturbation. Figure 2 presents the correctness of GPT-4o outputs as a result of prompt perturbation. By design, the accuracy in the Original Prompt condition is 1, since only the original prompts that the model correctly solved were included in our prompt perturbation analysis, as described in Section 2.3. Crucially, replacing a single keyword with its synonym already led to a drop in accuracy (0.93), with a slight further decline as more keywords were replaced (0.909 when two words were replaced; 0.907 when three were replaced). Within Easy questions, the number of correct solutions decreased from 227 to 226 and then to 225 across the 1-, 2-, and 3-word replacements. For the Medium category, the number of correct solutions was 341 when one word was replaced, which decreased to 330 when two words were replaced; interestingly, it increased slightly to 334 when three were replaced. For Hard questions, the number of correct solutions was 126 in both 1- and 2-word replacements, which decreased to 121 in the 3-word replacement condition. We found similar results with manual perturbation (see Appendix D).

In the 1-word replacement condition, we also examined how the relative frequency between the original word and its synonym affected model performance. The results demonstrated that the effect of relative frequency was minimal. Among the 750 questions analyzed, 65 had identical frequencies for both words and were excluded from the analysis. For the remaining 685 questions, the new word (synonym) had a higher frequency in 346 cases and a lower frequency in 339 cases, compared to the original word. When the new word was more frequent, the accuracy dropped from 1 to 0.95 after synonym replacement. When the new word had a lower frequency, the accuracy dropped from 1 to 0.92. These results suggest that changes in frequency in either direction had a comparable impact on performance.

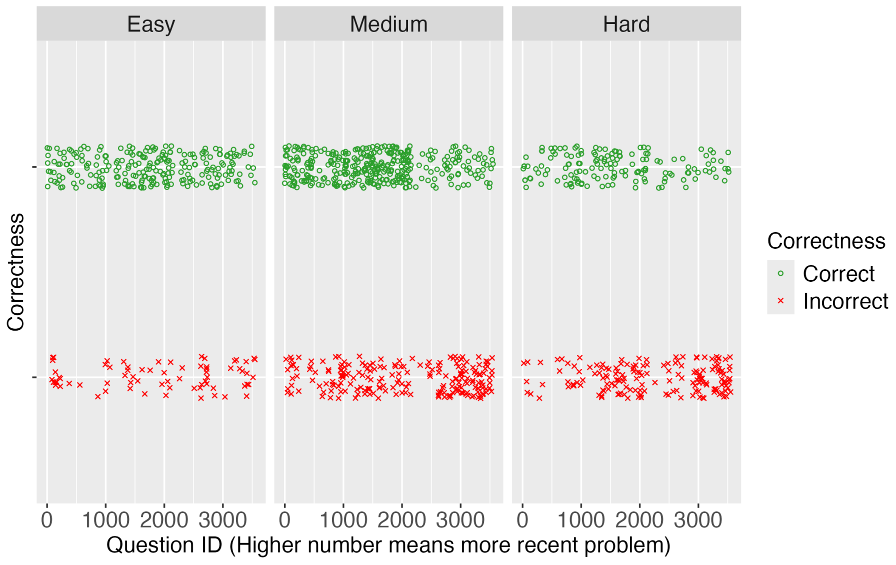

Figure 1: Correctness of GPT-4o outputs as a function of question ID, grouped by difficulty level

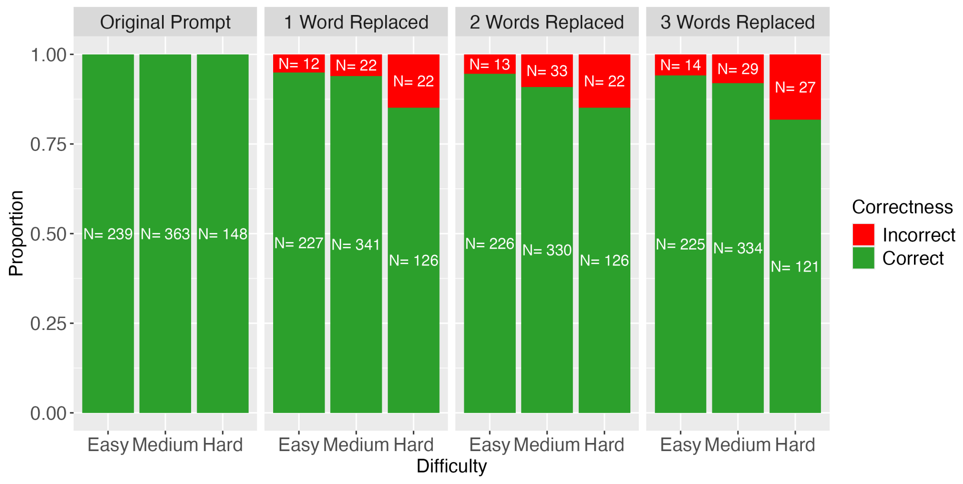

Figure 2: Correctness of GPT-4o outputs, grouped by the number of keywords replaced and difficulty level

3.3 Summary & Discussion

In summary, likely due to memorization from training data, LLMs exhibit decreasing accuracy on more recent questions and when a small portion of the prompts are perturbed, suggesting a reliance on surface-level patterns rather than true reasoning. Our findings are consistent with those of previous studies (Gonen et al., 2023; Razeghi et al., 2022) indicating that the more frequently the prompt appears in the training data, the more familiar the model is with it, leading to improved model performance. Similarly, numerous studies have reported a close relationship between pretraining data and task performance (e.g. Elazar et al. 2023; Kandpal et al. 2023). The issue of data contamination has been raised in studies that tested the ability of LLMs to solve LeetCode questions (Coignion et al., 2024; Xia et al., 2025), and our study shows that the model performance indeed degrades when tested on more recent questions.

4 Conclusion

We argue that LLMs are, by design, autoregressive word sequence predictors and that there is nothing inherent in them that enables reasoning. LLMs can only simulate reasoning by learning statistical patterns in large amounts of data and applying learned patterns in a way that appears logical. This work provides compelling evidence to this argument, showing that LLMs struggle with prompts less likely included in the training data and fail to solve the simple questions with subtle changes in wording. We believe that this work will motivate future research to improve LLMs' question solving ability via data augmentation using prompt perturbation and improved LLM architectures.

Acknowledgment

We are grateful to the anonymous reviewers for their helpful comments. We are also grateful to Graham Kim for helping with expanding the experimental data set for the camera ready version. This research is funded in part by Santa Clara University's Faculty-mentored Undergraduate Research Support (2FURS) to support Engelmann's undergraduate summer research. This article solely reflects the opinions and conclusions of its authors.

Limitations

One of the main limitations of this work is that the experiment is performed only on English query and result pairs. Although the experimental methods are not limited to English, models trained with other languages may or may not perform better than models trained mostly on English. Another limitation is the restricted set of models evaluated. Although this work evaluates the reasoning capabilities of many popular LLMs, it does not cover the full range of available models, mainly due to the ever increasing number of LLMs coming into the market. In addition, our work does not evaluate specialized reasoning models that employ Chain-of-Thought (CoT) reasoning. We argue that solving LeetCode questions may not be greatly influenced by CoT due to the nature of LeetCode questions asking for a one-step solution. Nevertheless, this claim is worth investigating in future work. In addition, this study examines performance based on a single LLM query, whereas CoT reasoning submits multiple LLM queries to achieve the desired outcome. Therefore, the result of the present work shows limited reasoning abilities of the models when a single query is submitted and evaluated.

Ethics Statement

We do not anticipate an immediate ethical or societal impact resulting from our work. However, we acknowledge that LeetCode is a widely used tool in the industry to assess interview candidates' programming aptitude. Thus, if the content and the result of this work is maliciously used, it can potentially lead to plagiarism on programming interviews. In addition, it is well known that LLMs hallucinate, i.e., generate invalid answers that seem correct. Thus, code generation via the approaches mentioned in this paper can also result in hallucinations, which may lead to unforeseen issues if the code is used directly in real-world applications.

References

Konstantine Arkoudas. 2023. Gpt-4 can't reason.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

M. Brysbaert and B New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods*, 41:977–990.

Tristan Coignion, Clément Quinton, and Romain Rouvoy. 2024. A performance study of llm-generated code on leetcode. In *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, EASE 2024, page 79–89. ACM.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. 2023. Measuring causal effects of data statistics on language model's 'factual' predictions.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.

Hila Gonen, Srini Iyer, Terra Blevins, Noah Smith, and Luke Zettlemoyer. 2023. Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10136–10148, Singapore. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. 2024. The llama 3 herd of models.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Yunzhuo Hao, Jiawei Gu, Huichen Will Wang, Linjie Li, Zhengyuan Yang, Lijuan Wang, and Yu Cheng. 2025. Can MLLMs Reason in Multimodality? EMMA: An Enhanced MultiModal ReAsoning Benchmark.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Llms can't plan, but can help planning in llm-modulo frameworks.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge.

Andrew K Lampinen, Ishita Dasgupta, Stephanie C Y Chan, Hannah R Sheahan, Antonia Creswell, Dharshan Kumaran, James L McClelland, and Felix Hill. 2024. Language models, like humans, show content effects on reasoning tasks. *PNAS Nexus*, 3(7):pgae233.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, et al. 2023. Starcoder: may the source be with you!

R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. 2023. Embers of autoregression: Understanding large language models through the problem they are trained to solve.

Philipp Mondorf and Barbara Plank. 2024. Beyond accuracy: Evaluating the reasoning behavior of large language models - a survey. In *First Conference on Language Modeling*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, et al. 2024. Gpt-4 technical report.

Akshara Prabhakar, Thomas L. Griffiths, and R. Thomas McCoy. 2024. Deciphering the factors influencing the efficacy of chain-of-thought: Probability, memorization, and noisy reasoning.

Princeton University. 2010. About WordNet.

Yao Qiang, Subhrangshu Nandi, Ninareh Mehrabi, Greg Ver Steeg, Anoop Kumar, Anna Rumshisky, and Aram Galstyan. 2024. Prompt perturbation consistency learning for robust language models. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1357–1370, St. Julian's, Malta. Association for Computational Linguistics.

Yasaman Razeghi, Raja Sekhar Reddy Mekala, Robert L Logan Iv, Matt Gardner, and Sameer Singh. 2022. Snoopy: An online interface for exploring the effect of pretraining term frequencies on few-shot LM performance. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 389–395, Abu Dhabi, UAE. Association for Computational Linguistics.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2022. Language models are multilingual chain-of-thought reasoners.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, Xueqian Wang, Peilin Zhao, and Dacheng Tao. 2023. Are large language models really robust to word-level perturbations?

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. 2025. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms.

Xinnuo Xu, Rachel Lawrence, Kshitij Dubey, Atharva Pandey, Fabian Falck, Risa Ueno, Aditya Nori, Rahul Sharma, Amit Sharma, and Javier González. 2025. Re-imagine: Symbolic benchmark synthesis for reasoning evaluation. In *ICLR 2025 - Workshop on Reasoning and Planning for LLMs*.

Zhun Yang, Adam Ishay, and Joohyung Lee. 2023. Coupling large language models with logic programming for robust and general reasoning from text. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5186–5219, Toronto, Canada. Association for Computational Linguistics.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Gong, and Xing Xie. 2024. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *Proceedings of the 1st ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, LAMPS '24, page 57–68, New York, NY, USA. Association for Computing Machinery.

Terry Yue Zhuo, Zhuang Li, Yujin Huang, Fatemeh Shiri, Weiqing Wang, Gholamreza Haffari, and Yuan-Fang Li. 2023. On robustness of prompt-based semantic parsing with large pre-trained language model: An empirical study on codex. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1090–1102, Dubrovnik, Croatia. Association for Computational Linguistics.

A Question Structure

Each LeetCode question has three parts: question description, a set of constraints and code snippet to guide the answer generation. Here is a sample question.

> *Description:*
>
> Special Positions in a Binary Matrix. Given an $m \times n$ binary matrix `mat`, return the number of special positions in `mat`.
>
> A position `(i, j)` is called special if `mat[i][j] == 1` and all other elements in row `i` and column `j` are `0` (rows and columns are 0-indexed).
>
> *Constraints:*

```
m == mat.length
n == mat[i].length
1 <= m, n <= 100
mat[i][j] is either 0 or 1.
```

> *Code Snippet:*

```
def numSpecial(mat):
    """
    :type mat: List[List[int]]
    :rtype: int
    """
```

Given such question template, the prompt involves prepending the following statement:

> *Write Python code to solve the following coding problem*

To test whether the generated code is correct, each question also has test cases with IDs that specify a set of inputs and expected outputs.

ID	Input	Output
1	`[[[1,0,0],[0,0,1],[1,0,0]]]`	1
2	`[[[1,0,0],[0,1,0],[0,0,1]]]`	3
3	`[[[1,1,0],[0,0,0],[0,0,0]]]`	0

If the generated code is correct, the code should produce the correct output for all inputs in a reasonable amount of time.

B Evaluation results with additional models (Figure 3)

- Accuracy by model: GPT-4o exhibited the highest accuracy (0.79), followed by Llama 3 (0.64), GPT-3.5 (0.59), StarCoder (0.46), and Gemini 1.5 showed the lowest accuracy (0.41).
- Accuracy by difficulty: the accuracy of output was highest among the Easy questions (0.82), intermediate among the Medium (0.59), and lowest among the Hard ones (0.33).

C Prompt Perturbation Example

An example of prompt perturbation is as follows. Consider the following original prompt:

> There are n **people** and 40 types of **hats** labeled from 1 to 40. Given a 2D integer array **hats**, where **hats**[i] is a list of all **hats** preferred by the i th person. Return the number of **ways** that n **people** can wear different **hats** from each other. Since the answer may be too large, return it modulo 10 9 + 7.

The words 'hats', 'people', and 'ways' were identified as keywords (boldfaced above), and the synonyms of these words are selected as 'headwear', 'individuals', and 'methods'.

D Manual prompt perturbation results (Figure 4)

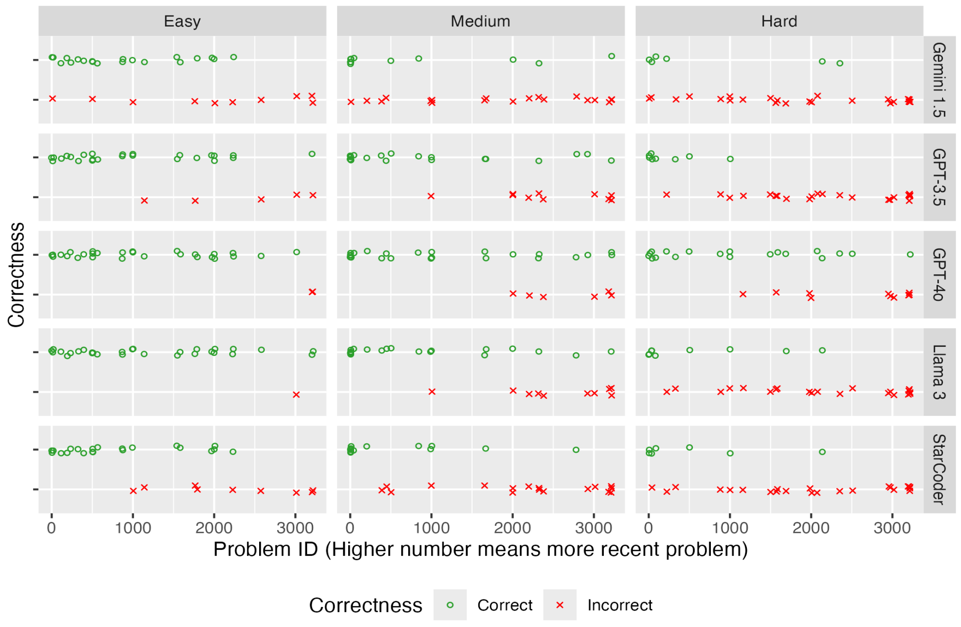

Figure 3: Correctness of model outputs as a function of Question ID, grouped by difficulty level and LLM type

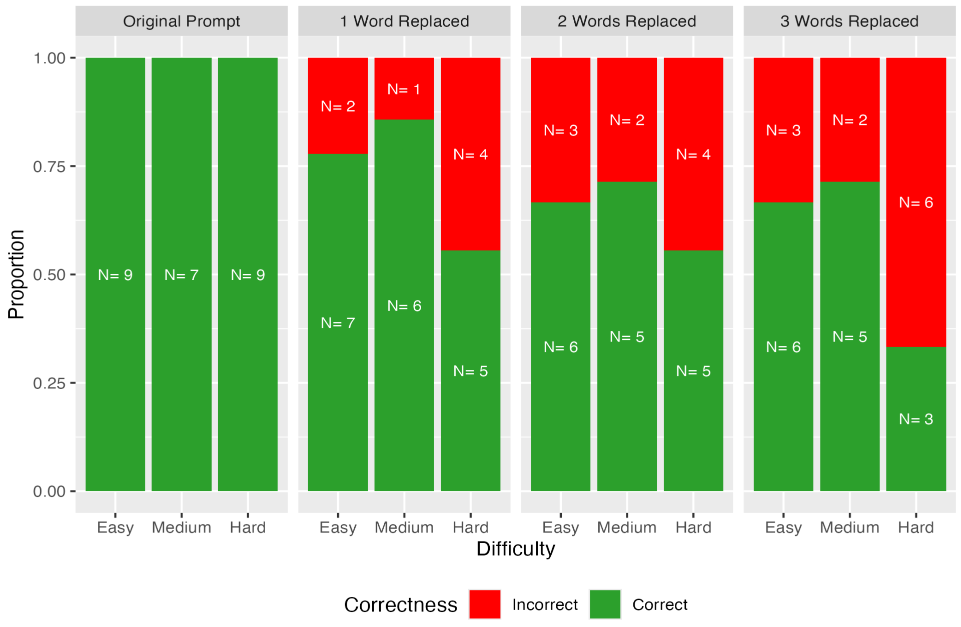

Figure 4: Correctness of GPT-4o outputs grouped by the number of keywords replaced and difficulty level in the manual prompt perturbation analysis