In-Context Learning for LLMs: SDS 410 Research Paper

Kat Brady, Kimberly By Goytia, Yelena Davidson, Hau Phan, Lucia Qin, and Nicole Sanchez-Flores *

May 8, 2026

Abstract

Large language models are widely used for tasks that require rule inference, structured reasoning, and constrained behavior, but their reliability in these settings is still not well understood. We evaluate three GPT-5 models (GPT-5, GPT-5-mini, GPT-5-nano) across three tasks: a Truth-or-Dare turn-taking setup, two rule-inference experiments (character substitution and palindrome induction), and a relational reasoning task over fictional social scenarios.

Performance declines with both model scale and task ambiguity, and the failures are systematic. In rule-inference tasks, models tend to commit to the simplest rule consistent with the examples instead of abstaining when the rule is underdetermined. In relational reasoning, models struggle with inverse retrieval, conflicting constrains, and multi-step dependencies, often collapsing to single-actor explanations. Overall, the results suggest that models default to low-complexity rules without checking alternatives, and that confidence does not track competence, all of which highlight key limitations for real-world usage.

Keywords:: In-Context Engineering, LLMs, Model Failures

*Department of Statistical and Data Sciences, Smith College. We thank Professor Nikko Stevens, Dr. Ryan Lingo, and the Team at Honda Research Labs for their support throughout our project.

1 Introduction

Large language models are increasingly used in settings where their outputs are treated as reliable: pulling facts from context, inferring rules from examples, navigating social and relational information, and maintaining consistent behavior across a conversation. What most of these use cases have in common is that the model is not being retrained — it is being asked to adapt to a new task at inference time, using only the information supplied in its prompt. This ability, commonly referred to as in-context learning, is one of the central capabilities that has driven the practical adoption of modern LLMs (Brown et al.). The broader practice of structuring prompts, examples, memory, and retrieved information to make this adaptation work reliably — what recent work has begun to call context engineering — has become one of the main levers practitioners have for shaping model behavior without touching model weights (Mäkinen).

We are interested in the structure of the mistakes LLMs make when they are wrong, and how those mistakes interact with the way a task is framed in context. Our guiding question is where in-context learning succeeds, where it breaks, and whether the breakages follow recognizable patterns across different kinds of tasks.

In this paper, we evaluate three models from OpenAI's GPT-5 family — GPT-5, GPT-5-mini, and GPT-5-nano — on four test cases designed to surface distinct in-context failure modes rather than to produce a single headline accuracy number. The first examines how models manage turn-taking and instruction-following in a Truth-or-Dare game, including whether a markdown-based persona configuration (soul.md) supplied in context changes their behavior. The second asks models to infer character-level substitution rules from a small number of in-context examples and apply them to a new sentence. The third extends this rule-inference setup to palindrome-based tasks where the provided examples are, in some conditions, genuinely insufficient to identify a unique rule, and the correct response is to abstain. The fourth places models in fictional friend-group and family scenarios containing conflicting norms, vague relationships, and multi-character situations supplied entirely

through context, to probe how they reason under messiness that more closely resembles real-life relational dynamics.

Each of these tasks isolates a different way in-context learning can go wrong. Rule induction from examples can fail through tokenization losses or through the model committing too quickly to the simplest hypothesis consistent with the demonstrations. Ambiguous in-context examples can fail through the model refusing to signal uncertainty even when the prompt explicitly permits it. Rich in-context worlds with conflicting rules can fail through oversimplification or through an inability to reason across multiple constraints at once. Persona configurations supplied in context can fail to persist across turns. Studied in isolation, each of these failures could be read as a one-off quirk of a particular benchmark. Studied together, they begin to suggest something more general about how current models use the information we put in their context window — and where additional context helps, where it is ignored, and where the way a task is framed determines whether the model ever considers alternatives to its first guess.

We report results by model, by task difficulty, and by failure type, and we note where model confidence tracks model correctness and where it does not. Our findings are broadly consistent with the view that scale helps but does not eliminate these failure modes, and that many of the mistakes we observe are systematic rather than random — which is the relevant property for anyone trying to engineer around them. The remainder of the paper reviews the relevant literature on in-context learning and context engineering, describes our methods for each test case in detail, presents results, and closes with a discussion of what these patterns imply for future research directions.

2 Background

Recent research on large language models and their behavior has increasingly focused on how models can adapt to new tasks at inference time rather than through task-specific

retraining. One major example is GPT-3, a 175-billion-parameter autoregressive language model that demonstrated strong zero-shot, one-shot, and few-shot performance across a wide range of tasks (Brown et al.). These findings suggest that scale substantially improves task-agnostic generalization, allowing models to perform translation, arithmetic, and reasoning tasks from examples embedded directly in the input rather than from gradient-based fine-tuning. However, these gains do not eliminate important limitations: the model still struggles with some forms of logical inference and raises concerns about contamination from internet-scale training data (Brown et al.).

Later work examined the mechanisms that may underlie this apparent adaptability. In-context learning in transformers has been shown, in linear regression settings, to implicitly implement recognizable learning procedures such as gradient descent and ridge regression without updating model parameters (Akyürek et al.). These findings suggest that in-context learning can function as an algorithmic process in which the model encodes and updates internal representations during prediction. Essentially, it prevents the model from mimicking an answer without understanding the underlying rules. As model depth and hidden size increase, transformers move through distinct algorithmic regimes and eventually approach the behavior of Bayesian estimators. Probing experiments further indicate that models store intermediate quantities such as moment matrices and weight vectors while making predictions (Akyürek et al.).

More recent scholarship has expanded this discussion into the broader context of the engineering framework. Context engineering can be understood as the systematic design of methods that improve how large language models manage information during inference (Mäkinen). In this framework, context includes not only the visible prompt window, but also memory systems, model parameters, and external databases. This broader framing reflects the practical challenges of using LLMs in complex environments, especially the quadratic growth in computation and memory demands as context length increases, and the "lost-in-the-middle" problem, where models fail to use information in the center of long inputs

(Mäkinen). Prompt engineering and retrieval-augmented generation have been identified as particularly practical approaches that improve performance without requiring expensive retraining (Mäkinen). The context-engineering approach, therefore, focuses more on the structure, retrieval, and management of information related to the model's inputs.

Additionally, one of the most influential approaches within this broader context-engineering literature is Retrieval-Augmented Generation, or RAG. This new framework combines parametric memory in a pre-trained sequence-to-sequence model with non-parametric memory in an external Wikipedia index (Lewis et al.). Instead of relying solely on what the model has stored internally, RAG uses a neural retriever to identify relevant documents, and then conditions generation on the retrieved information. This architecture is significant for several reasons. First, it allows knowledge to be updated by replacing the document index rather than retraining the full model. Second, because retrieved documents remain human-readable, the system provides stronger provenance and interpretability than purely parametric models. Third, experiments show that RAG produces language that is more factual, specific, and diverse than parametric-only baselines, while also establishing state-of-the-art performance on open-domain question answering (Lewis et al.). The RAG approach essentially demonstrates that combining parametric and nonparametric memory can improve both accuracy and interpretability in knowledge-intensive tasks.

Overall, this literature suggests that large language model behavior emerges from an interaction between pretraining and inference-time support. Examples based in context, retrieval mechanisms, and broader context-engineering strategies all influence how well a model can adapt to a task and generate reliable outputs (Brown et al.; Mäkinen; Lewis et al.).

3 Methods

3.1 Truth or Dare

The type of models we used are GPT-5-mini, GPT-5-nano and GPT-5. Note the difference between each model. In comparison to GPT-5-mini, GPT-5 in the current literature is said to be the highest reasoning of the models we are using, moderate speed but the highest cost (Nigel et al., 2025). In general, GPT-5 is the flagship model of OpenAI, it tends to have a high performance, but it is more costly and has the slowest speed out of all the tested models. In combinations with these model differences we will have the added variable of soul.md. Soul.md is a markdown file that configures a model's framework to have a core identity, personality and values. We tested the models using zero shot prompts which are simple prompts without examples. Through these prompts, we created standardized questions and responses for the model. These questions consist of truth question, dare question, truth or dare question, and responses if the model breaks. The dare questions include: "Write a short poem," "Describe your perfect day", "Write a short song", "Describe your perfect day", "Write a short song"and "Draw a bunny", "Write rap lyrics, "Draw a flower" and "Describe your worst day". The truth questions includes: "What's your favorite color?", "What's your hobby?", "What's your favorite animal?", "What do you like about yourself?", "Who would you want to be?", "What's your favorite season?", "Who is your favorite celebrity?". We tested each model with 15 questions consisting of asking the model truth or dare and any truth or dare questions for 5 trials. These questions vary depending on the model's truth or dare choice. To measure the accuracy of the model on truth or dare prompts, we measured the proportion of turns the model followed correctly by the number of questions the user asked. Additionally, we also wrote comments on the models' performance for each trial for qualitative findings.

3.2 Parts

3.2.1 Rule-Based Substitution Experiment

This test case explores how LLMs handle prompts that require active examination of specific parts of those prompts. LLMs do not process prompts in the same way that humans would. It is well-established that LLMs convert prompts into a series of tokens, which are then assigned unique integers for identification, and finally converted into vectors, which are what the neural networks are able to process. Although this process works efficiently for many tasks, information can be lost in the process of encoding and decoding.

In order to better understand this potential problem, a prompting procedure was developed. Three different models of OpenAI's Generative Pre-trained Transformer (GPT) were used: GPT 5, GPT 5 Mini, and GPT 5 Nano. Each of these models were to be instructed to guess a specific, language-based rule, often involving substitution of letters, deletion of letters, or similar modifications. Then, the model would be given five examples following the rule, containing a sentence before and after the rule was applied, and an unaltered sentence. The model was then to guess the rule in one line, and apply said rule to the unaltered sentence that was provided.

A total of 25 different rules, or cases, were created and then given in prompt to each of the three models, resulting in 75 data points. These prompts were assessed on the expected level of difficulty for the model to handle, ranging from 1 to 3. Prompts rated a difficulty of 1 contained only one type of substitution, substituting only one character, and generally referred to a broader subject than more complex categories, such as referencing altering vowels rather than a seemingly random group of letters. Prompts rated a 2 are more difficult. They may either contain one substitution on a more complex category, or two substitutions on a simpler category. Prompts rated a 3 were the most difficult. They contained a substitution of several characters or a string, or generally multiple substitutions on a more complex category.

The results were manually evaluated on a number of metrics focused on the model's suggested rule, suggested sentence transformation, and overall response. Correctness of the rule, correctness of the answer, and correctness of the formatting were all assigned either a 0 or 1, with 0 meaning that the model was incorrect in that category, and 1 indicating correctness. A rule was considered correct if it matched the content of the actual rule completely, although wording was not required to be identical. A transformed sentence was considered correct only if it matched the expected sentence output entirely. Even one small error disqualified an answer from being considered correct. As the models were initially instructed to provide a one line answer for both rule and sentence, the models were considered to have correct formatting if they did so. Models that provided anything other than a two line or two sentence response were said to have incorrect formatting.

Then, the model's total score on the previous three metrics was summed, giving models a score from 0 to 3 on overall correctness. The models were also evaluated on a strict pass rate, which means that models that had all 3 metrics correct were given a 1, and models that passed less than 3 of the metrics were given a 0. Finally, the models were evaluated for their certainty and given a 1 if the model did not express doubt, refuse to answer, or question its own correctness, and given a 0 if doubt, refusal, or concern was present. Ultimately, these categories provided a means by which to evaluate the success of the models on the twenty-five different cases. The results of this process can be found under the Parts subsection of the Results section.

3.2.2 Palindrome Rule-Inference Experiment

Survey design: I evaluated three models (gpt-5, gpt-5-mini, gpt-5-nano) on a rule-induction task designed to probe two distinct capabilities: inferring a transformation rule from a small number of input-output examples, and recognising when the provided examples are insufficient to identify a unique rule. Each trial gives the model a fixed set of demonstrations followed by a single test input; the model must state the inferred rule and produce

the output, or say "cannot be determined" if the rule is underdetermined.

[link to appendix 1 (basic structure/commands of the prompt]

Prompt templates: I constructed 10 prompt templates grouped into two rule regimes. In the certain regimes (T3, T4, T6, T8, T10), the examples fed into the model uniquely identify a reversal rule, and the test input admits exactly one correct output. In the ambiguous regime (T1, T2, T5, T7, T9), the examples in the prompt are consistent with at least two distinct rules – typically identity (f(x)=x) and reversal (f(x)=reverse(x)). The desired output is "cannot be determined."

[link to appendix 2 (table of different rules per prompt)]

Each template is paired with a set of test inputs. For certain reverse templates, inputs vary along length, character class (ASCII, Unicode, emoji), punctuation patterns, and structural complexity (e.g. URLs, SQL statements), with expected outputs computed by literal character reversal. For ambiguous templates, inputs vary based on whether the input itself is a palindrome, or whether it contains separators seen in the demonstrations. In total, 201 model-test-case trials were collected (67 per model).

Scoring: The Format correctness is measured by whether the output contains both RULE:... and ANSWER:..., or consists of a bare "Cannot be determined" acceptable for ambiguous trials. The Decision correctness is measured by whether the models' choice matches the expected rule status of the trial ("Cannot be determined" for the ambiguous regimes, a concrete rule for certain regimes). The rule correctness is documented for certain trials only, where it is measured by whether the stated rule contains lexical markers consistent with reversal (e.g. reverse, backward, mirror, end-to-start). Lastly, the answer correctness is measured by whether there is an exact string match between the model's answer and the expected output.

A trial was recorded as a strict pass only if it satisfied all these dimensions. Latency and error metadata were logged alongside each result.

3.3 Relationships

Our test case explores how LLMs handle relationships. Specifically, we are interested in how LLMs navigate rules that are messy and reflect real life dynamics. To simulate real life human relationships we created two distinct test cases one based on fictional friend group dynamics and the other based on family dynamics. In creating our test cases we were intentional about creating worlds in which there were conflicting or complicated rules, vague or ambiguous interpersonal relationships, and consequences for certain events or decisions. We wanted to assess how the models made judgment calls relating to specific rules and relationships.

To conduct our experiment, we conducted a 5x5 prompting style: five rounds of five questions. After each round we began a new session with the LLM. We measured performance across the following three models: GPT-5, GPT-5-Nano, and GPT-5-Mini. At the beginning of each session we would prompt the model with a world structure containing all of the information about the world it could use to answer the questions including defined relationships between characters in the world, friend group or family norms and rules, secrets, and situations under which characters could lose life points. When initiating our prompting rounds we instructed the model to answer the question with the information that it was given and if it was unable to answer the question it should respond with "unknown".

We made several of the rules and norms arbitrary, conflicting, or vague to test how the models reasoned under these scenarios. During our preliminary prompting rounds we identified five key model failures, which we used to derive the question types used in our data collection. We define a model failure as any instance where a model answers the proposed question incorrectly.

To quantify model performance we defined five distinct categories of questions: retrieval, inverse retrieval, messy, rule application, and conflict resolution. When the model answered a question correctly this prompt was assigned a 1 in the data set, if the question was answered incorrectly it was assigned a zero (see Technical Appendix: Relationships for more detailed

explanations of question types). While conducting our experiment, we attempted to improve model accuracy by implementing a soul.md, the results of this are interpreted in our results section.

The results of our prompting rounds were uploaded to Langfuse and the data produced is interpreted in the Results: Relationships subsection.

4 Results

4.1 Truth or Dare

Overall, all models performed slightly better with Soul.md. GPT-5 performed the best among the models for the Truth or Dare test case. Both GPT-5 models with and without Soul.md mostly received corrected output when asked truth or dare. However, when given a truth or dare question, the model responds well, but is unable to alternate turns. Additionally, all models tend to disclose truth and dare questions when asking the user truth or dare. There was also a drastic difference between the performance of GPT-5 and GPT-5-Nano. In Figure 1 below, GPT-5-Nano had 0 accuracy score out of 15 questions, but when given a soul, it performs slightly better and outputs one correct response. Although GPT-5-Mini also performs poorly compared to GPT-5, it still makes one correct output without Soul.md compared to GPT-5-Nano.

Figure 1: Model Performance: With Soul vs Without Soul(Averaged Across Trials)

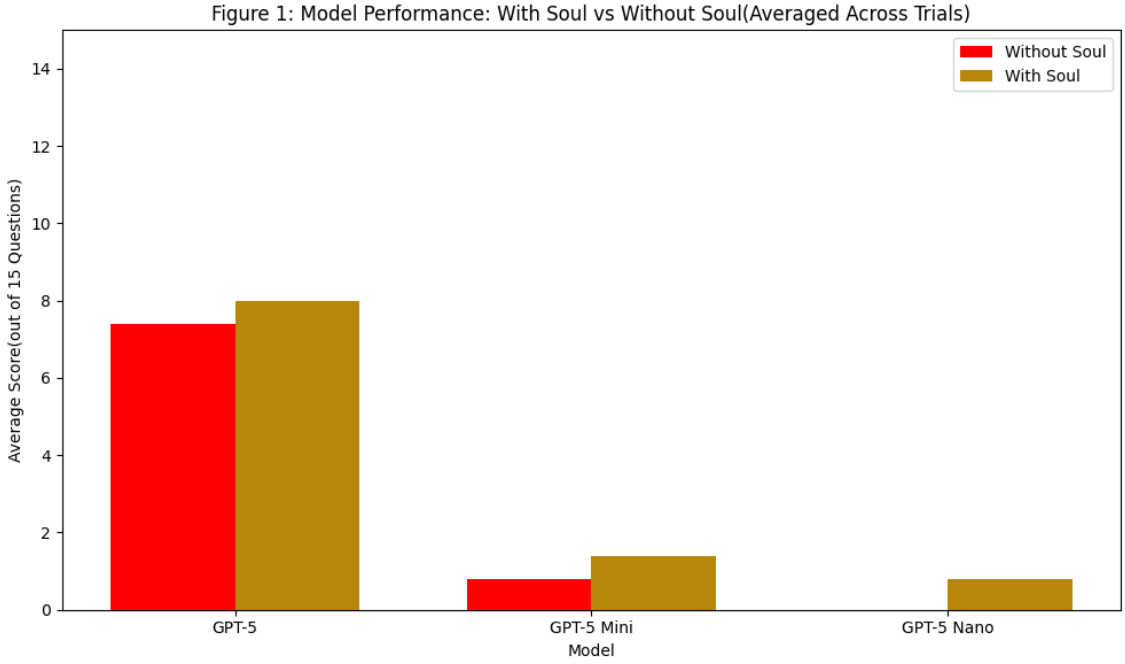
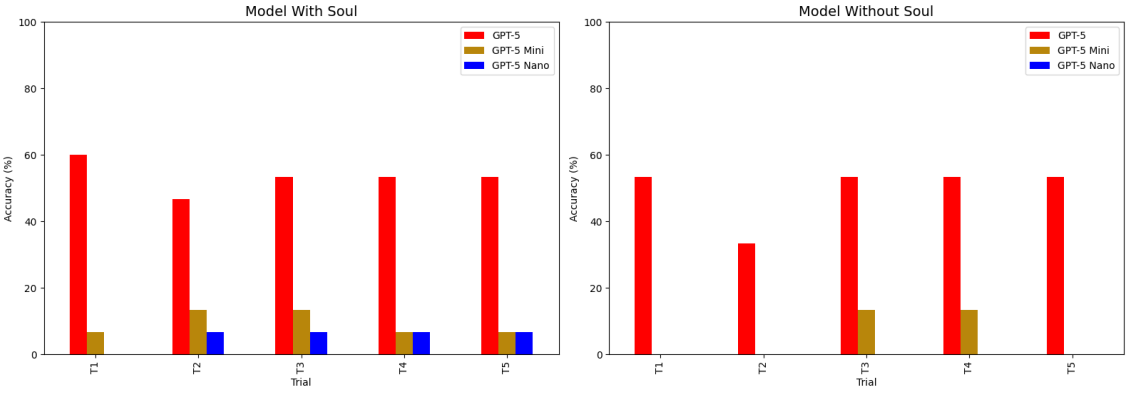


Figure 2 below shows the accuracy of the models with and without soul, based on trials. This is to evaluate the model's performance after multiple trials. For both the GPT-5 model with and without soul, the accuracy performed the best in the first trial while performing the worst in the second trial. Compared to GPT-5, GPT-5-Nano still performs drastically worse, but the accuracy remains consistent as the accuracy remains consistent after the second trial. With GPT-5-Mini, the accuracy of the model increases in the third and fourth trials. The inconsistent accuracy in each trial may suggest that the next steps for this project are to increase the number of trials.

Figure 2: Accuracy by Trial - Model With Soul vs Model Without Soul



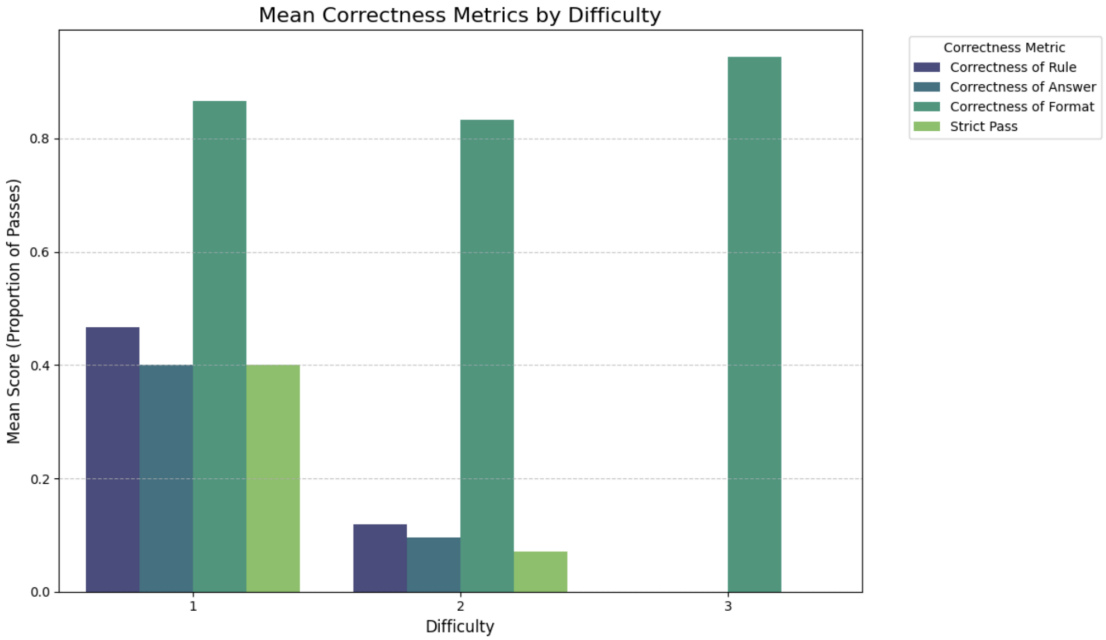
Compared to GPT-5, qualitative findings that explain the drastic difference in accuracy

from GPT-5-Mini and GPT-5-Nano were due to the problems of echoing. Echoing is when the model repeats the prompts or input instead of giving actual answers. When asked truth or dare, the two models reflect the truth or dare question back to the user instead of choosing for themselves. Another problem the models have is when they choose truth or dare, they choose them for the user and prompt the user with the question they chose. The slight improvement from these models when given a soul may be due to the soul.md giving these models a sense of agency. This could explain why the models made improvements when asked truth or dare, as it learns how to choose for itself. Additionally, when the user chooses "dare", GPT-5-Mini and GPT-5-Nano lose track of whose turn it is. When the user inputs just the word "truth," all models lose focus of the game and get confused with the actual meaning of "truth" instead of prompting a truth question to the user. As honesty is also emphasized in the Soul.md, "truth" may have been the trigger word for these models.

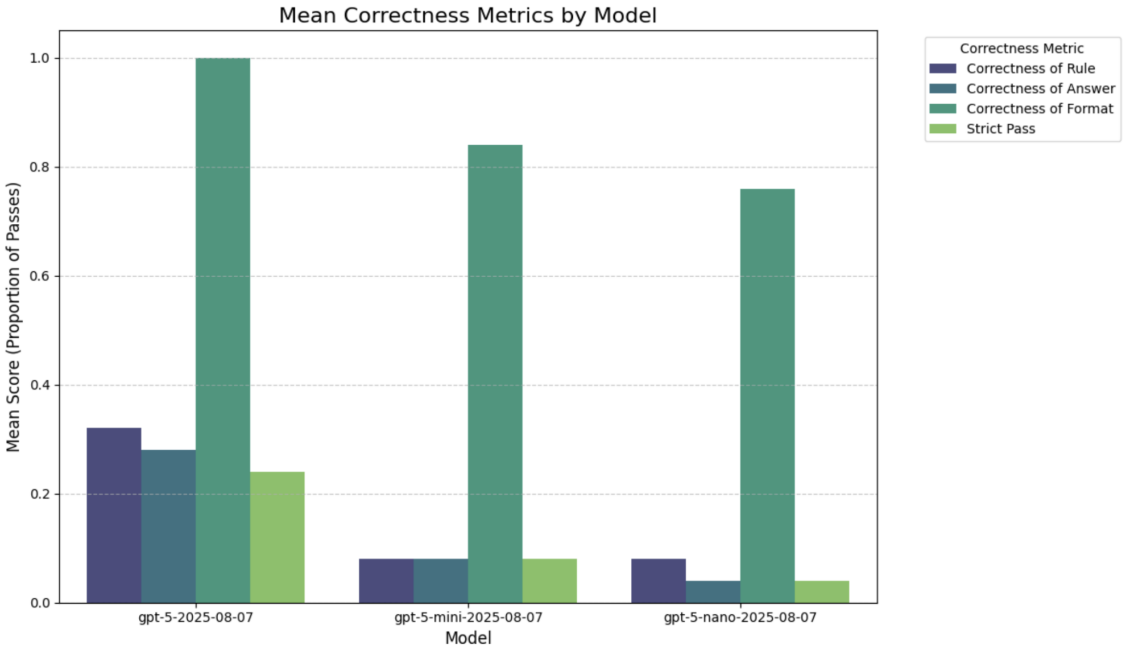
4.2 Parts

4.2.1 Rule-Based Substitution Experiment

Out of 75 data points, only 16%, correctly identified the rule. Only about 13%, provided the correct answer. About 87%, provided the correct formatting when responding, however. Overall, only 12% of responses provided the correct rule, correct answer, and correct formatting. In spite of this low success rate, models remained largely certain of their own correctness, with 89% of responses expressing certainty.



Examining results by difficulty of prompt was quite telling. About 47% of difficulty 1, 12% of difficulty 2, and 0% of difficulty 3 prompts led to the model guessing the correct rule. A similar trend was seen with the models' performance on correctly transforming the provided sentence. About 40% of difficulty 1, 10% of difficulty 2, and 0% of difficulty 3 prompts were correct. Most responses were formatted correctly regardless of difficulty. Certainty decreased very slightly as the models became more difficult, but certainty remained relatively consistent overall. Ultimately, 40% of difficulty 1, 7% of difficulty 2, and 0% of difficulty 3 models counted as strictly correct.



Examining results by model generally suggested better performance from GPT-5 versus the Mini and Nano models, but not much difference was found between Mini and Nano models themselves. About 32% of GPT-5 responses, 8% of GPT-5-Mini responses, and 8% of GPT-5-Nano guessed the correct rule. 28% of GPT-5, 8% of GPT-5-Mini, and 4% of GPT-5-Nano responses correctly modified the provided sentence to match the rule. All GPT-5 responses were in the right format, as were 84% and 76% of GPT-5-Mini and GPT-5-Nano models, respectively. Ultimately, 24% of GPT-5, 8% of GPT-5-Mini, and 4% of GPT-5-Nano responses were strictly correct.

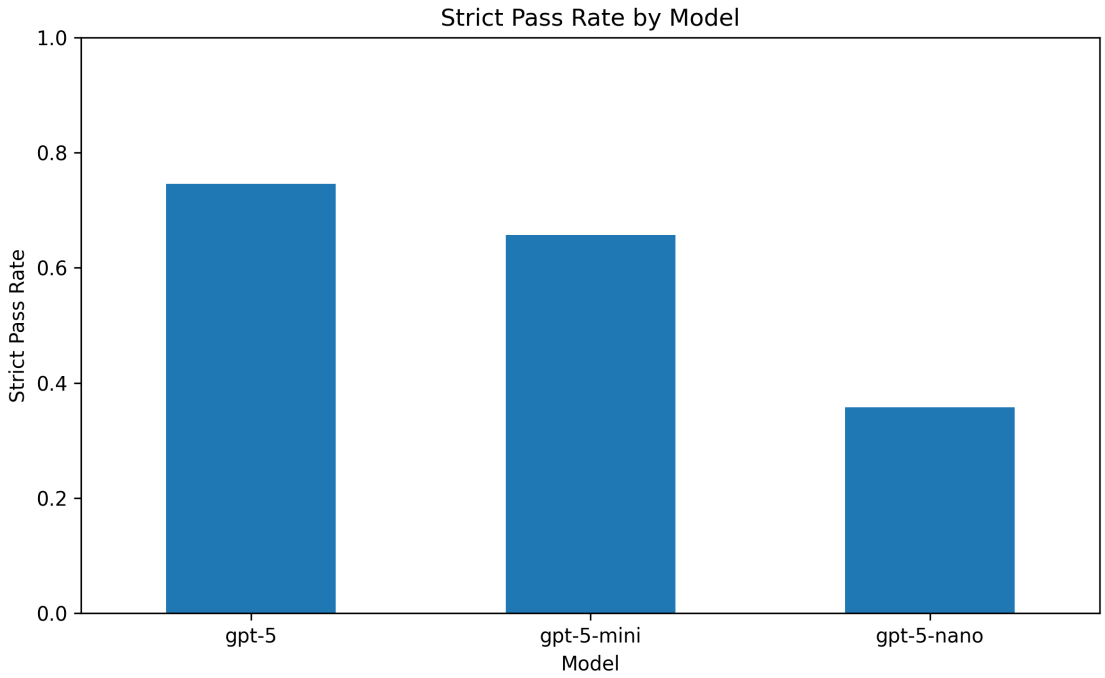
As a data scientist, one must understand the difference between natural variation in means and statistical significance. As such, an ANOVA test was conducted for each of the aforementioned metrics. It was found that there is a statistically significant relationship between correctness of rule ($p<0.001$), correctness of answer ($p<0.001$), strict pass ($p<0.001$), overall correctness ($p<0.03$) and prompt difficulty. There is also a statistically significant relationship between correctness of rule ($p<0.03$), correctness of answer ($p<0.03$), correctness of format ($p<0.04$), overall correctness ($p<0.001$) and GPT model.

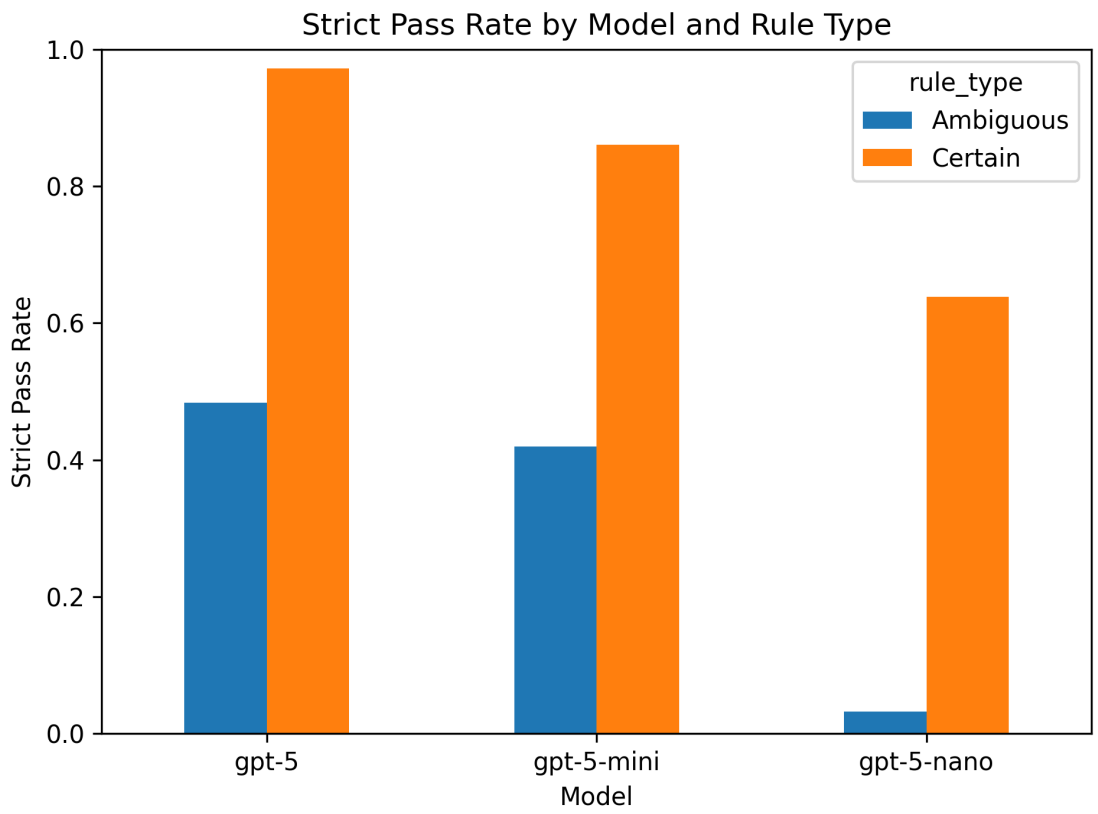
Therefore, it can be concluded that there is a relationship between model and performance, as well as between difficulty of prompt and performance. With that, it is clear that

the way that the LLMs are able to tokenize is affecting their performance, especially, as the prompts become more complex and thus more liable to lose information in the encoding and decoding process.

4.2.2 Rule-Based Palindrome Experiment

Across all 201 trials, strict-pass rates decreased with model scale: gpt-5 passed 50/67 (75%), gpt-5-mini 44/67 (66%), and gpt-5-nano 24/67 (36%). On certain-reverse trials, all three models performed well — gpt-5 passed 35/36 (97%), gpt-5-mini 31/36 (86%), and gpt-5-nano 23/36 (64%), though gpt-5-nano's deficit here is driven largely by 10 empty responses rather than wrong answers. On ambiguous trials, the gap is much sharper: gpt-5 passed 15/31 (48%), gpt-5-mini 13/31 (42%), and gpt-5-nano 1/31 (3%).





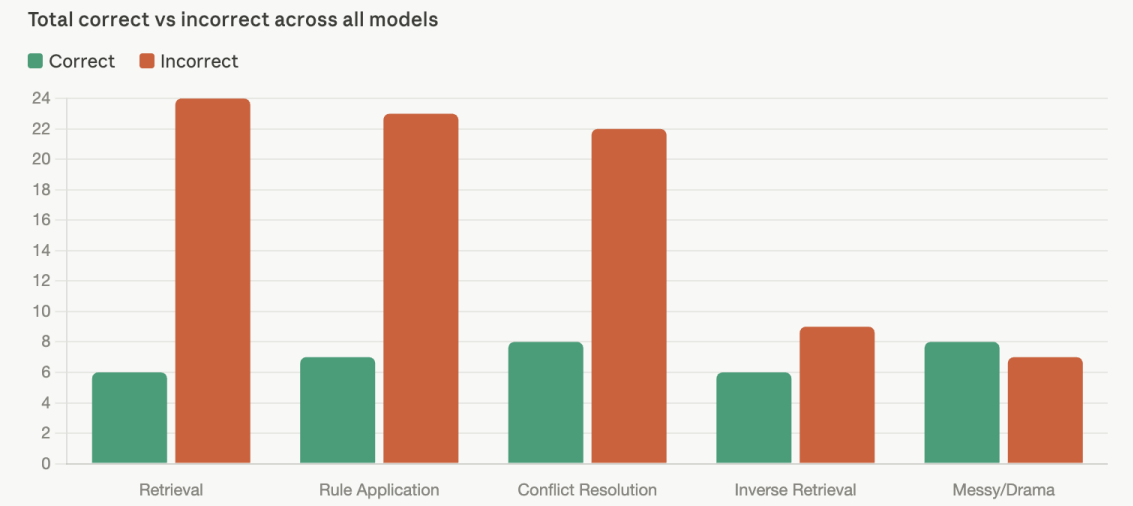
The primary failure mode on ambiguous trials is committing to the identity rule rather than abstaining. On ambiguous failures, we counted roughly 14 identity-rule commits from gpt-5, 17 from gpt-5-mini, and 28 from gpt-5-nano (the remaining ambiguous failures are empty responses or non-identity commits like reversal). Committed rules used recurring wording like "unchanged," "identity," "equals x," and "returns S exactly as given," and were typically paired with an answer equal to the test input.

Interestingly, the behaviors are not uniform across forms. On T1 (palindrome identity, plain alphabetic strings) gpt-5 abstained on 7 of 10 trials, but on T5 (strings with separators like snake_case_var, hello/world) it abstained on only 2 of 10, and on T7 (palindrome+identity) it abstained on 0 of 3. T9 (emoji palindromes) patterns closest to T1: gpt-5 abstained on 3 of 4, gpt-5-mini on 2 of 4. This indicates that the abstention rate is not simply a function of how ambiguous the setup is — T1, T5, T7, and T9 all have the identity/reverse ambiguity structure, but commitment rates differ greatly. In addition, the

templates where models commit most (T5, T7) are those where the demonstration strings most resemble conventional "variable-like" tokens; T9's unfamiliar emoji context and T1's mix of palindromes and non-palindromes leave more surface cues that something other than identity could be going on.

4.3 Relationships

Out of 266 trials we recorded an accuracy rate of 67% for all models for all question types. Additionally, we recorded a 79% accuracy rate across all models for all question types pertaining to scenario one and a 75% accuracy rate across all modes for all questions pertaining to scenario two. Figure X shows the total number of questions answered correctly and incorrectly across models. From this figure, all models noticeably struggled to answer questions in the messy/drama question group as well as inverse retrieval questions. For all models, the recorded accuracy rate for messy questions was 67% and 60% for inverse retrieval.



Out of 266 trials we recorded an accuracy rate of 67% for all models for all question types. Additionally, we recorded a 79% accuracy rate across all models for all question types pertaining to scenario one and a 75% accuracy rate across all modes for all questions pertaining to scenario two. Figure X shows the total number of questions answered correctly and incorrectly across models. From this figure, all models noticeably struggled to answer

questions in the messy/drama question group as well as inverse retrieval questions. For all models, the recorded accuracy rate for messy questions was 67% and 60% for inverse retrieval.

From our qualitative analysis, we noticed that all models struggled comprehending relational tenses. For example, all models failed to understand that "ex" implied no longer being together and when prompted would respond with the name of a character's ex when asked whom they were currently dating. We also noticed that in messy scenarios under which multiple characters had broken rules or group norms the models would arbitrarily choose one character who was at fault and would fail to assign the blame to multiple characters. This reveals that many models are trained to operate under black and white reasoning and struggle to reason along a continuum. For the messy questions, we also notice that models struggled to apply multiple rules at once. For example, a model may successfully apply a rule when asked to apply that rule in isolation, but may be unable to apply said rule successfully when it is asked in conjunction with other rules. Similarly, we also noticed that the models struggled to reason in reverse. One question the model had particular challenges with was determining where one character works. The models were given the facts that this character works at the hospital and that there was only one hospital in town the expected answer would have been the name of the hospital however all three of the models responded with unknown

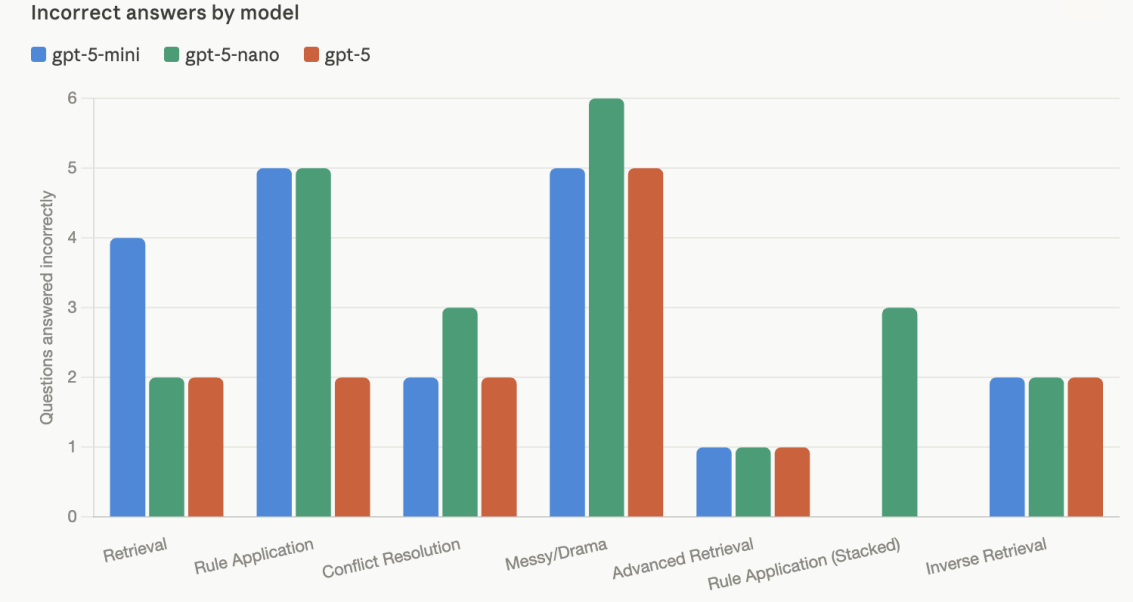


Figure X shows performance between models. Specifically, this figure shows how many questions each model answered incorrectly. Across questions types GPT-5-mini and GPT-5-nano generally performed more poorly than GPT-5. All models had problems with rule application and messy questions.

5 Conclusion/Future Research

This paper studies how large language models behave when they are asked to adapt to new tasks purely through context. We tested on three types of tasks - rule induction, structured reasoning, and relational decision-making. We found that models consistently default to simple, low-complexity rules that are compatible with the provided examples, even when those examples are insufficient to uniquely determine a rule. In ambiguous settings, this manifests as a failure to abstain: models prefer to commit to a plausible rule, most often identity, rather than acknowledge uncertainty.

Across tasks, we also observe that model confidence is poorly calibrated to actual performance, and that increasing model scale reduces but does not eliminate these patterns. Taken together, these findings suggest that current LLMs exhibit consistent inductive biases

in how they interpret and act on context.

These results point toward a shift in how we think about improving model behavior. Rather than treating these failures as isolated errors, they can be understood as predictable responses to how information is presented. This opens the door to teaching patches: lightweight, context-level interventions designed to explicitly guide model behavior during inference without modifying model weights.

In the rule-inference task, patches can focus on directing attention to character-level structure—for example, by explicitly segmenting strings or highlighting relevant substrings to reduce tokenization-related information loss. In the Relationships task, introducing a clearer sequential hierarchy in the prompt (e.g., organizing global rules, group norms, and individual attributes) or incorporating RAG-like retrieval within the context may help models better organize and access relevant information for each query. In the Truth-or-Dare task, patches that enforce turn-based tracking—such as explicit counters for the number of truths and dares, or constraints that prevent reuse of trigger words like "truth" after selection—may improve consistency across multi-turn interactions.

Patches can also be on measuring whether models appropriately align confidence with correctness. More broadly, teaching patches suggest a practical pathway for improving reliability in deployed systems. If model failures are systematic, then they are, in principle, correctable through systematic changes to context. Understanding which patches generalize across tasks, and how they interact with model scale and architecture, remains an open question. Addressing this will be critical for moving from observing model behavior to actively shaping it.

6 References

References

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). *Language Models are Few-Shot Learners*. Advances in Neural Information Processing Systems (NeurIPS).

[2] Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. (2023). *What Learning Algorithm is In-Context Learning? Investigations with Linear Models*. International Conference on Learning Representations (ICLR).

[3] Mäkinen, J. (2025). *Context Engineering for AI-Assisted Software Development*.

[4] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. Advances in Neural Information Processing Systems (NeurIPS).

[5] Dsouza, N. (2025). *Comparative Analysis of Leading Generative AI Conversational Systems: ChatGPT, Grok AI, Gemini, and Meta AI*. TechRxiv. https://doi.org/10.36227/techrxiv.175393328.84629021/v1