Benchmarking LLM Sensitivity to Prompt Formats: A Contamination-Free Approach

Fumio Miyata

Abstract

The evaluation of Large Language Models (LLMs) is complicated by prompt sensitivity and data contamination, which obscures the distinction between genuine reasoning and rote memorization. To address this, we introduce a reproducible, contamination-free benchmark that measures how LLM responses vary with the prompt's language, style, and syntactic format. Our methodology uses the constructed language Lojban—virtually absent from pre-training corpora—and a suite of novel symbolic prompting tasks to assess a model's ability to interpret unfamiliar formal systems. The results indicate three key findings: (1) prompt strictness can elicit latent capabilities, but its effectiveness is limited to familiar languages; (2) models exhibit a significant ceiling in algorithmic complexity, failing to produce bug-free code for novel tasks; and (3) performance appears more indicative of sophisticated pattern matching than abstract reasoning. This work provides a comprehensive dataset and a rigorous framework for evaluating the generalization and true reasoning abilities of LLMs.

1 Introduction

Evaluating the capabilities of Large Language Models (LLMs) is a central challenge in modern AI research [13]. While their performance is impressive, assessing it accurately is complicated by two well-documented issues: **Prompt Sensitivity**, where even subtle rephrasings can lead to vastly different outputs, and **Data Contamination**, where benchmark evaluation data is inadvertently included in a model's training set [9]. This contamination is a critical flaw in many public benchmarks, as it makes it difficult to determine if a model is demonstrating genuine reasoning or simply recalling a memorized answer.

The core of this problem is a gap in evaluation methodology. Existing benchmarks often cannot reliably test a model's ability to generalize to truly novel problems. To address this gap, our study introduces a new benchmark designed around the principle of data purity. We employ two primary strategies to create a contamination-free evaluation environment. First, we use the constructed language **Lojban**, a logical language with an unambiguous grammar virtually absent from pre-training corpora. Second, we developed **Symbolic Prompting**, a suite of custom tasks requiring the interpretation of formal rule sets in formats like S-expressions and JSON. This approach allows us to test a model's ability to adapt to unfamiliar formalisms, a closer proxy for genuine reasoning than performance on known benchmarks.

In this paper, we present the experimental framework, the resulting dataset, and a detailed analysis of our findings. We report on a series of experiments applying these prompts to a wide range of open-source LLMs. Our results suggest that (1) prompt format and strictness are critical factors in eliciting model capabilities, (2) models exhibit a clear ceiling when faced with complex, novel algorithmic implementation tasks, and (3) their performance is more consistent with sophisticated pattern-matching than with abstract reasoning. These findings have significant implications for how we should interpret LLM capabilities and design future evaluation methods.

2 Experimental Environment

2.1 Setting up the Execution Environment

To reproduce this experimental package, it is recommended to set up a Python virtual environment following the procedure described in the project's `README.md` file.

2.2 Evaluated Models and Inference Parameters

This study evaluates a range of open-source language models with fewer than 8 billion parameters, accessed via Ollama. The specific models and their inference hyperparameters are summarized in Table 1.

Table 1: Evaluated Models and Inference Hyperparameters

Model Name (Ollama)	Parameters	Quant. (Est.)	Temp	Top P	Seed
`gemma3:270m`	0.27B	Q4_K_M	0.0	1.0	0
`smollm:360m`	0.36B	Q4_K_M	0.0	1.0	0
`qwen:0.5b`	0.5B	Q4_K_M	0.0	1.0	0
`tinyllama:1.1b`	1.1B	Q4_K_M	0.0	1.0	0
`deepseek-r1:1.5b`	1.5B	Unknown	0.0	1.0	0
`stablelm2:1.6b`	1.6B	Q4_K_M	0.0	1.0	0
`qwen:1.8b`	1.8B	Q4_K_M	0.0	1.0	0
`gemma:2b`	2B	Q4_K_M	0.0	1.0	0
`falcon3:3b`	3B	Unknown	0.0	1.0	0
`llama3.2:3b`	3.2B	Unknown	0.0	1.0	0
`phi3:mini`	3.8B	Q4_K_M	0.0	1.0	0
`gemma3:4b`	4B	Q4_K_M	0.0	1.0	0
`qwen:4b`	4B	Q4_K_M	0.0	1.0	0
`yi:6b`	6B	Q4_K_M	0.0	1.0	0
`gemma:7b`	7B	Q4_K_M	0.0	1.0	0
`mistral:7b`	7B	Q4_K_M	0.0	1.0	0
`llama2:7b`	7B	Q4_K_M	0.0	1.0	0
`deepseek-llm:7b`	7B	Q4_K_M	0.0	1.0	0
`deepseek-r1:8b`	8B	Unknown	0.0	1.0	0
`llama3:8b`	8B	Q4_K_M	0.0	1.0	0

Note: Quantization methods are based on the Ollama library's defaults, typically `Q4_K_M` where specified. Details for some models marked 'Unknown' were not available.

To ensure deterministic reproducibility, all experiments were conducted with inference hyperparameters fixed at `temperature: 0.0`, `top_p: 1.0`, and `seed: 0`. This setup minimizes stochastic variations in model responses, allowing for a focused evaluation of the impact of the prompt format itself.

2.3 Evaluation Criteria

This benchmark defines task-specific success criteria as follows:

Code Generation Tasks: Success is defined as the generated Python code passing all predefined `Pytest` unit tests **without any modification**. Each task is attempted only once, making this a strict **Pass@1** evaluation.

String Generation Task (`filtered_list`): Success requires that the list-formatted string returned by the model perfectly matches the expected output at the character level. This test

evaluates the ability to follow instructions and adhere to a format directly, rather than generating code.

Logical Reasoning Tasks:

- **Implementation Correctness**: Success is defined as the generated code correctly implementing the given logic (from S-expressions, JSON, a custom token language, etc.) and passing all associated unit tests.
- **Qualitative Analysis of Thought Process**: For tasks requiring Chain-of-Thought, the generated reasoning is qualitatively analyzed for faithfulness to the rules, logical consistency, and its influence on the final answer.

3 Overview of Experimental Design

The benchmark comprises a suite of tasks that span from simple code generation to complex logical reasoning. For a complete description of each test, refer to the `README.md` file. This section highlights the conceptual design of four key experiments.

3.1 Conceptual Diagrams of Each Test

The following diagrams illustrate the workflows of the main tests in this benchmark.

4 Factual Results

This section presents the primary quantitative and qualitative results from the benchmark experiments. It is intentionally descriptive, reserving all interpretation for Section 5. Full data tables for all 30 runs of each test are available in Appendix A.

4.1 Outcome of the `Filtered List` Task

The `Filtered List` task, which required models to return a formatted string rather than code, was executed under two conditions.

1. **With an Ambiguous Prompt**: The initial prompt, which simply asked for a list, led most models to erroneously generate Python code that would produce the list.
2. **With a Strict Prompt**: When the prompt was modified with a strict constraint—"Return only the list string, not a program code"—a significant number of models, including `deepseek-r1:8b` and `llama3:8b`, achieved high success rates for prompts in English and Japanese. For the same task in Lojban, however, all models failed, even with the strict constraint.

4.2 Observed Failure Pattern in the `einstein_token_test`

The `einstein_token_test` confronted models with a multi-step reasoning task involving an unknown language. We observed a consistent, phased failure pattern:

1. Given only the puzzle in the token language, no model produced comprehensible output.
2. After adding the language's grammar rules to the prompt, some models attempted to generate Python code, but this code failed with `SyntaxError`.
3. With the further addition of a few-shot example demonstrating a backtracking algorithm, models like `llama3:8b` produced syntactically valid code. However, this code failed at runtime, producing `KeyError` exceptions or timing out.

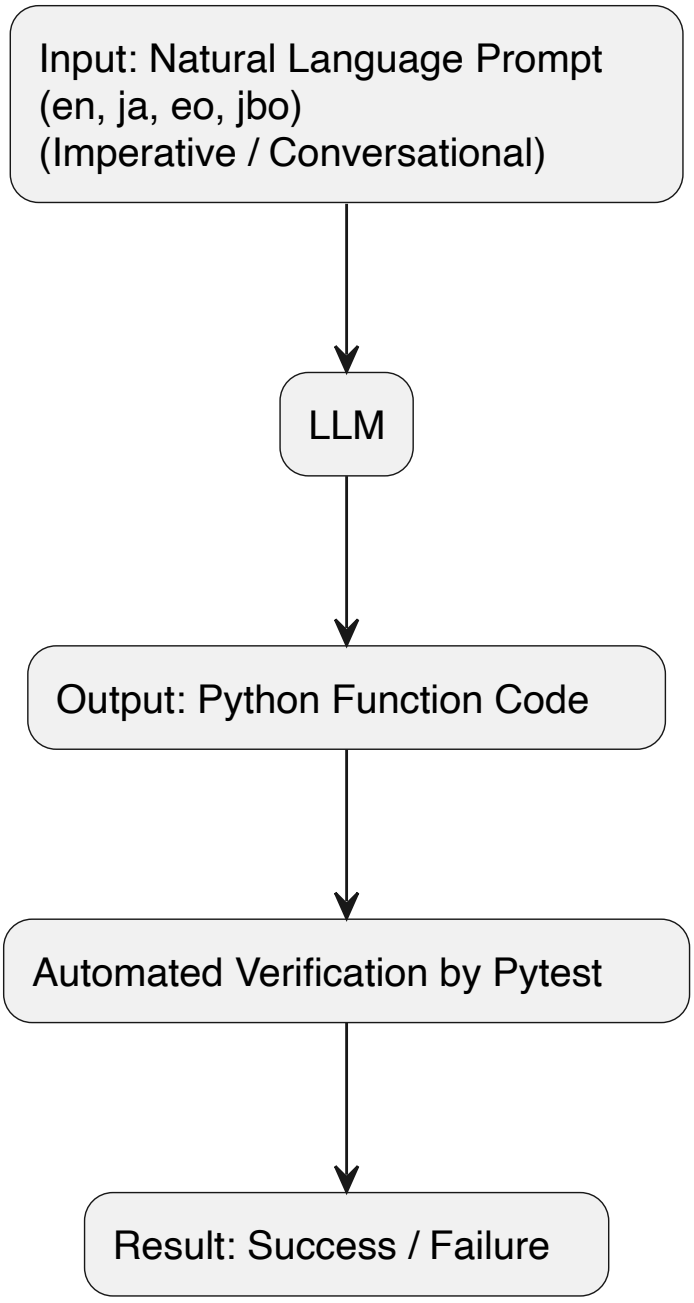

Figure 1: Conceptual workflow for simple code generation tasks.

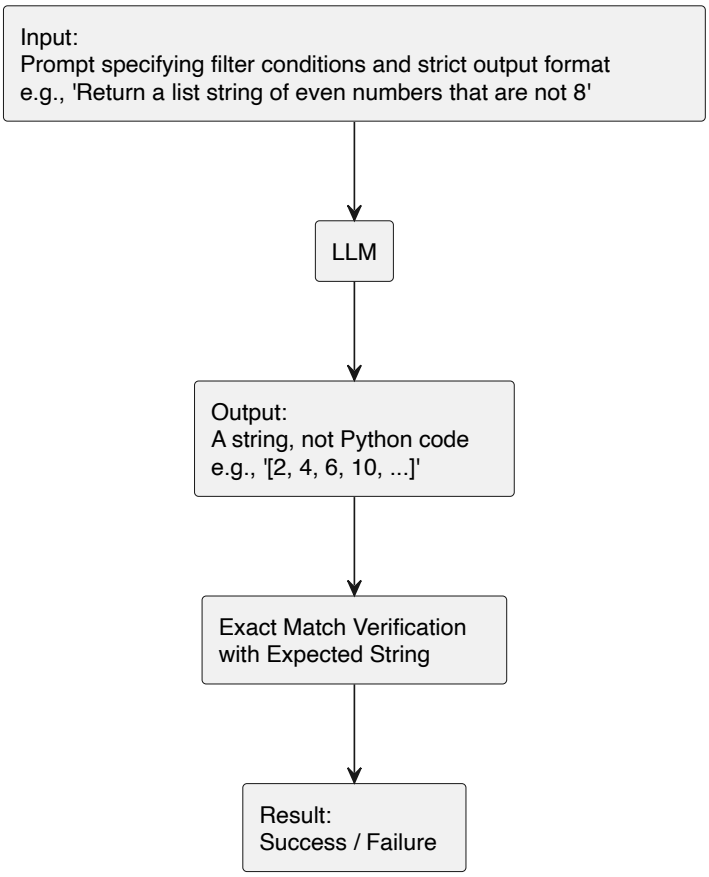

Figure 2: Workflow for the `filtered_list` task.

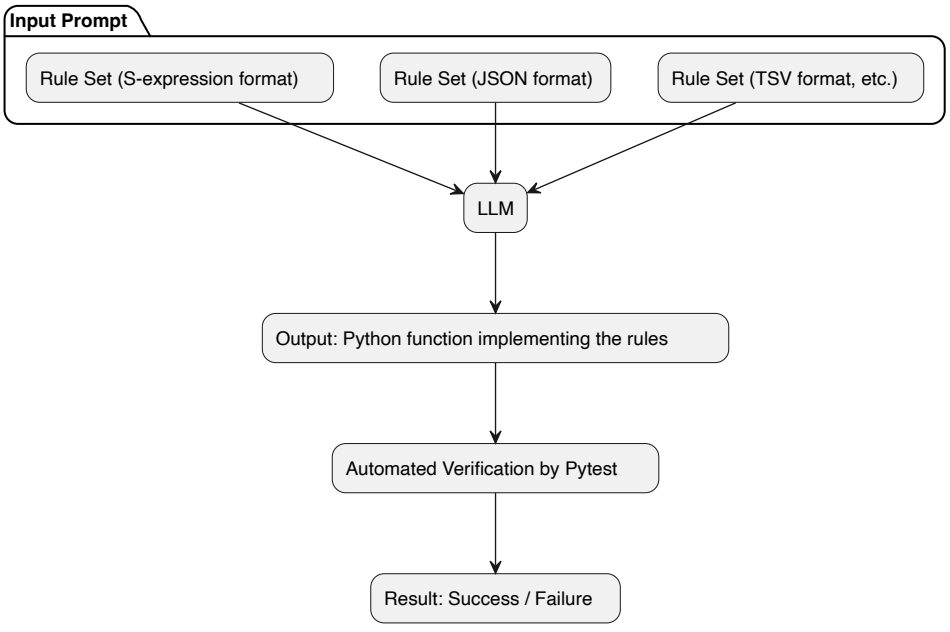

Figure 3: Workflow for symbolic reasoning tasks.

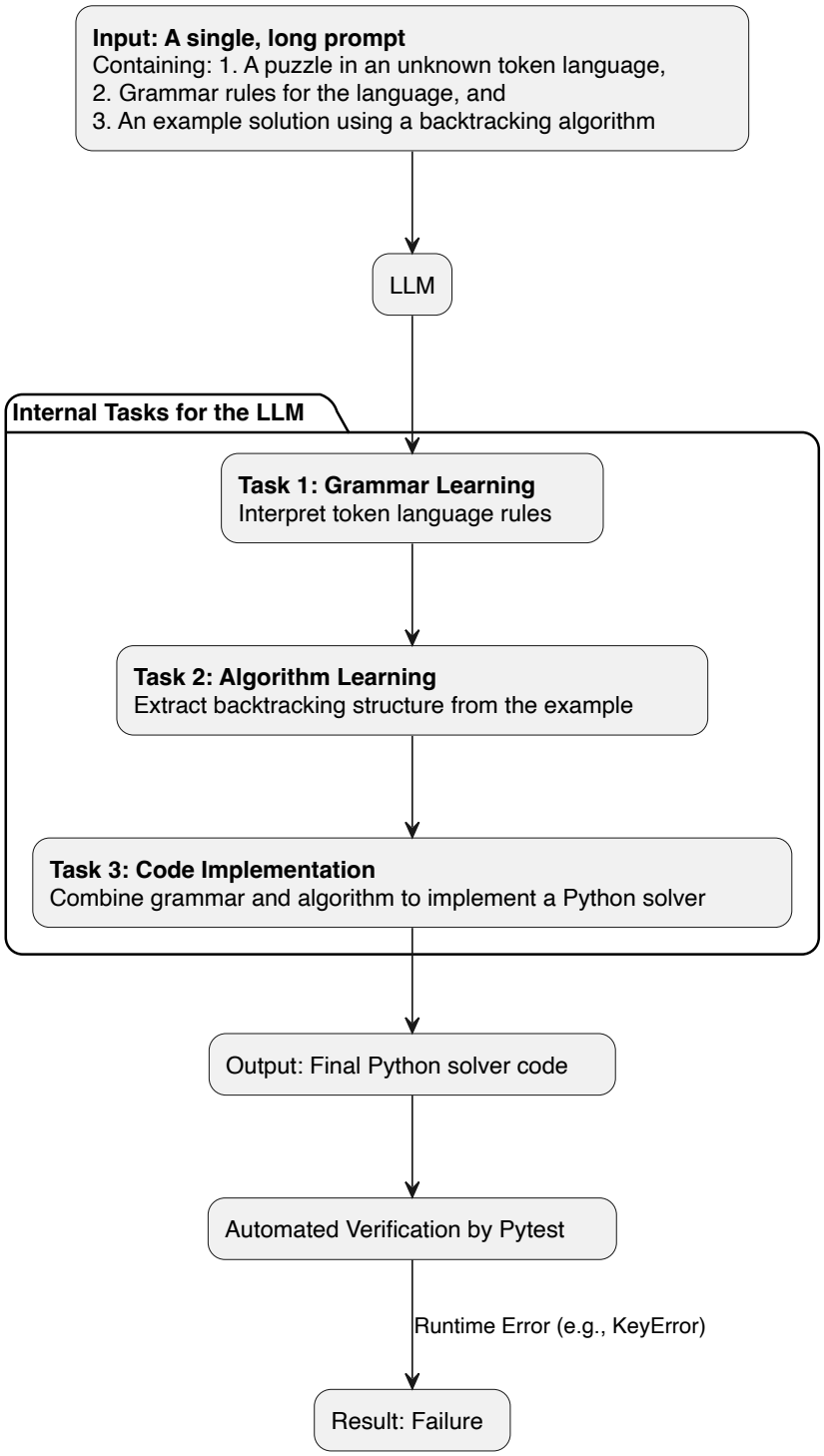

Figure 4: Workflow for the `einstein_token_test` task.

4.3 Performance on Symbolic vs. Natural Language Prompts

In logical reasoning tasks such as `diagnosis`, models were prompted with rulesets in both natural language and symbolic formats. For some models, prompts using unambiguous symbolic languages like S-expressions and JSON yielded higher success rates than their natural language counterparts.

4.4 Overall Performance and Model Scaling

The aggregated results across all tests did not show a clear correlation between model parameter count and overall performance. As detailed in the rankings in Section 5.4, several smaller models (e.g., `yi:6b`, `gemma3:4b`) outperformed larger ones.

5 Discussion

In this section, we offer an interpretation of the factual results presented in Section 4, discussing their potential implications for understanding LLM capabilities and evaluation.

5.1 Interpretation of Prompt Strictness and Familiarity

The results from the `Filtered List` test suggest that prompt strictness can be a crucial factor in eliciting correct model behavior. The dramatic performance improvement after adding a simple constraint indicates that models may possess latent abilities that are only unlocked when the expected output format is unambiguously specified.

However, this elicitation effect appears to be contingent on linguistic familiarity. The universal failure on the Lojban version of the same task suggests that the models did not leverage the language's unambiguous grammar for logical deduction. A plausible interpretation is that the models process unfamiliar languages like Lojban as sequences of unknown tokens, making them unable to perform the semantic task of mapping the language's logic to a specific output format. This asymmetry is consistent with phenomena like the "Reversal Curse" [2] and points to a potential broader issue of language understanding versus logical execution.

5.2 Interpreting the Ceiling in Algorithmic Implementation

The phased failure pattern in the `einstein_token_test` may indicate a ceiling in the ability of current LLMs to implement novel, complex algorithms. The final stage of failure—runtime errors in syntactically plausible code—is particularly revealing. A qualitative analysis of the code generated by `llama3:8b` highlights a potential cause: a failure to ensure state independence during recursion.

```
# Schematic excerpt of code generated by llama3:8b
def solve_puzzle(rules, assignments):
    # ...
    for value in domain_values(var):
        # The dictionary is mutated directly,
        # potentially leaking state across search paths.
        assignments[var] = value
        if is_consistent(assignments, rules):
            result = solve_puzzle(rules, assignments)
            # ...
    # A backtracking step to undo the mutation is missing.
    return None
```

One interpretation of this error is that while an LLM can imitate the superficial structure of an algorithm (loops, recursion), it may not be grasping the underlying logical requirements, such as state isolation in backtracking. This finding aligns with reported systematic failures in compositional reasoning and suggests that model abilities in this area may be confined to sophisticated pattern matching. It also contrasts with findings on Chain-of-Thought prompting, where including step-by-step reasoning can improve performance on complex tasks [10], indicating that the format of reasoning is highly consequential.

5.3 Interpreting the Effectiveness of Symbolic Interfaces

The higher success rates on certain tasks when using symbolic prompts (S-expressions, JSON) suggest that these formats can serve as a more effective computational interface for LLMs in logically structured problems. Two factors may contribute to this: (1) **Syntactic Unambiguity**, which could reduce a model's interpretive overhead, and (2) **Token Efficiency**, which allows more complex logic to be represented concisely. This finding lends support to the potential value of hybrid symbolic-neural approaches for creating more reliable systems [5].

5.4 Interpreting Overall Performance and Lojban Failure

The lack of a clear correlation between model size and performance on our benchmark suggests that factors like architecture and training data composition may play a more significant role than parameter count alone for these specific reasoning tasks.

Furthermore, the consistent and near-total failure on Lojban tasks provides additional evidence that model performance may be rooted in pattern recognition rather than abstract reasoning. In a separate suite of basic Lojban translation tasks, all 20 models failed, suggesting the limited successes in the main benchmark were likely superficial "transpilation" of token patterns. One hypothesis is that this difficulty may originate at the tokenizer level, where Lojban's unique morphology could be fragmented into meaningless sub-tokens, thereby impeding the learning of word-level semantics.

6 Limitations

This study's focus on lightweight, open-source models invites a comparative analysis with larger models (e.g., GPT-4, Claude 3). The algorithmic tests could also be expanded beyond backtracking to other complex tasks like dynamic programming to probe the generality of our findings. Finally, the use of deterministic generation parameters (`temperature: 0.0`) leaves room to explore how stochastic settings affect performance.

7 Conclusion

This paper presented a novel, contamination-free benchmark to probe the reasoning and implementation abilities of LLMs. Our experiments yielded several key observations: the effectiveness of prompt strictness appears contingent on linguistic familiarity; models may exhibit a ceiling in complex algorithmic implementation; and overall performance seems more rooted in pattern matching than abstract reasoning. These observations lend further support to the 'mirage' theory of emergent abilities [7], which posits that such capabilities are a predictable outcome of scale, rather than an unpredictable 'emergence' of new abilities [11]. We hope this framework contributes to a more nuanced evaluation of LLM capabilities and informs the future design of models that can reason more robustly.

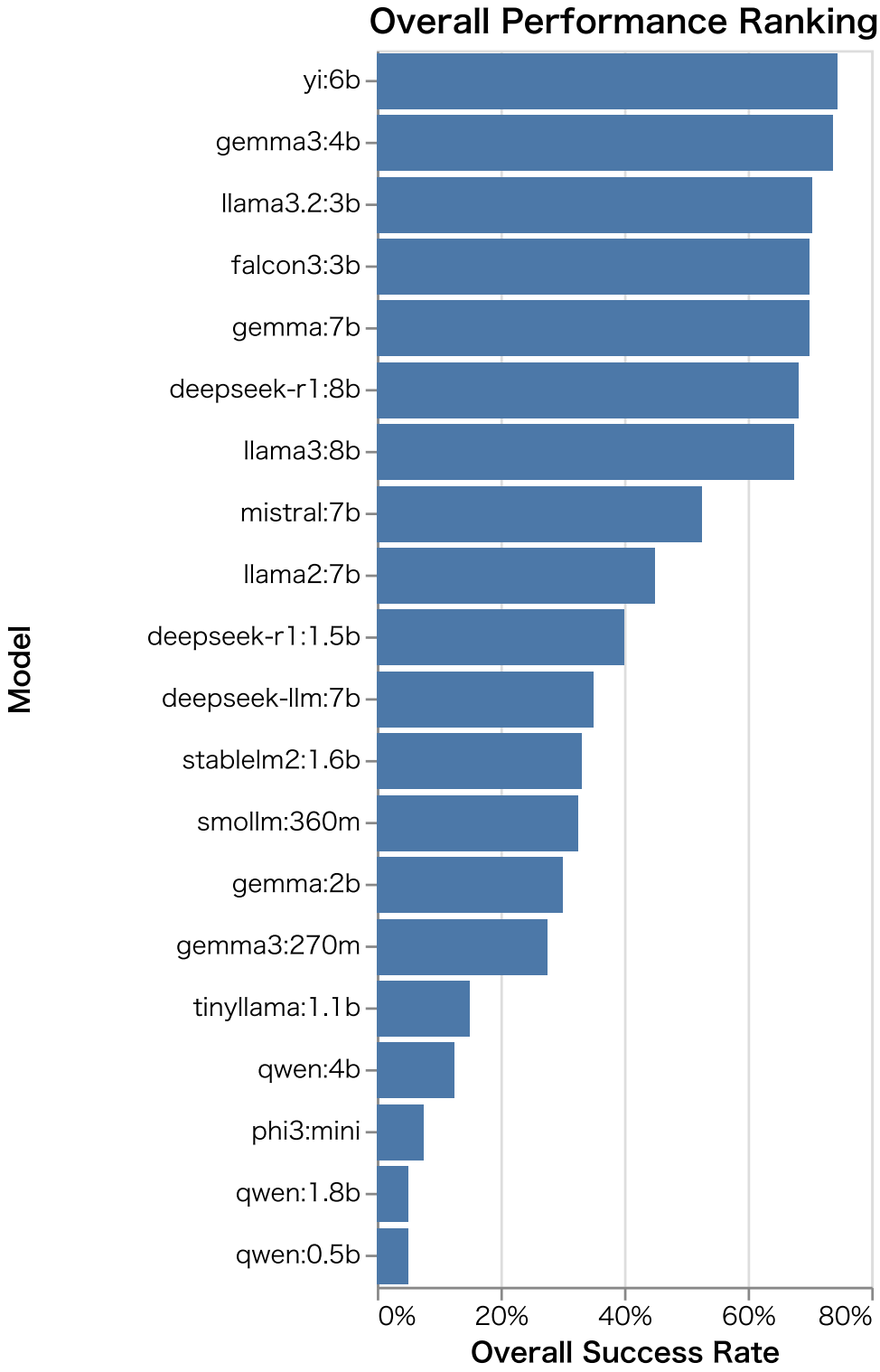

Figure 5: Overall Performance Ranking.

8 Ethical Considerations

This study highlights significant performance disparities in LLMs when prompted with non-mainstream languages like Lojban and Esperanto. These results suggest that current models are predominantly trained on data from mainstream languages such as English, creating a bias that risks the inadequate representation of other languages and cultures. From an ethical standpoint, our work underscores the need for greater diversity and inclusivity in the datasets used for future model development.

References

[1] Anam, M. (2025). *Prompt Engineering and the Effectiveness of Large Language Models in Enhancing Human Productivity*. arXiv preprint arXiv:2507.18638.

[2] Berglund, L., et al. (2023). The Reversal Curse: LLMs trained on Ä is B̈fail to learn B̈ is A∴ In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*.

[3] Besta, M., et al. (2023). Graph of Thoughts: Solving Elaborate Problems with Large Language Models. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*.

[4] Gao, L., et al. (2022). Program-Aided Language Models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, vol. 202, pp. 10764-10799. PMLR.

[5] Marcus, G. (2020). *The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence*. arXiv preprint arXiv:2002.06177.

[6] Ronanki, S., et al. (2025). *Prompt Engineering Guidelines for Using Large Language Models in Requirements Engineering*. arXiv preprint arXiv:2507.03405.

[7] Schaeffer, R., et al. (2023). Are Emergent Abilities of Large Language Models a Mirage?. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

[8] Thomas, A. W., et al. (2023). *Unsolvable Problems for Large Language Models: A Formal Language Approach*. arXiv preprint arXiv:2310.16799.

[9] Vaugrante, L., et al. (2025). Prompt Engineering Techniques for Language Model Reasoning Lack Replicability. In *Transactions on Machine Learning Research*.

[10] Wei, J., et al. (2022a). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022))*, pp. 24824–24837.

[11] Wei, J., et al. (2022b). Emergent Abilities of Large Language Models. In *Transactions on Machine Learning Research*.

[12] Yao, S., et al. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*.

[13] Zhao, W. X., et al. (2023). *A Survey of Large Language Models*. arXiv preprint arXiv:2303.18223.

A Appendix A: Detailed Experiment Results

Below is a summary of the number of successful trials from a large-scale experiment with 30 runs for each test.

Table 2: Return One Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	0/30	0/30	0/30	0/30
smollm:360m	0/30	30/30	30/30	0/30
qwen:0.5b	0/30	30/30	0/30	0/30
tinyllama:1.1b	30/30	30/30	30/30	0/30
deepseek-r1:1.5b	30/30	30/30	30/30	0/30
stablelm2:1.6b	30/30	30/30	30/30	1/30
qwen:1.8b	30/30	30/30	0/30	1/30
gemma:2b	30/30	30/30	30/30	0/30
falcon3:3b	30/30	30/30	30/30	0/30
llama3.2:3b	30/30	30/30	30/30	30/30
phi3:mini	30/30	30/30	30/30	0/30
gemma3:4b	30/30	30/30	30/30	30/30
qwen:4b	30/30	30/30	30/30	0/30
yi:6b	30/30	30/30	30/30	30/30
gemma:7b	30/30	30/30	30/30	30/30
mistral:7b	30/30	30/30	30/30	30/30
llama2:7b	30/30	30/30	30/30	0/30
deepseek-llm:7b	30/30	30/30	30/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	30/30
llama3:8b	29/30	30/30	30/30	30/30

Table 3: Copy List Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	30/30	30/30	0/30	0/30
smollm:360m	30/30	30/30	30/30	30/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	30/30	30/30	30/30	0/30
deepseek-r1:1.5b	30/30	30/30	0/30	0/30
stablelm2:1.6b	0/30	30/30	30/30	0/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	30/30	30/30	30/30	0/30
falcon3:3b	30/30	30/30	30/30	30/30
llama3.2:3b	30/30	30/30	30/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	15/30
qwen:4b	0/30	30/30	0/30	0/30
yi:6b	30/30	30/30	30/30	30/30
gemma:7b	30/30	30/30	0/30	30/30
mistral:7b	30/30	30/30	30/30	30/30
llama2:7b	0/30	30/30	30/30	0/30
deepseek-llm:7b	30/30	30/30	30/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	30/30
llama3:8b	0/30	30/30	0/30	0/30

Table 4: Simple Sort Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	30/30	30/30	0/30	0/30
smollm:360m	0/30	30/30	0/30	0/30
qwen:0.5b	30/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	30/30	30/30	0/30	30/30
stablelm2:1.6b	0/30	30/30	30/30	0/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	30/30	0/30	0/30	0/30
falcon3:3b	30/30	30/30	30/30	30/30
llama3.2:3b	30/30	30/30	30/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	30/30
qwen:4b	0/30	30/30	0/30	0/30
yi:6b	30/30	30/30	30/30	30/30
gemma:7b	30/30	30/30	30/30	30/30
mistral:7b	30/30	30/30	30/30	30/30
llama2:7b	30/30	30/30	30/30	0/30
deepseek-llm:7b	0/30	30/30	30/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	30/30
llama3:8b	30/30	30/30	30/30	30/30

Table 5: Reverse Sort Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	0/30	0/30	0/30	0/30

Table 5 – continued from previous page

Model / Language	ja	en	eo	jbo
smollm:360m	0/30	30/30	30/30	0/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	30/30	30/30	30/30	0/30
stablelm2:1.6b	22/30	30/30	0/30	30/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	0/30	0/30	0/30	0/30
falcon3:3b	30/30	30/30	30/30	30/30
llama3.2:3b	30/30	30/30	30/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	30/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	30/30	30/30	30/30	30/30
gemma:7b	30/30	30/30	30/30	30/30
mistral:7b	30/30	30/30	30/30	30/30
llama2:7b	30/30	30/30	30/30	0/30
deepseek-llm:7b	0/30	30/30	30/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	30/30
llama3:8b	30/30	30/30	30/30	30/30

Table 6: Length Sort Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	30/30	30/30	0/30	0/30
smollm:360m	30/30	30/30	30/30	30/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	30/30	30/30	0/30	0/30
stablelm2:1.6b	0/30	30/30	29/30	0/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	0/30	0/30	0/30	0/30
falcon3:3b	0/30	30/30	30/30	30/30
llama3.2:3b	30/30	30/30	30/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	30/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	30/30	30/30	30/30	0/30
gemma:7b	30/30	30/30	30/30	0/30
mistral:7b	30/30	29/30	30/30	0/30
llama2:7b	0/30	30/30	30/30	30/30
deepseek-llm:7b	0/30	30/30	30/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	30/30
llama3:8b	30/30	30/30	30/30	30/30

Table 7: Custom Sort Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	0/30	0/30	0/30	0/30
smollm:360m	0/30	0/30	0/30	0/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	30/30	0/30	30/30	0/30
stablelm2:1.6b	0/30	0/30	0/30	0/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	0/30	0/30	0/30	0/30
falcon3:3b	30/30	30/30	30/30	0/30
llama3.2:3b	30/30	30/30	30/30	0/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	0/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	30/30	30/30	30/30	30/30
gemma:7b	30/30	30/30	30/30	0/30
mistral:7b	0/30	0/30	0/30	0/30
llama2:7b	0/30	0/30	0/30	0/30
deepseek-llm:7b	0/30	0/30	30/30	0/30
deepseek-r1:8b	29/30	30/30	30/30	0/30
llama3:8b	0/30	30/30	0/30	0/30

Table 8: Roundtrip (Fibonacci S-Expr) Success Rates

Model / Format	sexpr
gemma3:270m	0/30
smollm:360m	0/30
qwen:0.5b	0/30
tinyllama:1.1b	0/30
deepseek-r1:1.5b	0/30
stablelm2:1.6b	0/30
qwen:1.8b	0/30
gemma:2b	0/30
falcon3:3b	0/30
llama3.2:3b	0/30
phi3:mini	0/30
gemma3:4b	30/30

Table 8 – continued from previous page

Model / Format	sexpr
qwen:4b	0/30
yi:6b	23/30
gemma:7b	0/30
mistral:7b	30/30
llama2:7b	0/30
deepseek-llm:7b	0/30
deepseek-r1:8b	8/30
llama3:8b	0/30

Table 9: Diagnosis Logic Success Rates

Model / Format	S-expression	json	tsv	token-test
gemma3:270m	0/30	0/30	0/30	0/30
smollm:360m	0/30	0/30	0/30	0/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	30/30	0/30	0/30	0/30
stablelm2:1.6b	28/30	0/30	0/30	4/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	30/30	0/30	30/30	30/30
falcon3:3b	30/30	30/30	30/30	0/30
llama3.2:3b	30/30	30/30	5/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	30/30	30/30	30/30	30/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	30/30	0/30	30/30	30/30
gemma:7b	30/30	30/30	30/30	30/30
mistral:7b	0/30	0/30	0/30	30/30
llama2:7b	0/30	0/30	0/30	30/30
deepseek-llm:7b	0/30	0/30	0/30	30/30
deepseek-r1:8b	29/30	1/30	1/30	0/30
llama3:8b	30/30	30/30	30/30	30/30

Table 10: Einstein Riddle Success Rates (Symbolic Formats)

Model	S-expression	json	token-test
gemma3:270m	30/30	30/30	0/30
smollm:360m	0/30	0/30	0/30
qwen:0.5b	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30
deepseek-r1:1.5b	0/30	0/30	0/30
stablelm2:1.6b	0/30	0/30	0/30
qwen:1.8b	0/30	0/30	0/30
gemma:2b	30/30	0/30	0/30
falcon3:3b	0/30	30/30	30/30
llama3.2:3b	0/30	0/30	0/30
phi3:mini	0/30	0/30	0/30
gemma3:4b	0/30	0/30	0/30
qwen:4b	0/30	0/30	0/30
yi:6b	30/30	30/30	0/30
gemma:7b	0/30	30/30	0/30
mistral:7b	0/30	0/30	0/30
llama2:7b	0/30	0/30	0/30
deepseek-llm:7b	0/30	0/30	0/30
deepseek-r1:8b	0/30	0/30	0/30
llama3:8b	30/30	30/30	0/30

Table 11: Einstein Riddle Success Rates (CoT Formats)

Model	CoT (ja)	CoT (en)	CoT (eo)	CoT (jbo)
gemma3:270m	30/30	30/30	0/30	30/30
smollm:360m	0/30	0/30	0/30	0/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	0/30	0/30	0/30	0/30
stablelm2:1.6b	0/30	7/30	4/30	3/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	0/30	0/30	30/30	0/30
falcon3:3b	30/30	0/30	30/30	0/30
llama3.2:3b	0/30	0/30	0/30	30/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	0/30	30/30	0/30	0/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	0/30	0/30	0/30	30/30
gemma:7b	0/30	30/30	0/30	0/30
mistral:7b	0/30	0/30	0/30	1/30
llama2:7b	30/30	0/30	30/30	0/30
deepseek-llm:7b	0/30	0/30	0/30	0/30
deepseek-r1:8b	0/30	0/30	0/30	0/30
llama3:8b	0/30	0/30	30/30	0/30

Table 12: Filtered List Success Rates

Model / Language	ja	en	eo	jbo
gemma3:270m	0/30	0/30	0/30	0/30
smollm:360m	0/30	0/30	0/30	0/30
qwen:0.5b	0/30	0/30	0/30	0/30
tinyllama:1.1b	0/30	0/30	0/30	0/30
deepseek-r1:1.5b	0/30	0/30	0/30	0/30
stablelm2:1.6b	0/30	0/30	0/30	0/30
qwen:1.8b	0/30	0/30	0/30	0/30
gemma:2b	0/30	0/30	0/30	0/30
falcon3:3b	0/30	0/30	0/30	0/30
llama3.2:3b	0/30	30/30	0/30	0/30
phi3:mini	0/30	0/30	0/30	0/30
gemma3:4b	0/30	30/30	0/30	0/30
qwen:4b	0/30	0/30	0/30	0/30
yi:6b	0/30	0/30	0/30	0/30
gemma:7b	0/30	30/30	0/30	0/30
mistral:7b	0/30	0/30	0/30	0/30
llama2:7b	0/30	30/30	0/30	0/30
deepseek-llm:7b	0/30	0/30	0/30	0/30
deepseek-r1:8b	30/30	30/30	30/30	0/30
llama3:8b	30/30	30/30	0/30	0/30