NLP 2025 Spring Final
The influence of order bias and input-output format to the LLMs

Cheng-Ju, Tsai & Wei-Tse, Chin
Department of Computer Science and Information Engineering
National Taiwan University
NLP lecture team 6

Abstract

Large language models exhibit sensitivity to the order of options in multiple-choice questions, potentially undermining the reliability of model evaluations. This report presents a comprehensive analysis of order effects across three large language models (Gemini-2.0-Flash, Mistral-Small-2503, and Llama-v3p1-8b-instruct) using six analytical dimensions: model stability comparison, cross-linguistic patterns, shuffle method effectiveness, domain-specific vulnerabilities, input/output format sensitivity, and confidence-stability relationships. Through systematic evaluation employing Relative Standard Deviation (RSD), Standard Deviation of Recalls (RStd), and Fluctuation Rate (FR) as evaluation metrics, our findings reveal significant cross-linguistic differences in order sensitivity, domain-specific vulnerability patterns, and counterintuitive relationships between model confidence and response stability. All experimental data and results are available at https://drive.google.com/drive/folders/1i8t7PfIz_s5GWo-x5LxE28Z21rnS_xnc?usp=drive_link.

1 Introduction

The reliability of large language model (LLM) evaluations critically depends on the consistency of model responses across equivalent presentations of the same task. However, recent research has revealed that LLMs can exhibit significant sensitivity to seemingly irrelevant factors, such as the order of options in multiple-choice questions. This order sensitivity poses fundamental challenges for fair model comparison, benchmark reliability, and practical deployment scenarios where consistent performance is essential.

While previous studies have documented order effects in specific models or languages, comprehensive cross-model analyses examining the multidimensional nature of order sensitivity remain limited. This study aims to fill this gap by conducting a systematic evaluation across multiple models, languages, domains, and presentation formats, thereby establishing a thorough understanding of order effects in contemporary LLMs.

2 Experimental Design

2.1 Model Selection

We evaluated three representative large language models that span different architectural designs and parameter scales:

Model	Type	Provider
Gemini-2.0-Flash	Multimodal	Google
Mistral-Small-2503	Text-only	Mistral AI
Llama-v3p1-8b-instruct	Text-only	Meta

Table 1: Models evaluated in the study

2.2 Implementation Framework

We implemented our experimental pipeline using LangChain as the primary framework for API integration, input formatting, and automated inference across all model configurations. All API costs were covered by the research team to ensure unbiased evaluation conditions.

2.3 Experimental Configuration

Our experimental design covered multiple factors that influence order sensitivity:

Languages: English (en) and Japanese (ja-jp) to examine cross-linguistic patterns

Shuffle Methods: Instead of random shuffling, we designed semantic sorting methods to perform the shuffling. This approach aims to make the change in order more meaningful.

Four common shuffle methods:

- *default*: Original option order
- *reverse*: Reverse of original order
- *longest-first*: Sorted by option content length at the character level in descending order
- *shortest-first*: Sorted by option content length at the character level in ascending order

Although character level isn't how the model understands the text, sorting at the token level is a model-specific method that can result in different option orders for a question across models. Usually, users aren't aware of the token length in the prompt, but only of the string length. Therefore, we simply use character length for sorting.

For ja-jp, we introduce two more methods:

- *fewest-kana-ratio*: Sorted in ascending order of Japanese kana density, then by descending content length.
- *most-kana-ratio*: Sorted in descending order of Japanese kana density, then by ascending content length.

These methods focus on checking whether the sorting kana ratio introduces any bias.

In additional, these methods can serve as forward-backward pairs for computing the Fluctuation Rate.

Input/Output Formats: Our combinations of input presentation and output formatting:

- Input formats: Free text (base), JSON, XML
- Output formats: Free text (base), JSON-full, XML-full

"(Format)-full" means the response, including reasoning and answers, must be entirely in that format. When structured output is needed, most APIs restrict their responses to the full format. That's why we choose the "full" variant for the experiments.

We designed several prompts to produce different input/output formats across tasks.

```
{
    "Task": "Answer the following multiple choice question.",
    "Output_format": "",
    "Instruction": "Think step by step before answering.",
    "Question": "",
    "Options": {
        "A": "",
        "B": "",
        "C": "",
        "D": ""
    }
}
```

Figure 1: Unfilled template for JSON input format

```
<root>
  <Task>Answer the following multiple choice question.</Task>
  <Output_format />
  <Instruction>Think step by step before answering.</Instruction>
  <Question />
  <Options>
    <A />
    <B />
    <C />
    <D />
  </Options>
</root>
```

Figure 2: Unfilled template for XML input format

```
Your response should be of the following format:

{
    "Reasoning": "<Your step-by-step reasoning, written in one line>",
    "Answer": "<$LETTER>"
}
where LETTER must be one of ABCD. Output your answer in the valid JSON object.
Ensure the output is valid and parseable as JSON.
```

Figure 3: Prompt for JSON-full output format

```
Your response should be of the following format:
<root>
    <Reasoning>[Your step-by-step reasoning, written in one line]</Reasoning>
    <Answer>[$LETTER]</Answer>
</root>
where LETTER is one of ABCD. Only output the valid XML object, and nothing else.
Ensure the output is valid and parseable as XML.
```

Figure 4: Prompt for XML-full output format

We first divided the base prompt into several parts: task, output format, instruction, question, and options. These components were then used to define keys or tags for constructing prompt templates. To improve adherence to the expected output format, we also included additional instructions to guide the model's response.

2.4 Data Structure and Collection

Our result of experimental data is systematically organized in CSV format with the following structure in Table 2. Note that only the Llama model provides log probability outputs for the top-5 token candidates at each response position. We specifically recorded the probabilities for answer tokens (A, B, C, D) to analyze both the original correct answer's probability and alternative options' probabilities. When answer letters did not appear in the top-5 candidates, we assigned zero probability, as exclusion from top-5 indicates significantly lower probability. The log probabilities are stored in JSON format:

$$\{"A" : P_A, "B" : P_B, "C" : P_C, "D" : P_D\}$$

Field	Description
Model	Model identifier (gemini-2.0-flash, mistral-small-2503, Llama-v3p1-8b-instruct)
Language	Language code (en, ja-jp)
Subtask	All subtasks in MMMLU
Question ID	Unique identifier within each subtask
Shuffle Method	Reordering approach (default, reverse, shortest-first, etc.)
Original to Shuffled	Mapping from original option order to shuffled positions
Input Format	Question presentation format (xml, json, base)
Output Format	Expected response format (full, json, base)
Query	Complete formatted question text sent to model
Original Correct Answer	Correct option in original ordering
Shuffled Correct Answer	Correct option after shuffling
Response Answer	Model's selected option
Model Output	Raw model response text
Logprobs	Token-level log probabilities (Only available in Llama)

Table 2: Structure of experimental data CSV format

2.5 Metrics and Analysis

For each model-language-format-domain combination, we collected responses across all shuffle methods and calculated the Relative Standard Deviation (RSD) and accuracy scores:

$$\text{RSD} = \frac{\sigma_{\text{accuracy}}}{\mu_{\text{accuracy}}}$$

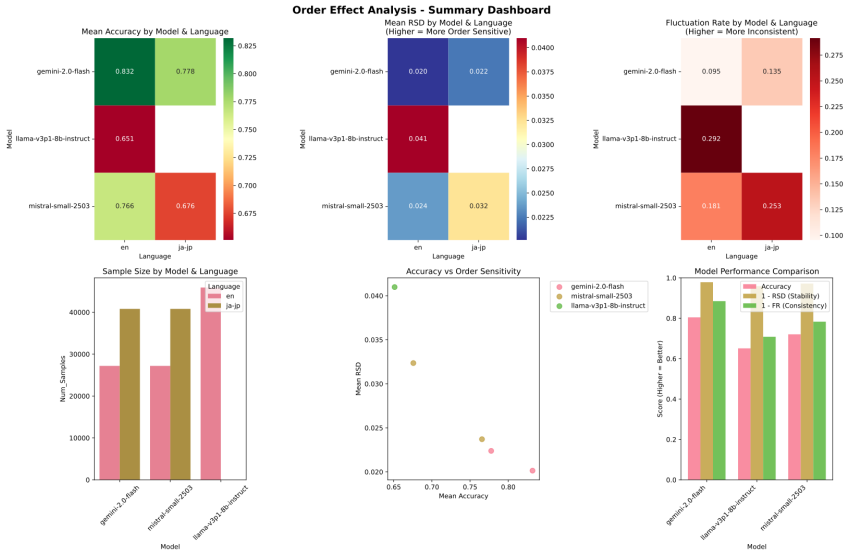

Figure 5: Model stability ranking showing in Accuracy, RSD, FR across all experimental conditions in two language.

where σ_{accuracy} is The standard deviation of accuracies for each option across different shuffle methods and μ_{accuracy} is the mean accuracy. RSD provides a normalized measure of variability that enables comparison across different performance levels.

Additionally, we measure the Fluctuation Rate to evaluate the consistency of each model across different languages.

$$FR = \frac{\sum_{i=1}^{N}(r_{forward}(i) \neq r_{backward}(i))}{N}$$

2.6 Experiment Details

Due to limited resources, we were unable to perform all combinations for all models. We assigned Gemini and Mistral to perform the core experimental conditions, while Llama was used for additional exploratory analysis.

- **Gemini-2.0-flash**: Both language, all shuffle methods, format: (base to base, base to JSON-full, JSON to JSON-full and XML to base)
- **Mistral-small-2503**: Both language, all shuffle methods, format: (base to base, base to JSON-full, JSON to JSON-full and XML to base)
- **Llama-v3p1-8b-instruct**: English only, three shuffle methods (default, reverse, shortest-first), 9 (3x3, base, JSON, XML) format combinations.

3 Results

3.1 Overall Model Stability Comparison

Figure 5 and Figure 6 presents the overall stability ranking across all experimental conditions present in two languages. The analysis reveals a clear hierarchy:

Gemini-2.0-Flash demonstrates superior performance across all metrics: highest accuracy (0.83 in English, 0.78 in Japanese), exceptional stability (median RSD of 0.020 in English, 0.022 in Japanese), and highest consistency (Fluctuation Rate of 0.095 in English, 0.135 in Japanese), indicating robust performance across different option orderings.

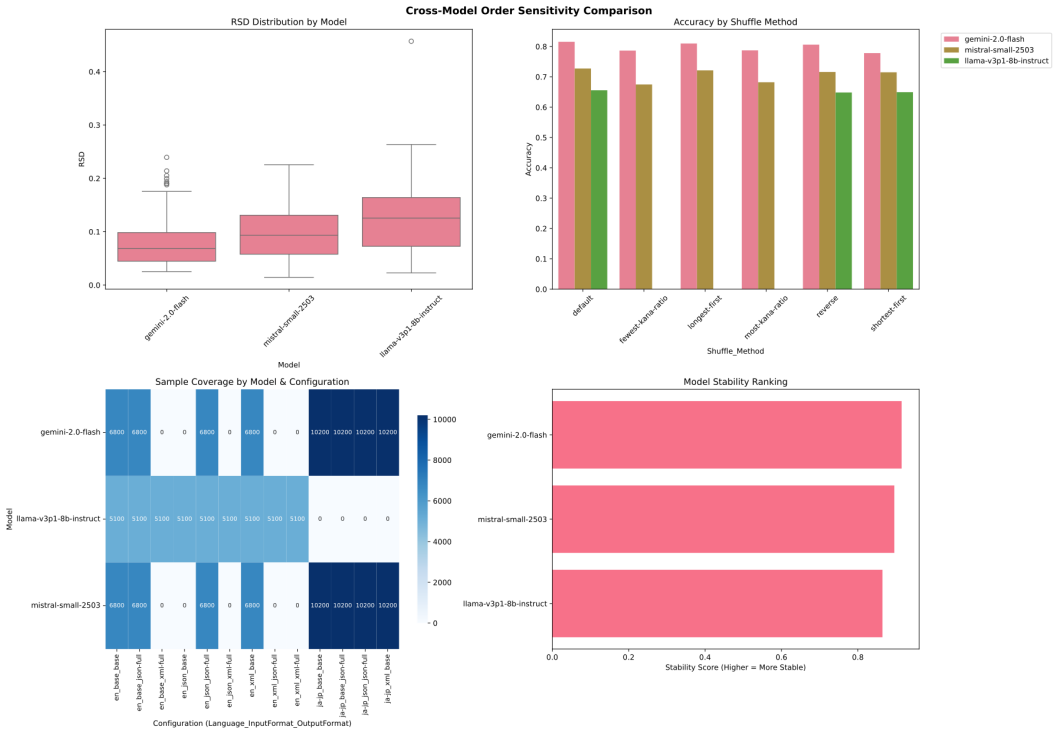

Figure 6: Comprehensive analysis dashboard showing model performance across accuracy, RSD, Fluctuation Rate, sample coverage, and stability metrics. This multi-panel view enables systematic comparison across all experimental dimensions.

Mistral-Small-2503 exhibits moderate performance with accuracy of 0.766 in English and 0.676 in Japanese, median RSD around 0.024 in English and 0.032 in Japanese, and moderate Fluctuation Rates (0.181 in English, 0.253 in Japanese), showing some sensitivity to order effects while maintaining reasonable consistency.

Llama-v3p1-8b-instruct displays the highest order sensitivity with accuracy of 0.649 in English, RSD values of 0.044 in English, and the highest Fluctuation Rate (0.294 in English), indicating substantial performance variations based on option ordering.

Notably, Japanese consistently shows lower accuracy, higher RSD, and higher Fluctuation Rates compared to English across both Gemini and Mistral models, likely due to factors such as relatively fewer language resources and other language-specific challenges, suggesting vulnerabilities to order effects.

3.2 Cross-Linguistic Order Sensitivity

The bilingual analysis (Figure 7) reveals systematic differences between English and Japanese across shuffle methods using RSD as the order sensitivity metric:

Japanese consistently exhibits higher order sensitivity than English across all models that support both languages. For Mistral-Small-2503, Japanese RSD values are frequently 1.5-2× higher than English equivalents.

Language-specific shuffle method effectiveness varies significantly. Methods based on linguistic features (kana-ratio shuffling) show different patterns between languages, suggesting that tokenization and linguistic structure influence order sensitivity.

Shuffle method effectiveness in English: Since Llama model experiments were conducted only in English, we can compare the effectiveness of three shuffle methods exclusively in this language. The results show that shortest-first ordering produces the lowest RSD values,

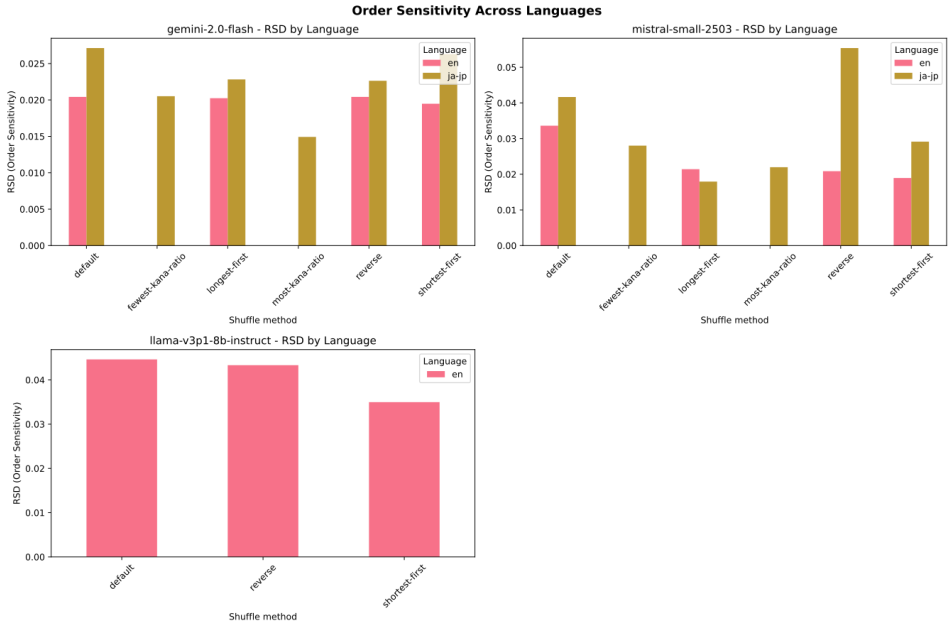

Figure 7: Cross-linguistic comparison of order sensitivity. Japanese consistently shows higher RSD values than English across all shuffle methods.

reverse ordering generates the highest RSD, and default ordering falls between them. This pattern suggests that *length-based ordering may provide inherent stability cues*, while reverse ordering maximally disrupts learned positional biases.

3.3 Domain-Specific Vulnerability Analysis

The subtask analysis (Figure 8) identifies consistent patterns of domain-specific order sensitivity:

High-sensitivity domains across models include:

- Security
- Virology
- Machine learning
- Global facts

Complex reasoning domains demonstrate higher order sensitivity, suggesting that cognitively demanding tasks are particularly vulnerable to order effects in smaller models, potentially due to increased reliance on superficial ordering cues when reasoning resources are limited.

3.4 Input/Output Format Effects

Figure 9 shows format combination analysis revealing model-specific patterns in order sensitivity, while Figure 10 demonstrates how these format combinations affect overall model accuracy.

Model Performance Hierarchy: The accuracy analysis reveals a clear performance hierarchy, with Gemini-2.0-Flash and Mistral-Small-2503 achieving 69-81% accuracy across format combinations, while Llama-v3p1-8b-instruct operates at a lower range of 64-70%. This performance gap persists regardless of format configuration.

Format Sensitivity Patterns:

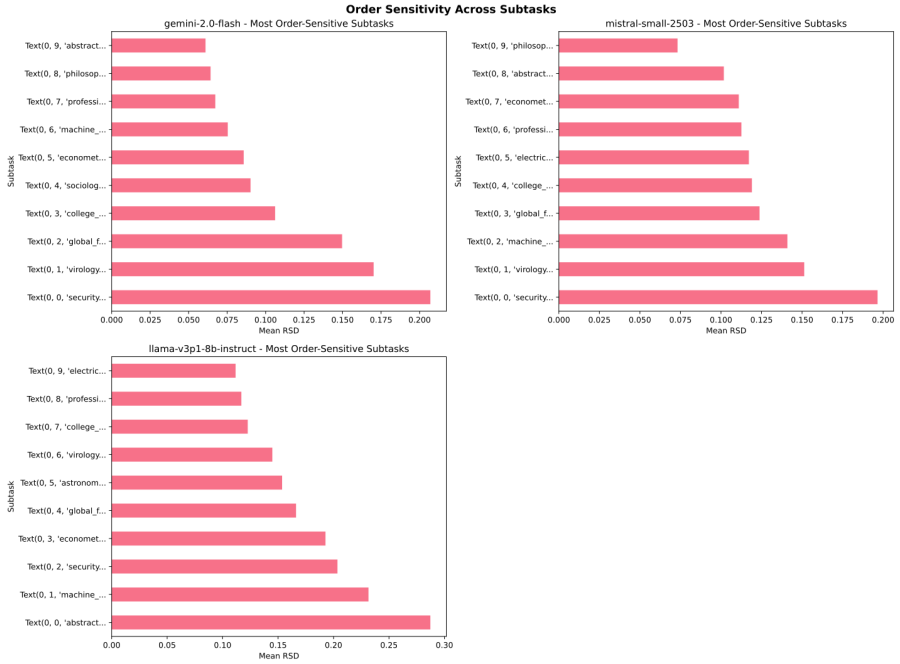

Figure 8: Domain-specific order sensitivity analysis showing the most vulnerable subtasks for each model. Security and virology consistently appear as high-sensitivity domains.

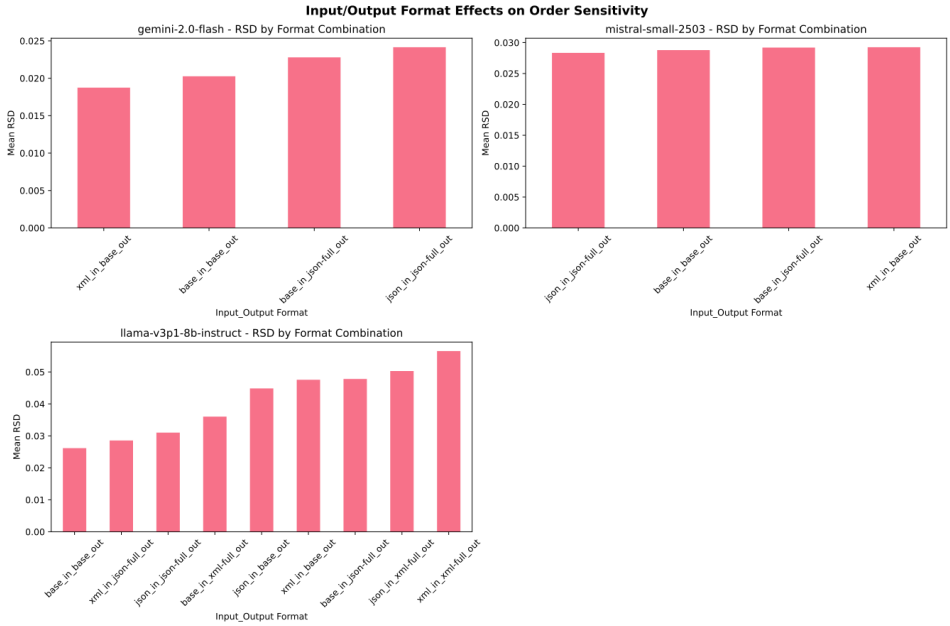

Figure 9: Input/output format combination effects on order sensitivity across models. Different format combinations show varying degrees of stability.

- **Gemini-2.0-Flash** demonstrates both high accuracy and low order sensitivity (RSD ¡ 0.025), with base_base and xml_base combinations yielding optimal performance (81% accuracy).

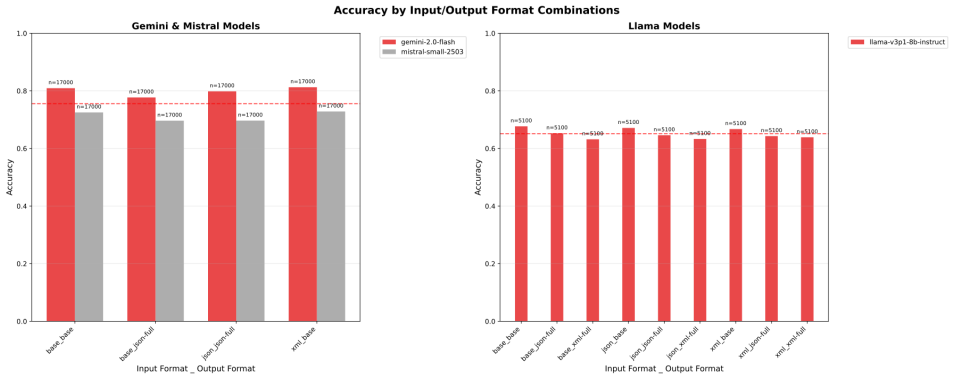

Figure 10: Accuracy comparison across input/output format combinations. Gemini and Mistral models (left) show higher overall performance and greater format sensitivity compared to Llama models (right).

- **Mistral-Small-2503** shows consistent behavior with uniform RSD values (0.029) but exhibits slight accuracy variations, performing best with base_base combinations (74% accuracy).
- **Llama-v3p1-8b-instruct** displays significant format dependency in order sensitivity, with base_in_json_full_out producing high RSD values (¿0.065), yet maintains relatively stable accuracy across its nine format combinations (64-70%).

Structured vs. Base Format Effects: Contrary to expectations, structured formats (JSON/XML) do not universally improve performance. Base input with base output consistently ranks among the top-performing combinations for Gemini and Mistral models. However, structured output formats appear to provide more consistent cross-model behavior, suggesting their value lies in standardization rather than accuracy enhancement.

Format Robustness: The analysis indicates that simpler format combinations (base_base, xml_base) demonstrate superior accuracy-stability trade-offs, while complex structured combinations may introduce additional variability without commensurate performance gains.

3.5 Confidence-Stability Relationship

3.5.1 Llama

Figure 11 presents the confidence analysis for Llama-v3p1-8b-instruct, revealing a counterintuitive pattern: higher model confidence correlates with lower accuracy standard deviation (higher stability).

The plot shows three distinct horizontal clusters, representing different accuracy performance levels (approximately 0.0, 0.57, and 0.71), with the declining red trend line indicating that higher-confidence predictions exhibit lower standard deviation in accuracy across different shuffle methods. This suggests that when the model is more confident in its responses, those responses are less susceptible to order-induced variations, potentially indicating that confidence serves as a indicator for order-robust predictions.

3.5.2 Gemini & Mistral

For Gemini and Mistral, since reliable probability are unavailable from the APIs, we used high-temperature answer distributions with the base prompt (base input, base output, default order) as a proxy for confidence.

First, we set the maximum temperature for each model (2.0 for Gemini, 1.0 for Mistral), and sampled 10 times to form the answer distribution. Each answer falls into one of five categories: 'A', 'B', 'C', 'D', or 'Others (invalid)'.

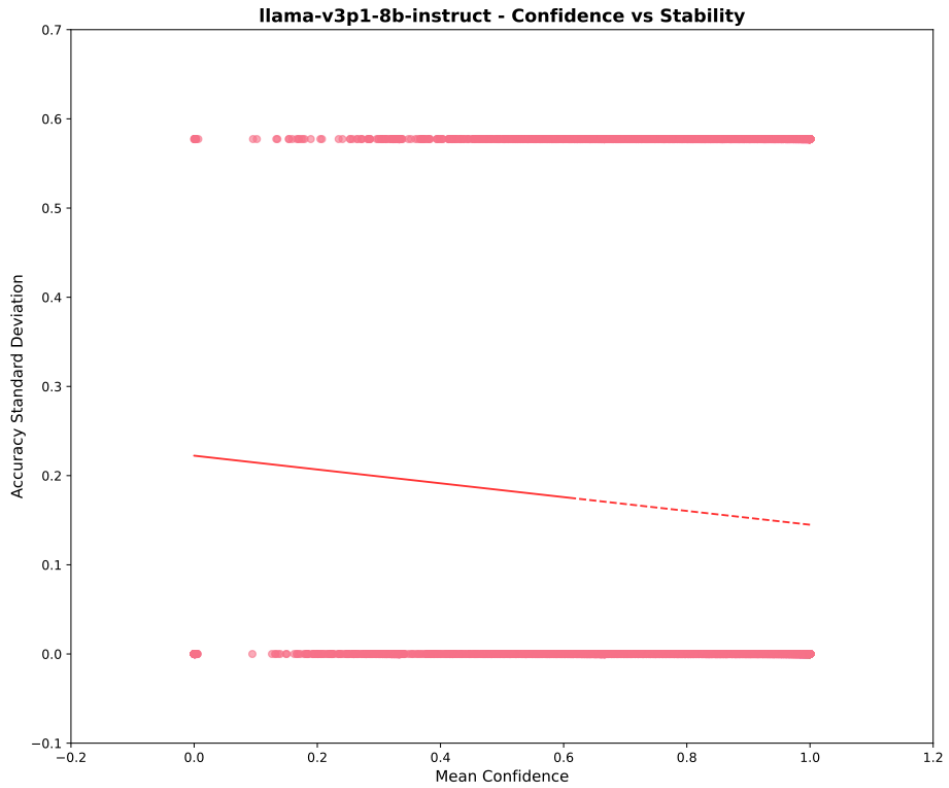

Figure 11: Confidence vs. stability relationship in Llama-v3p1-8b-instruct. The declining red trend line shows that higher confidence correlates with more stable responses across different orderings. Horizontal clusters represent different accuracy performance levels, while the trend line indicates that confident responses show less variation due to ordering changes.

Then, we have two methods to compute the confidence from that distribution:

- **Maximum probability**: Defined as the empirical probability (relative frequency) of the most frequent response across samples.
- **Normalized entropy**: Confidence is computed as $1 - H_{norm}$, where higher entropy indicates lower confidence.

The normalized entropy H_{norm} is defined as:

$$H_{norm} = \frac{H}{\log N}$$

such that its values range from 0 to 1. Here, N is the number of options. In our case, we have 4 options plus an 'Others', so $N = 5$.

Similarly, in the case of normalized answer entropy, the distributions are formed from model outputs generated using varied shuffle methods and formats.

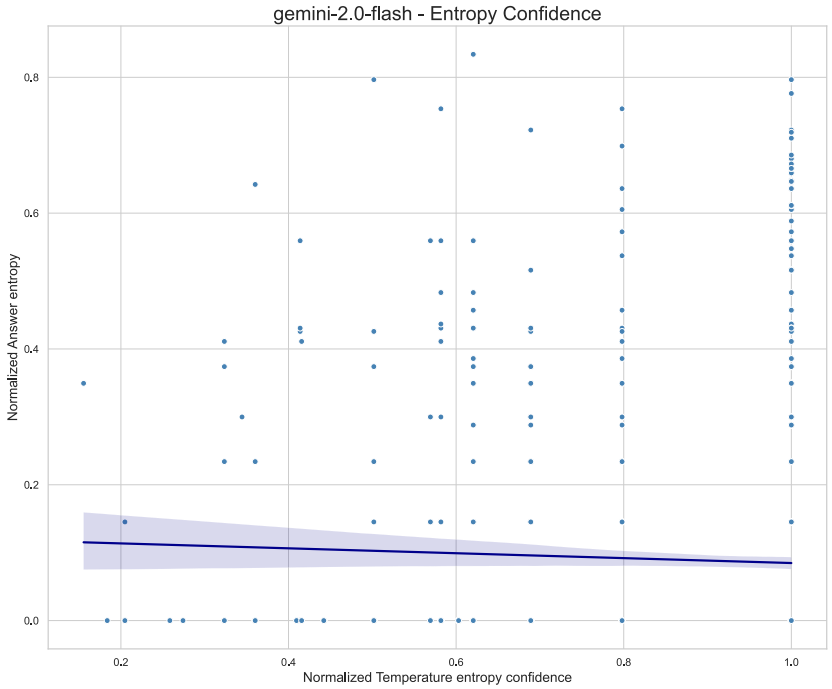

Figure 12: Gemini - Entropy Confidence to the answer entropy in English

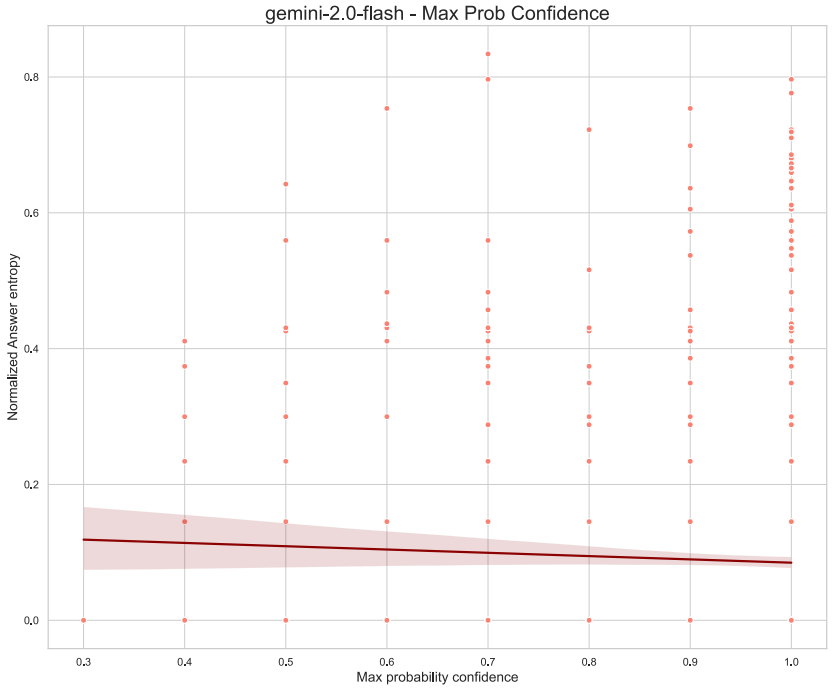

Figure 13: Gemini - Max prob Confidence to the answer entropy in English

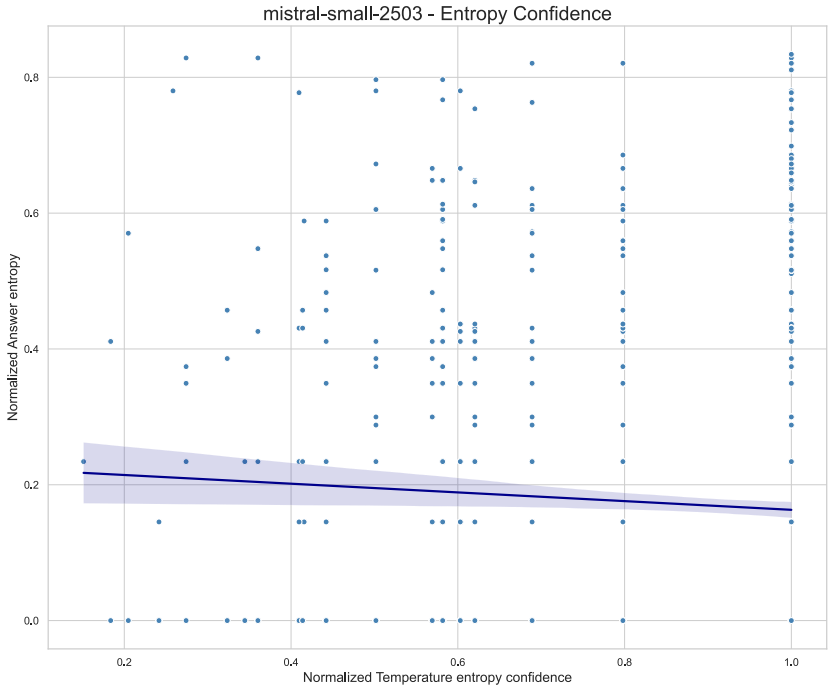

Figure 14: Mistral - Entropy Confidence to the answer entropy in English

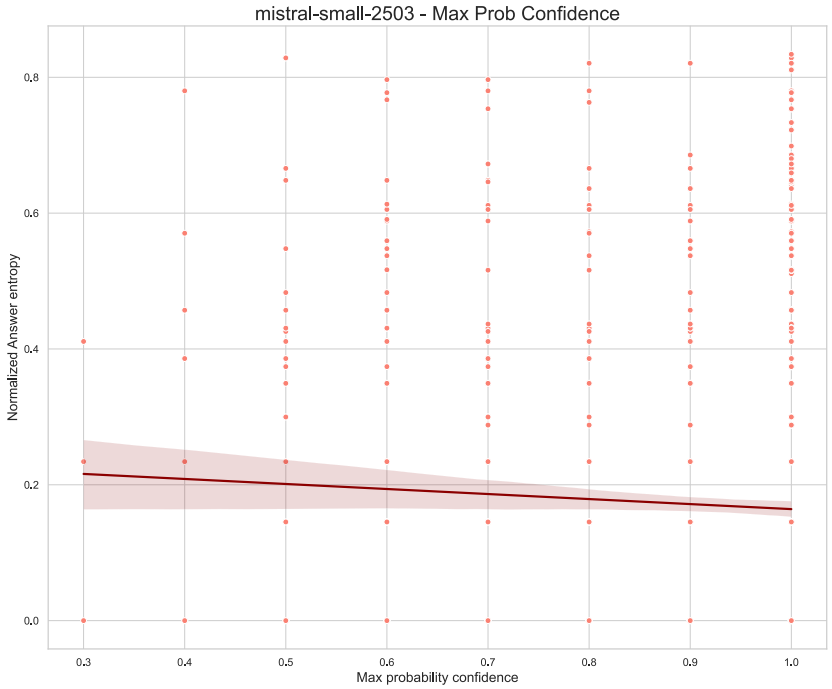

Figure 15: Mistral - Max prob Confidence to the answer entropy in English

Even though similar declining trends can be observed, the confidence in this approach is limited by our sample size. Questions that the model had high confidence in under the base setting still suffer from order and format bias. We cannot rely on confidence alone to

determine the presence of bias in a given question. For instance, a confidence score of 1 does not guarantee low answer entropy.

3.6 Shuffle Method Effectiveness

Across all models, the *reverse* shuffle method consistently produces higher order sensitivity than the default ordering, while length-based shuffling methods show variable effectiveness depending on the model and language combination.

3.7 The Bias on the Invalid Responses

We observed that Gemini-2.0-flash can easily get stuck in repetition. It tends to output the same patterns or sentences repeatedly until reaching the maximum output length sometimes.

To examine the relationship between order, formats, and the repetition phenomenon, we count the invalid (non-extractable) answers across different combinations.

In the base setting (base-in, base-out, default order), Gemini-2.0-flash exhibits the repetition phenomenon only in some cases. However, after we change the formats or shuffle the order, Gemini-2.0-flash begins to avoid choosing any option, without repetition, if it believes none are correct or if it thinks it cannot answer the question.

Moreover, Llama3 showed persistent repetition behavior in approximately 25% of cases across all input/output format combinations (base, JSON, XML), often resulting in responses that exceeded the maximum token limit of 4,096 tokens without providing a definitive answer. To address the repetition issue, we implemented an adaptive retry mechanism with progressively more directive prompts when models exhibited repetitive behavior or reached the token limit:

```
retry_instructions = [
    "\n\nSkip the reasoning steps, give the answer directly.",
    "\n\nProvide only the final answer without explanation.",
    "\n\nBe concise and direct in your response.",
    "\n\nAnswer briefly without showing work."
]
```

The order/format bias may influence the willingness to answer. Overall, formatted output appears to result in more invalid answers, suggesting that the output format might have a greater impact on Gemini. Japanese has more invalid answers than English, as expected, since English is the model's primary and more well-supported language.

Compared to Gemini, Mistral has only a few invalid examples. The order and formats do not seem to have much influence on Mistral. This may indicate that Gemini is more cautious in its decisions.

```
{
  "Reasoning": "I do not have access to external websites or specific files,
 including those that would contain statistics about internet usage in Brazil in 2017.
  Therefore, I cannot definitively answer this question."
  "Answer": "I cannot answer"
}
```

Figure 16: An example of a refusal response where Gemini-2.0-flash did not choose any option and did not fall into repetition.

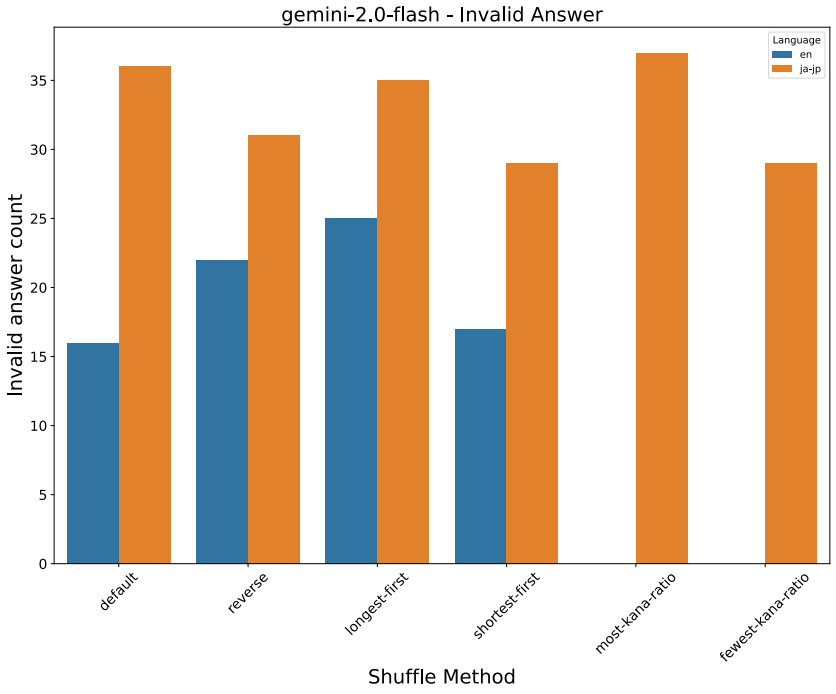

Figure 17: Invalid answer counts across shuffling methods in Gemini-2.0-flash

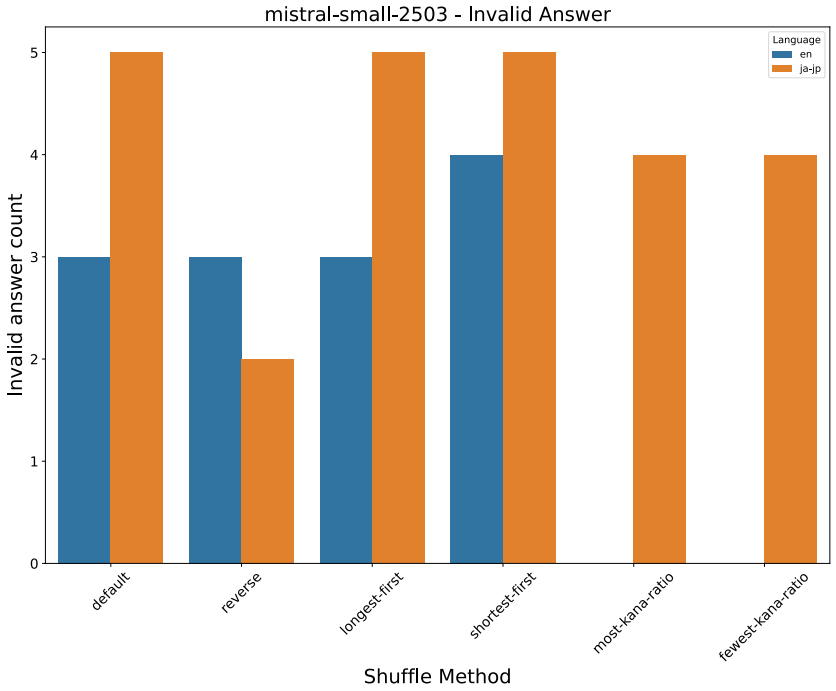

Figure 18: Invalid answer counts across shuffling methods in Mistral-small-2503

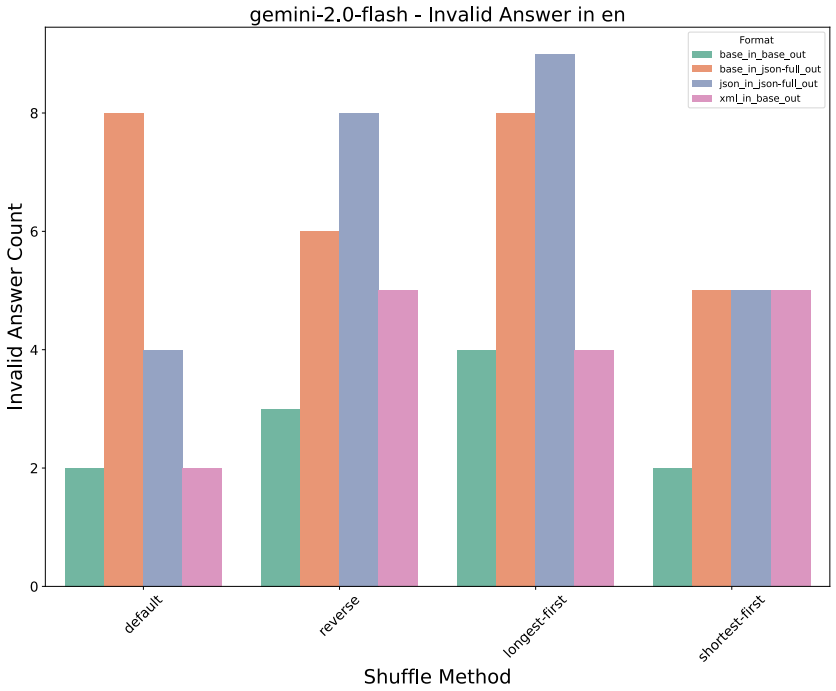

Figure 19: Invalid answer counts in English across the formats in Gemini-2.0-flash

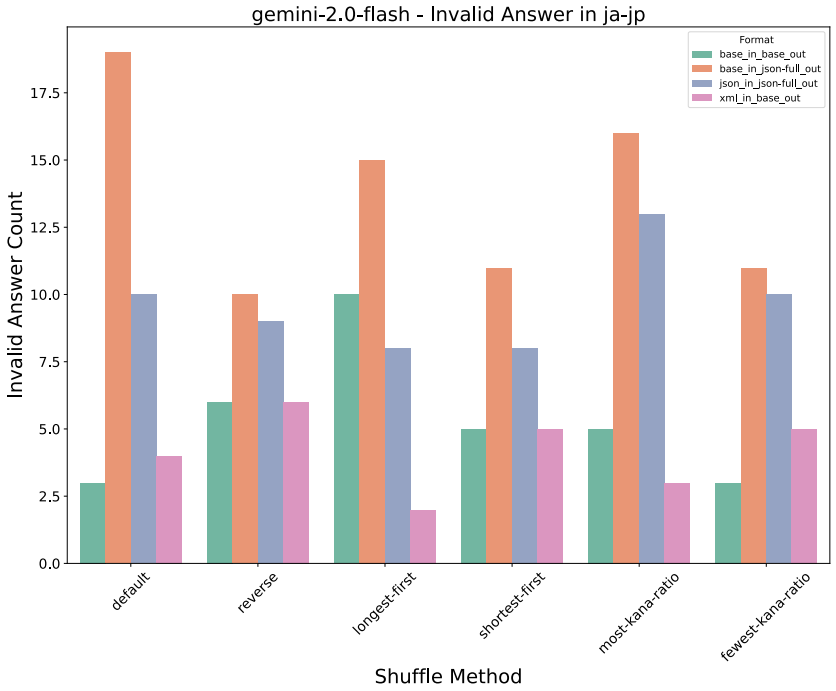

Figure 20: Invalid answer counts in Japanese across the formats in Gemini-2.0-flash

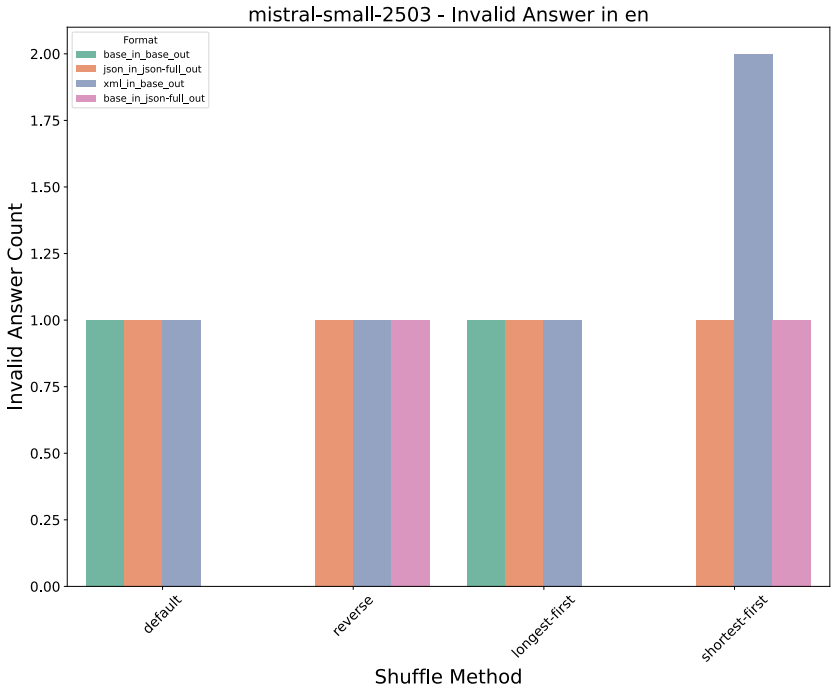

Figure 21: Invalid answer counts in English across the formats in Mistral-small-2503

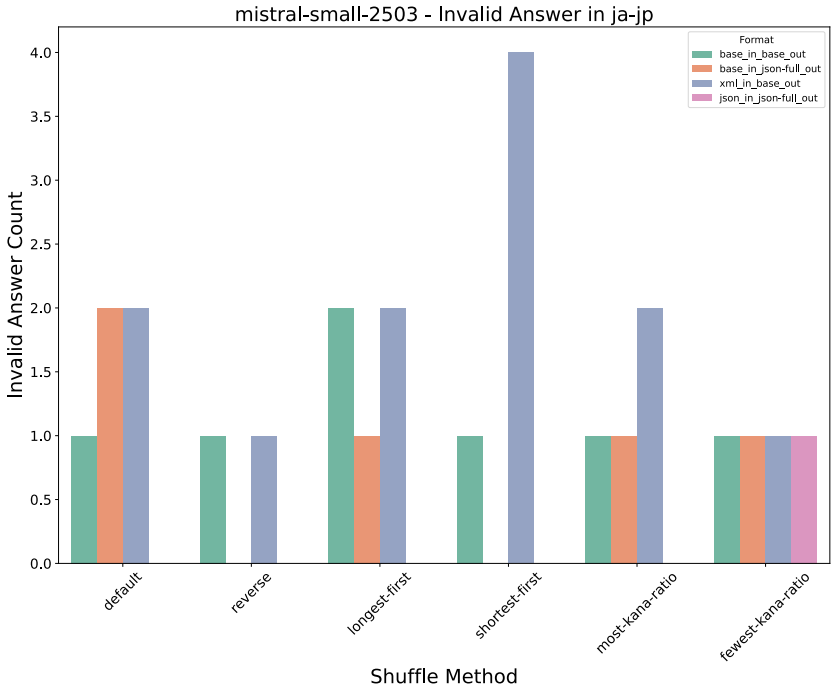

Figure 22: Invalid answer counts in Japanese across the formats in Mistral-small-2503

4 Discussion

4.1 Implications for Model Evaluation

The clear stability hierarchy (Gemini $>$ Mistral $>$ Llama) has important implications for evaluation protocol design. Higher-performing models in terms of raw accuracy also demonstrate greater consistency across different presentations of the same content, suggesting that robustness and capability may be related dimensions of model quality.

The substantial language differences in order sensitivity indicate that multilingual evaluations require additional robustness considerations. The consistently higher sensitivity observed in Japanese suggests that evaluation frameworks developed primarily for English may underestimate reliability challenges in other languages.

4.2 Domain-Specific Considerations

The identification of high-sensitivity domains (security, virology, machine learning) suggests that evaluation in these areas requires particular attention to presentation format consistency. The domain-specific patterns may reflect differences in training data distribution, question complexity, or reasoning requirements.

4.3 Practical Recommendations

Based on our findings, we recommend:

1. **Similar question re-evaluation**: Designing the QA dataset to include questions with similar descriptions and answers, but differing in the order of options.
2. **Multi-order evaluation**: Standard evaluation protocols should include multiple option orderings, particularly for high-stakes assessments
3. **Language-specific calibration**: Non-English evaluations should account for potentially higher order sensitivity
4. **Domain awareness**: In sensitive fields, evaluations need extra consistency checks.
5. **Format standardization**: Consistent input and output formatting can enhance evaluation reliability, especially for models exhibiting high sensitivity to formatting.

5 Limitations

We lack a greater variety of format combinations for Gemini and Mistral. Additionally, confidence measurements are limited by the sample size and are assessed indirectly, relying on the answer distributions under the base settings. Due to the large data size and limited resources, we were unable to select interesting examples for closer analysis.

6 Conclusion

This comprehensive analysis establishes systematic patterns of order sensitivity across contemporary large language models, revealing significant implications for evaluation reliability and practical deployment. Our evaluation of three representative LLMs across multiple dimensions demonstrates a clear stability hierarchy: Gemini-2.0-Flash exhibits superior robustness, followed by Mistral-Small-2503, while Llama-v3p1-8b-instruct shows the highest sensitivity to option ordering.

Key contributions of this study include the identification of cross-linguistic vulnerabilities, with Japanese consistently exhibiting 1.5 to 2 times higher order sensitivity than English, and the discovery of domain-specific vulnerability patterns in security, virology, and machine learning tasks. Our analysis reveals that input/output format combinations significantly impact model stability, with structured formats generally improving consistency across models.

The relationship between model confidence and response stability is complex; confidence metrics alone cannot fully indicate the presence of order-induced bias. This suggests that relying solely on confidence is insufficient for bias identification, underscoring the need for more robust evaluation criteria.

These findings demonstrate that order sensitivity is a fundamental challenge rather than a mere technical issue, requiring careful attention in evaluation and deployment. The observed patterns provide practical guidance for developing stronger evaluation methods and highlight the importance of multi-dimensional consistency testing.

Future research should extend these analyses to emerging model architectures, investigate the underlying causes of order sensitivity, and develop mitigation strategies that preserve performance while improving consistency. As language models become increasingly integrated into critical applications, ensuring consistent and reliable responses across equivalent task presentations is essential for maintaining trust and effectiveness.