On the Brittleness of LLMs: A Journey around Set Membership

Lea Hergert[1] **Gábor Berend**[1]

Mario Szegedy[2] **György Turán**[3,4] **Márk Jelasity**[1,4]

[1] University of Szeged, Hungary
[2] Rutgers University, USA
[3] University of Illinois at Chicago, USA
[4] HUN-REN–SZTE Research Group on AI, Hungary

Abstract

Large language models (LLMs) achieve superhuman performance on complex reasoning tasks, yet often fail on much simpler problems, raising concerns about their reliability and interpretability. We investigate this paradox through a focused study with two key design features: simplicity, to expose basic failure modes, and scale, to enable comprehensive controlled experiments. We focus on set membership queries—among the most fundamental forms of reasoning—using tasks like "Is apple an element of the set {pear, plum, apple, raspberry}?". We conduct a systematic empirical evaluation across prompt phrasing, semantic structure, element ordering, and model choice. Our large-scale analysis reveals that LLM performance on this elementary task is consistently brittle, and unpredictable across all dimensions, suggesting that the models' "understanding" of the set concept is fragmented and convoluted at best. Our work demonstrates that the large-scale experiments enabled by the simplicity of the problem allow us to map and analyze the failure modes comprehensively, making this approach a valuable methodology for LLM evaluation in general.

1 Introduction

While LLMs show impressive performance, it is important to understand their various deficiencies as well (McCoy et al., 2024; Cuskley et al., 2024; Passerini et al., 2024; Shojaee et al., 2025; György et al., 2025).

One such potential deficiency is *brittleness* that, interestingly, has often been cited as a main criticism of *classical AI* (e.g., expert systems) that is unable to handle the flexibility of human intelligence and deal with the imperfect (noisy, incomplete) nature of knowledge and reasoning (Mitchell, 2021). Recently, *LLMs have been shown to be quite brittle as well*, but from different aspects. There is a growing literature on the failure of LLMs to handle simple tasks such as common sense inference, counting, and so on (Huckle and Williams, 2025; Zhou et al., 2024; Lewis and Mitchell, 2025).

A particularly interesting source of such brittleness is semantics interfering with reasoning. Human performance on simple reasoning tasks is known to be influenced by semantics, and this has been observed for LLMs as well on simple natural language inferences, syllogisms, and the Wason selection task (Wason, 1968; Dasgupta et al., 2022; Lampinen et al., 2024). A related phenomenon called semantic leakage has also been described for instruction-tuned models (Gonen et al., 2025).

To study brittleness, we focus on a task where it is reasonable to expect perfect and robust performance: set membership in small, explicitly listed sets. The task is well-motivated also because the set is, perhaps, the most fundamental concept in human thinking, both for commonsense reasoning and mathematics. The questions we will ask will be as simple as *"Is apple an element of the set {pear, plum, apple, raspberry}?"*. This task is of utmost simplicity. As far as we know, it is simpler than the basic reasoning tasks studied in the related literature. Therefore, any mistakes are interesting and could reveal relevant insights into fundamental LLM behavior and failure modes.

The simplicity of the task makes it possible to perform a systematic, thorough experimental analysis. We explore a large space of prompts (including natural language and Python coding) using a diverse collection of sets with and without semantic relationships. Overall, more than 60 million set membership queries are tested using 7 instruction-tuned LLMs.

Our contributions are the following.

- We reduce the task presented to the LLM to great simplicity (set containment) to enable a large-scale, holistic investigation.
- We design a systematic set of experiments aimed at multiple possible sources of brittleness and

study them in conjunction; this could serve as a benchmark for future evaluations as well.
- We map a diverse set of error patterns, including *high sensitivity to element ordering, minor prompt variations, and semantic relations*, none of which should play a role in our task.
- We show that the semantic relatedness of set elements can be both an advantage (semantic boosting) and a disadvantage (semantic leakage) in certain cases.
- We demonstrate that different LLMs exhibit different, and often wildly dissimilar error patterns.

2 Related Work

In this section we give a brief overview of some of the work most relevant to our paper, from the vast related literature.

Prompt sensitivity is studied in (Sclar et al., 2024) by formulating a grammar for prompt formatting and evaluating samples of prompt formats over a wide range of tasks. Another framework is presented in (Zhuo et al., 2024), evaluating prompts on the instance level. Our approach expands on these works in that we focus on a single basic problem, set membership, and study prompt sensitivity along with many other dimensions relevant to the general brittleness of LLMs.

Research into the **consistency of world models** is also relevant, as our results also suggest that LLMs have *no consistent models of small sets*. (Wolfram and Schein, 2025) consider city populations and isotope half-lives, and find that responses are relatively consistent over different prompts, but often inconsistent on related pieces of information. Other papers in the same direction are (Elazar et al., 2021; Fierro and Søgaard, 2022; Sahu et al., 2022; Raj et al., 2023; Zheng et al., 2024). Invariance can be another required property (Benton et al., 2020), with permutation invariance being a prominent example (Egressy and Stühmer, 2025; Ravanbakhsh et al., 2017; Zaheer et al., 2017). In our case, responses to a set membership query should be invariant under permutations of the elements in the set, but we find that they are not.

In **cognitive science**, the mental models theory of (Johnson-Laird, 1983) is used in (Khemlani et al., 2014) to form a theory of human reasoning about set membership for a class of syllogisms, and present some experiments confirming the predictions of the theory. In psychology, the recognition problem[1] (Yonelinas et al., 2022) is studied using techniques like response times (Dewhurst et al., 2006) and neuroimaging (Wais et al., 2010). Students' acquisition of the set concept has been studied in (Razmjooei, 2013).

3 Experimental Setup

We study very basic set membership queries to reveal and understand the counter-intuitive mistakes that otherwise capable LLMs make over such a simple task. We use simple prompts based on natural language or Python code snippets, such as the two examples below:

```
Does the set {"Hearts", "Diamonds", "Clubs", "Spades"}
contain the element "North"?
```

```
Given the Python code below, what will be the value of the
‘result‘ variable?

“‘python
S = {"North", "South", "West", "East"}
result = "North" in S
“‘
```

Our experimental setup is based on a thorough, well-informed construction of a fixed collection of 22 sets and a large set of prompts that we generate from templates, as described later on.

3.1 Sets

We decided to work with sets of exactly four elements. Our preliminary experiments revealed that increasing the size of the set increases the error rates. A size of four is still very small but already allows for observing interesting patterns.

We defined 22 sets of 5 types: complete, related word, related number, unrelated word, and unrelated number. The list of our 22 sets is given in the Supplementary material. Here, we give the motivation for each type.

Complete word sets consist of four words that exhaustively cover a category such as {Hearts, Diamonds, Clubs, Spades}. We identified 10 such sets using preliminary experiments, where we tested candidate sets for completion: we removed one element and asked a set of LLMs what the missing element is. The selected 10 sets were completed by each LLM we tested without error, for every possible missing element, and every ordering of the set.

[1] "In its simplest form, sometimes referred to as "item recognition", subjects are first asked to study a list of items such as words, objects, or images. Then, after a delay, they are presented with a mixture of studied and nonstudied (i.e., new) items and are required to use their memory to indicate whether each item had been in the studied list or is new." (Yonelinas et al., 2022).

Related sets contain words or numbers that are similar but do not form a complete set (e.g. *some* set of four fruits, or *some* numbers divisible by 100, etc.). We included 3 such word sets and 3 number sets.

Unrelated sets contain words or numbers that have no obvious commonality or relation. We included 3 such word sets and 3 number sets. It is not evident that there is indeed no connection among a given set of numbers or words, so we tested a number of LLMs asking for commonalities within our sets of unrelated elements, and no meaningful connections were suggested.

3.2 Queries

Query types. Let us assume that we are given a set $S = \{x_1, x_2, x_3, x_4\}$. We define the following query types.

- *Positive:* We select an element $x_i \in S$ and ask whether $x_i \in S$. We do this for all $i \in \{1, 2, 3, 4\}$.
- *Negative-member:* We select an element $x_i \in S$ and ask whether $x_i \in S \setminus \{x_i\}$. We do this for all $i \in \{1, 2, 3, 4\}$.
- *Negative-intruder:* We select $y \notin S$ and ask whether $y \in S$. This query type is implemented only for related and complete sets, and intruder y is selected from a different set of the same type (word or number) at random.

Permutation. Clearly, the order of the presentation of the elements of a set *should not have any effect on the answer of the LLM*, if the LLM correctly understands the concept of a set. Yet, the elements still have to be presented in a fixed sequential order in every prompt. To examine the effect of this ordering, we test the queries over all the possible permutations of the set.

Query instances. A query is given by $[(z_1, \ldots, z_n), z]$, where $n = 3$ or $n = 4$ depending on the query type, the list $(z_1, \ldots, z_n)$ contains the elements of the set in the order they are to be listed, and z is the element we query for set membership.

3.3 Prompts

Templates. A prompt is created by the instantiation of a *prompt template* for a given query. For example, a prompt template could be `'Does the set { "`z_1`", "`z_2`", "`z_3`", "`z_4`" } contain the element "`z`"?'` Applying this template to the query instance [(Hearts, Diamonds, Clubs, Spades), North] results in the first prompt we presented earlier.

Template classes. We define four prompt template classes: two natural language classes (NL1 and NL2) and two Python code classes called CS (coding simple) and CA (coding algorithmic). Each class is defined by an abstract template in which a number of features can be selected from a list of options. Then, all the feature combinations are used to generate prompt instances for a given query.

The **NL1 template class** consists of two schemes:

```
Does the set|list <set> contain|include element|item|character sequence|string <element>?
```

```
Is the element|item|character sequence|string <element> included|contained in the set|list <set>?
```

The features that are defined for the NL1 class are the following:

- *arrangement*: set-first or element-first scheme is used
- *set*: set | list
- *element*: element | item | character sequence | string
- *contain*: include | contain (set-first scheme); or included | contained (element-first scheme)
- *quotation*: set elements are quoted as Hearts, 'Hearts', or "Hearts", using identical settings in the set definition `<set>` and the element definition `<element>`
- *listing*: defines how the set is presented. It has 5 options. The first three options are comma-separated lists enclosed in either { }, (), or []. The remaining two options are bullet point listings with bullet types '-' or '*'.

The **NL2 template class** also consists of two schemes:

```
A set|list consists of the following elements|items|character sequences|strings: <set> Does this set|list include|contain the element|item|character sequence|string <element>?
```

```
Does a set|list include|contain the element|item|character sequence|string <element> if it consists of the following elements|items|character sequences|strings: <set>?
```

The features of the NL2 class are identical to those of the NL1 class, noting that the *set* and *element* features assign identical values to each occurrence of the given feature within the prompt.

The **CS and CA Python template classes** contain more variation, the full description is given in the supplementary material. The main difference between the two classes is that in the case of CS the set membership is decided by Python's `in` operator, while in the case of CA a `for` loop is used to iterate through the data structure to determine membership.

For illustration, an example CS template is

````
Given the Python code below, what will be the value of the
‘result‘ variable?

“‘python
my_set = <set>
result = <element> in my_set
“‘
````

Replacing the computation of the `result` variable with

```
result = False
for set_item in my_set:
    if set_item == <element>:
        result = True
```

results in a CA template. To generate such templates, similarly to the NL schemes, we define Python schemes with features such as whether we have a function named `contains` to test for set membership, whether we have a docstring documenting the Python code, as well as several initialization methods for the data structure containing the set, and so on. Both the CS and CA template class contains 960 templates.

Notes on prompt engineering. We simply defined a large and diverse set of prompts that we considered unambiguous and intuitive, without using any prompt engineering techniques explicitly. The reason is that our goal was not to achieve perfect accuracy, but to investigate the error patterns of LLMs under extremely simple and clear task definitions. Still, some of our prompt schemes produced perfect accuracy (see section 4.4) justifying our prompting approach.

3.4 Practical Notes

The **number of prompt instances** is given by the number of queries multiplied by the number of prompt schemes. The number of queries is given by

$$3024 = \overbrace{16 \cdot (4! \cdot 4 + 3! \cdot 4 + 3! \cdot 4)}^{\text{semantically related queries}} + \underbrace{6 \cdot (4! \cdot 4 + 3! \cdot 4)}_{\text{semantically unrelated queries}},$$

where we have 16 semantically related and 6 unrelated sets, we considered positive, negative-intruder, and negative-member types, and $x!$ is the number of permutations of x elements. The number of prompt instances is thus

$$8,709,120 = 3024 \cdot (480 \cdot 2 + 960 \cdot 2),$$

where 480 is the number of prompt schemes in the NL1 and NL2 classes and we have 960 in the CS and CA classes.

Model	Accuracy	Member FPR	Intruder FPR
Llama-3.2-3B	94.670%	5.716%	4.363%
Llama-3.1-8B	99.470%	1.541%	0.594%
Mistral-24B	98.512%	0.019%	0.000%
Qwen2.5-32B	99.892%	0.134%	0.004%
Phi-3.5-MoE	98.832%	0.587%	0.191%
Llama-3.1-70B	99.467%	0.030%	0.036%
Llama-3.3-70B	99.462%	0.073%	0.076%
Average	98.615%	1.157%	0.752%

Table 1: Statistics of LLM performance.

Evaluation. We insert the prefix ‘`Answer with a single word.`’ in front of every prompt. We then search the LLM’s answer for negative (false, no) and positive (true, yes) words. If both or none of these answers are found, then the answer is undecided; these represent about 0.14% of the answers and were treated as incorrect. Almost every such undecided answer was found in the NL1 and NL2 prompt categories, there were none in the CA category and 11 cases (out of more than 20 million) in the CS category.

4 Experimental Results

The experiment design described in section 3 was executed on Llama-3.2-3B, Llama-3.1-8B, Llama-3.1-70B, and Llama-3.3-70B (Dubey et al., 2024), Mistral-24B (Jiang et al., 2023), Qwen2.5-32B (Team, 2024), and Phi-3.5-MoE (Abdin et al., 2024). Each model is instruction-tuned.

Overall, we executed more than 60 million prompts forming a complete grid covering every possible combination of sets, prompts, query types, and LLMs (see section 3.4). The overall accuracy of each LLM is shown in table 1. As we can see, performance is reasonably good overall, but since each LLM was tested 8,709,120 times, the small error rate still indicates a large number of errors (about 790,000 altogether). In the following, we will focus mostly on these errors.

Motivation. We will examine in more detail three dimensions, to which *the set membership problem is supposed to be invariant*: (1) variations of the prompts that formulate the same query (2) the order in which we present the members of the set in the prompt and (3) the semantic features and relations of the members of the set. We will demonstrate that along all these dimensions *there is significant variance* in the error patterns. We also demonstrate that the different LLMs show substantial differences in their error patterns as well.

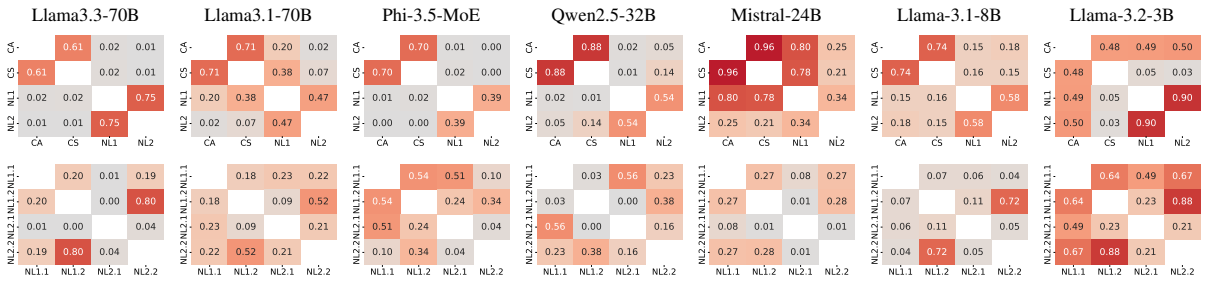

Figure 1: Cosine similarity between the average query error-rates of prompt categories for all the LLMs (top row). The bottom row shows the cosine similarity between sub-classes of NL1 and NL2, where the division is based on the *arrangement* feature (see section 3.3). NL1.1 and NL2.1 use the element-first, the others use the set-first arrangement.

4.1 Prompt sensitivity

All our queries were executed using a large number of prompt templates. These templates are differentiated by major features—natural language (NL1, NL2) or Python code (CS, CA)—as well as minor features such as the exact selection of wording and punctuation.

Prompt categories. Figure 1 illustrates the similarity between prompt categories. We computed cosine similarities between 3024 dimensional vectors that represent the set of queries we defined (see section 3.4). The vector of a prompt category contains, for each query, the average error-rate of those prompts from the given prompt category that implement the given query.

The conclusion we can draw is that the Python code prompt types CA and CS are similar, also the natural language prompts that use the same *arrangement* feature (that is, they present the set and the membership question in the same order) are somewhat similar as well, although less consistently so. Otherwise, the error pattern is not similar between different prompt categories, despite the fact the very same set membership queries are instantiated over the same LLM, and the different LLMs also show a large variability.

Prompt features. We also tested the effect of minor features such as wording and punctuation. While different explanation methods can give different results (Krishna et al., 2024), here, we opted for Shapley values (Lundberg and Lee, 2017) that describe the impact of individual features on the accuracy (fig. 2). For a given LLM, we performed the analysis over the set of all the prompts that belong to a certain prompt category. For example, in the NL1 category, this means $3024 \cdot 480$ prompts (see section 3.4).

Note that the most important feature is different in every LLM shown, and in general, the importance of the same feature can vary radically across LLMs. Still, some features have a consistent effect on the success of the models, despite being designed to be indifferent to the formulation of the query. One such feature is the *arrangement* feature (set-first or element-first prompt). Interestingly, *different LLMs might strongly prefer opposite arrangements*.

4.2 Permutation sensitivity

Another interesting question is whether the order in which the set elements are listed has an effect on the answer. Clearly, the concept of the set suggests that the element ordering should not matter. Yet, we see many counterexamples, like the one below, where Llama3.3-70B answered the first question correctly, but the second one incorrectly:

```
Does the set (Hearts, Clubs, Spades) contain the character
sequence East?
```

```
Does the set (Hearts, Spades, Clubs) contain the character
sequence East?
```

Figure 3 offers a more detailed insight, visualizing the sensitivity to ordering. To understand the plots, recall, that we execute separate queries for each different ordering of the same set. Now, if we group those queries together that are the same except the ordering of the set (thereby defining order-independent queries), we can compute the fraction of such order-independent queries that receive a consistent answer (i.e., the same answer for every permutation of the set elements). The figure shows statistics of the consistency of order-independent queries using a bar plot as a function of set-type and prompt-type.

Inconsistency seems to be relatively low at first sight (a few percent of the order-independent

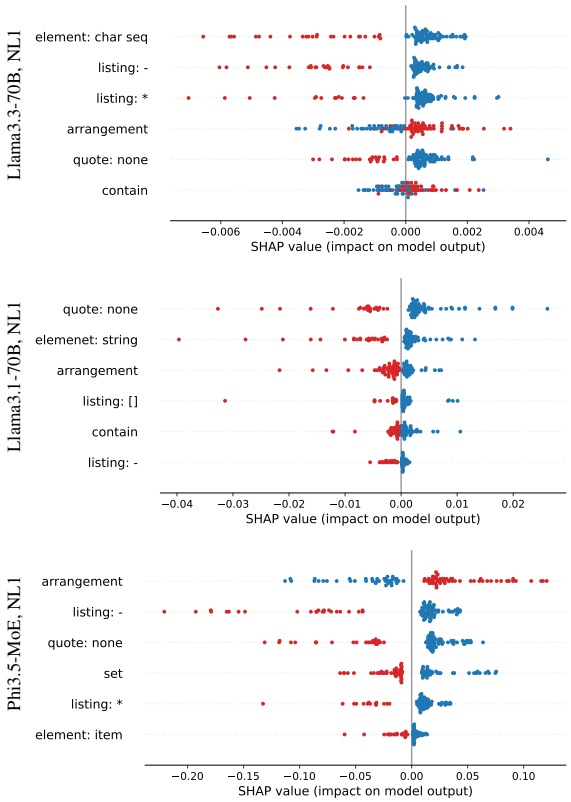

Figure 2: Top 6 Shapley values among binary features of the NL1 prompt category. The features are defined in section 3.3, non-binary features are converted to one-hot representation. A dot corresponds to a prompt template and the color red indicates that the feature is present.

queries), but that is because accuracy is high in general. In fact, inconsistency is much higher than the overall error rate (see table 1), indicating that many sets have "adversarial" orderings, in which the query fails. Interestingly, this is true even for the Python code prompts, to some extent, which is perhaps more surprising than the natural language case, given the formal nature of the domain. Note also the diversity among LLMs.

4.3 Semantic sensitivity

The semantic features of set elements should also be ignored by the design of the set membership task. Yet, we observe a strong semantic interference.

Semantic leakage. Table 1 presents the false positive rate (FPR) percentages for each investigated model, aggregated over every prompt type, for negative-member and negative-intruder query types. Apart from Llama-3.1-70B, FPRs are substantially larger for member words than for intruder words. In other words, when the correct answer is "not a member", most LLMs are more likely to say "member" when the word tested for membership is semantically related to the set elements. This is a clear case of semantic leakage.

Semantic boosting. Figure 5 illustrates a phenomenon that can be considered the opposite to semantic leakage, namely when semantic relatedness increases accuracy, as opposed to decreasing it. The scatter plots visualize the comparison of fixed prompt templates when the same template is applied to sets with and without semantically related elements. When points are under the diagonal, the errors tend to be smaller for semantically related sets. In the case of the NL1 prompt family, this is the case in every LLM.

However, for number sets with the CS prompt family, we can observe an almost opposite trend, where sets of related numbers seem to generate more errors. It is interesting, though, that the 70B Llama models are an exception, where the bias still favors the related sets.

Semantic preference. Figure 4 shows how the accuracy of different LLMs depends on the semantic type of the set in the query, for the case of the NL1 and CS prompt categories. The dependence on set-type is the clearest in the case of CS prompts, where number sets perform worse than word sets (but note that some LLMs do not show this bias). In the case of the NL1 prompts, there is a slight bias against sets of unrelated words.

4.4 LLM Sensitivity

Initially, we expected to see similar error patterns over the different LLMs, but our results contradict this hypothesis. Consider, for example, fig. 6 that illustrates the success of the different prompt templates over our set of 7 LLMs. If the LLMs had very similar error patterns then we would see most of the prompt templates in either column 0 or 7, but this is not the case. For our four classes of prompts we observe slightly different distributions, but in each case, there are only a few templates that are perfect—result in a correct answer in every single query over all the LLMs—but there are many templates that are specific to one or two LLMs, especially in the case of NL1 prompts. This is despite the fact that the overall accuracy is rather high for each LLM. In the NL1 class, there is in fact no perfect template and *the highest number of templates are perfect only on a single LLM*.

Let us illustrate some prompt templates, instantiated with the query instance [(North, South, West, East), North]. Examples for perfect templates in

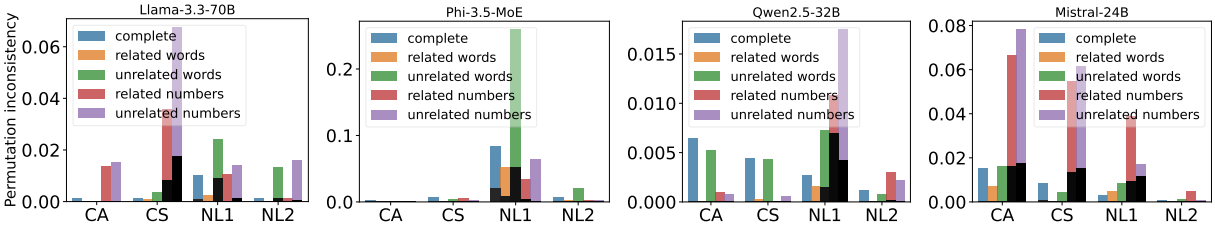

Figure 3: The fraction of order-independent queries where the answer of the LLM is not consistently correct for all the permutations of the set, by prompt type and set type. Values higher than 0.0 indicate the presence of sensitivity to ordering. The black region of a bar corresponds to the fraction of consistently incorrect order-independent queries.

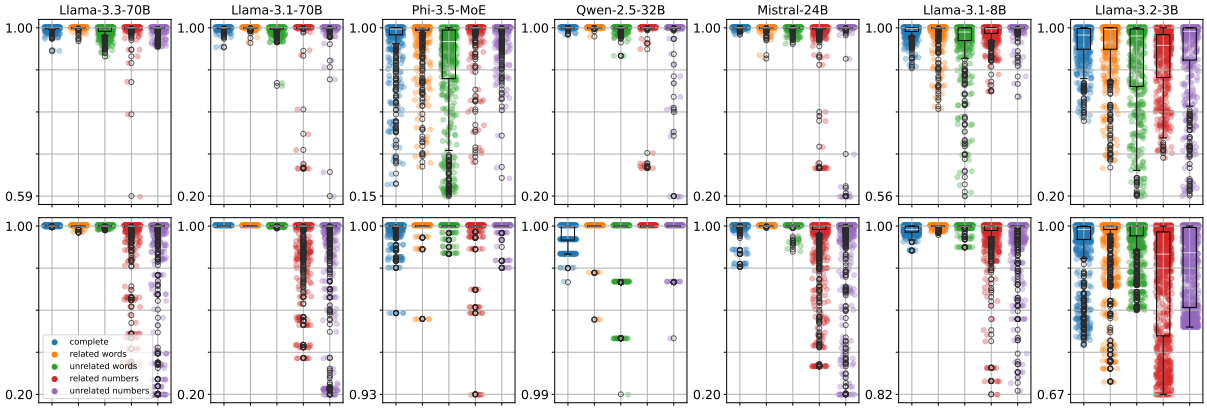

Figure 4: The average accuracy of queries with different set-types, for all the LLMs, in the NL1 (top) and CS (bottom) prompt category. Every point belongs to a fixed prompt template and represents the average of the queries that belong to the given set type. For example, for the complete set type, every point is the average accuracy of $10 \cdot (24 \cdot 4 + 6 \cdot 4 + 6 \cdot 4)$ prompts.

NL2 and CS include

```
A set consists of the following character sequences: {'North',
'South', 'West', 'East'}
Does this set include the character sequence 'North'?
```

```
Given the Python code below, what will be the value of the
‘result‘ variable?

“‘python
def contains(s, item):
    return item in s

my_set = set(('North', 'South', 'West', 'East'))
result = contains(my_set, 'North')
“‘
```

Examples for CS and NL1 templates that are never perfect include

```
Given the Python code below, what will be the output?

“‘python
def contains(s, item):
    return item in s

S = set("North South West East".split())
print(contains(S, "North"))
“‘
```

```
Does the set [North, South, West, East] include the string
North?
```

To investigate LLM similarity further, we computed the normalized pointwise mutual information between the error patterns of the LLMs, as shown in fig. 7. In more detail, for a pair of LLMs a and b we define the random variables x_a and x_b as indicator variables of giving an incorrect answer for the same random prompt by LLM a and b, respectively. We assume a uniform distribution over the possible 8,709,120 prompts. We then compute the normalized pointwise mutual information for the point $(x_a = 1, x_b = 1)$, i.e., the case when both LLMs are incorrect, according to

$$\text{npmi}(1,1) = \left[\log \frac{p_{a,b}(1,1)}{p_a(1)p_b(1)}\right] / -\log p_{a,b}(1,1).$$

The maximal value is 1, and 0 indicates independence.

Overall, in case of many pairs, the similarity is above the baseline level of 0, yet fairly low, except between the two 70B Llama models. It is interesting that the Qwen model is somewhat similar to three models from different families that are not necessarily similar to each other.

Finally, the diversity of LLM behavior is also well illustrated by some of the previously discussed results, when focusing on the differences between LLMs. The markedly different LLM behaviors include the different (sometimes opposite) effects of

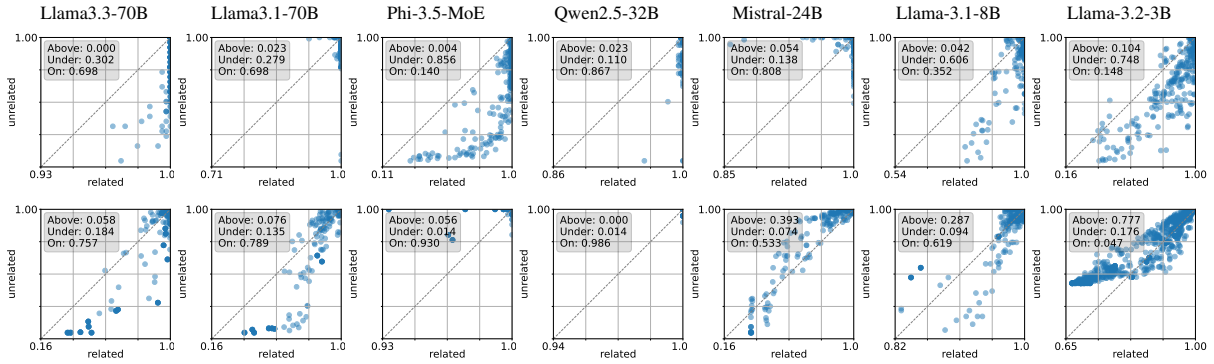

Figure 5: Scatter plots illustrating the relationship of unrelated and related word sets in the NL1 prompt category (top) and number sets in the CS prompt category (bottom). A point belongs to the same prompt template, the two coordinates are average accuracies over queries with related and unrelated sets of the given type.

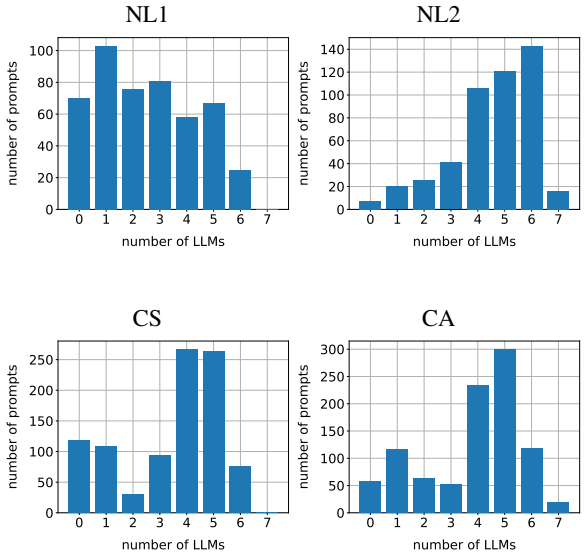

Figure 6: Column i shows the number of prompt templates that were perfect (resulted in a correct answer for every query) on exactly i LLMs.

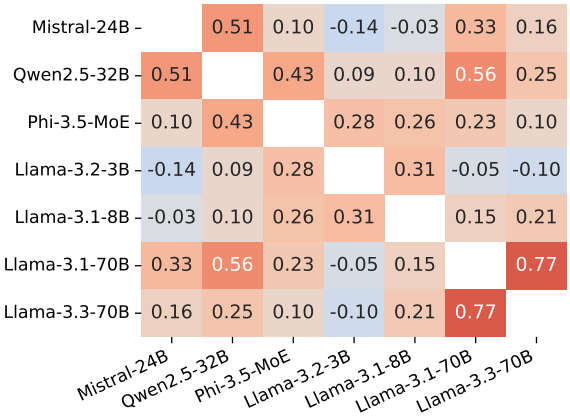

Figure 7: Normalized pointwise mutual information between the error patterns of the LLMs.

both minor (fig. 2) and major (figs. 1 and 4) prompt features, the different (in some cases, opposite) effect of semantic relatedness of the set elements (fig. 5), especially in the case of number sets, and the different sensitivity to the ordering of the set elements (fig. 3).

5 Conclusion

In this paper, we formulated the simplest possible problem we could think of: set membership on very small, explicitly given sets. Our goal was to examine the brittleness of instruction-tuned LLMs over this very basic task.

Our work uncovers a rather comprehensive set of brittleness issues. Most importantly, we found a strong sensitivity to three aspects the LLMs should be robust to in the context of set membership: prompt features, element ordering, and semantic relatedness. We also found that different LLMs often behave very differently. These results provide multi-faceted evidence for the *lack of a crisp and robust set concept in LLMs*.

We also believe that *our experimental design and analysis could be used as a benchmark*, beside the usual standard benchmarks based only on a few performance metrics that do not offer a lot of insight into potential design flaws. In a more general sense, we propose a valuable methodology for LLM evaluation that can be applied using other extremely simple problems as well.

Limitations

A limitation of our work is that—due to attempting to provide a comprehensive map of issues—we have not explored in depth the several directions we uncovered that look interesting and promising for

understanding the set concept of LLMs better. This would involve looking "under the hood" using, e.g., mechanistic interpretability techniques (Sharkey et al., 2025), or even human experiments to understand whether certain brittleness properties (e.g., semantic leakage or boosting) are specific to LLMs. The approach of (Yiu et al., 2025) using the performance of children on benchmarks is a possible starting point.

Acknowledgements

Support from Project 2024-1.2.3-HU-RIZONT-2024-00017 is acknowledged, financed by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, under the a 2024-1.2.3-HU-RIZONT funding scheme. Support from grants NSF 2217023 and NSF 2240532 is acknowledged.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, and 1 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

G. W. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. 2020. Learning invariances in neural networks from training data. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*.

Christine Cuskley, Rebecca Woods, and Molly Flaherty. 2024. The limitations of large language models for understanding human language and cognition. *Open Mind*, 8:1058–1083.

I. Dasgupta, A. K. Lampinen, S. C. Y. Chan, A. Creswell, D. Kumaran, J. L. McClelland, and F. Hill. 2022. Language models show human-like content effects on reasoning. *CoRR*, abs/2207.07051.

S. A. Dewhurst, S. J. Holmes, K. R. Brandt, and G. M. Dean. 2006. Measuring the speed of the conscious components of recognition memory: Remembering is faster than knowing. *Consciousness and Cognition*, 15:147–162.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

B. Egressy and J. Stühmer. 2025. Set-llm: A permutation-invariant llm. *Preprint*, arXiv:2505.15433.

Y. Elazar, N. Kassner, S. Ravfogel, A. Ravichander, E. Hovy, H. Schütze, and Y. Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

C. Fierro and A. Søgaard. 2022. Factual consistency of multilingual pretrained language models. pages 3046–3052, Dublin, Ireland. Association for Computational Linguistics.

Hila Gonen, Terra Blevins, Alisa Liu, Luke Zettlemoyer, and Noah A. Smith. 2025. Does liking yellow imply driving a school bus? semantic leakage in language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 785–798, Albuquerque, New Mexico. Association for Computational Linguistics.

A. György, T. Lattimore, N. Lazic, and Cs. Szepesvári. 2025. Beyond statistical learning: Exact learning is essential for general intelligence. *CoRR*, abs/2506.23908.

J. Huckle and S. Williams. 2025. Easy problems that llms get wrong. In *Future of Information and Communication Conference (FICC 2025)*, pages 313–332.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

P. N. Johnson-Laird. 1983. *Mental Models. Towards a Cognitive Science of Language, Inference and Consciousness*. Cambridge Univ. Press.

S. Khemlani, M. Lotstein, and P. Johnson-Laird. 2014. A mental model theory of set membership. In *Proceedings of the 36th Annual Meeting of the Cognitive Science Society, CogSci 2014*.

S. Krishna, T. Han, A. Gu, S. Wu, S. Jabbari, and H. Lakkaraju. 2024. The disagreement problem in explainable machine learning: A practitioner's perspective. *Trans. Mach. Learn. Res.*

A K Lampinen, I Dasgupta, S C Y Chan, and 1 others. 2024. Language models, like humans, show content effects on reasoning tasks. *PNAS Nexus*, 3:233.

M. Lewis and M. Mitchell. 2025. Evaluating the robustness of analogical reasoning in large language models. *Trans. Mach. Learn. Res.*

S. M. Lundberg and S.-I. Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 4765–4774.

R. T. McCoy, S. Yao, D. Friedman, M. D. Hardy, and T. L. Griffiths. 2024. Embers of autoregression show how large language models are shaped by the problem they are trained to solve. *Proceedings of the National Academy of Sciences*, 121(41):e2322420121.

M. Mitchell. 2021. Why AI is harder than we think. *CoRR*, abs/2104.12871.

A. Passerini, A. Gema, P. Minervini, B. Sayin, and K. Tentori. 2024. Fostering effective hybrid human-llm reasoning and decision making. *Frontiers Artif. Intell.*, 7.

H. Raj, D. Rosati, and S. Majumdar. 2023. Measuring reliability of large language models through semantic consistency.

S. Ravanbakhsh, J. G. Schneider, and B. Póczos. 2017. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017*, volume 70, pages 2892–2901. PMLR.

A. Razmjooei. 2013. Investigation of some cognitive difficulties in set theory. Technical report, University of Stockholm.

P. Sahu, M. Cogswell, Y. Gong, and A. Divakaran. 2022. Unpacking large language models with conceptual consistency. *Preprint*, arXiv:2209.15093.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2024. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Mary Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, and 10 others. 2025. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*. Survey Certification.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. 2025. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *Preprint*, arXiv:2506.06941.

Qwen Team. 2024. Qwen2.5: A party of foundation models. Alibaba Cloud.

P. E. Wais, L. R. Squire, and J. T. Wixted. 2010. In search of recollection and familiarity signals in the hippocampus. *J. Cogn. Neurosci.*, 22:109–123.

Peter C Wason. 1968. Reasoning about a rule. *Quarterly journal of experimental psychology*, 20(3):273–281.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Christopher Wolfram and Aaron Schein. 2025. World models and consistent mistakes in llms. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML '25. PMLR.

E. Yiu, M. Qraitem, A. Noor Majhi, C. Wong, Y. Bai, S. Ginosar, A. Gopnik, and K. Saenko. 2025. Kiva: Kid-inspired visual analogies for testing large multimodal models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*.

A. P. Yonelinas, M. M. Ramey, and C. Riddell. 2022. *Recognition memory: The role of recollection and familiarity*. Oxford Handbook of Human Memory, Oxford Univ. Press.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 3391–3401.

D. Zheng, M. Lapata, and J. Z. Pan. 2024. How reliable are llms as knowledge bases? re-thinking factuality and consistency. *Preprint*, arXiv:2407.13578.

L. Zhou, W. Schellaert, F. Martínez-Plumed, Y. Moros-Daval, C. Ferri, and J. Hernández-Orallo. 2024. Larger and more instructable language models become less reliable. *Nat.*, 634(8032):61–68.

J. Zhuo, S. Zhang, X. Fang, H. Duan, D. Lin, and K. Chen. 2024. ProSA: Assessing and understanding the prompt sensitivity of llms. In *Findings of the ACL: EMNLP 2024*, pages 1950–1976.

Appendix

In this Appendix, we present our experimental setup in full detail to allow for reproducibility. We also present all the plots that were not included in the main paper.

A Setup

We performed our empirical evaluations with Huggingface's `transformers` library (Wolf et al., 2020). For every query, we used the greedy search generation strategy, i.e. during generation the most likely token was selected.

Set Type	Intruder Element	Semantic Connection
Complete Sets		
North, West, East, South	incisors	cardinal directions
NE, NW, SE, SW	plasma	ordinal directions
freshman, sophomore, junior, senior	Thymine	years in high school
Gryffindor, Hufflepuff, Ravenclaw, Slytherin	Spades	houses of Hogwarts
earth, fire, air, water	molars	classical elements
solid, liquid, gas, plasma	freshman	fundamental states of matter
incisors, canines, premolars, molars	multiplication	types of human teeth
addition, subtraction, multiplication, division	Hufflepuff	basic arithmetic operations
Hearts, Diamonds, Clubs, Spades	East	French card suits
Adenine, Thymine, Cytosine, Guanine	canines	bases of DNA
Related Word Sets		
cat, horse, deer, bear	amethyst	animals
cherry, pear, melon, banana	attack	fruits
shirt, dress, skirt, sweater	candy	clothes
Unrelated Word Sets		
candy, amethyst, belt, pond	-	-
gas, bee, cheese, actor	-	-
attack, daisy, attic, commerce	-	-
Related Number Sets		
100, 102, 104, 106	257	arithmetic sequence with d=2
100, 175, 250, 325	276	arithmetic sequence with d=75
100, 200, 300, 400	399	arithmetic sequence with d=100
Unrelated Number Sets		
399, 690, 734, 847	-	-
276, 508, 661, 863	-	-
257, 356, 650, 935	-	-

Table 2: The 22 sets used in our experiments.

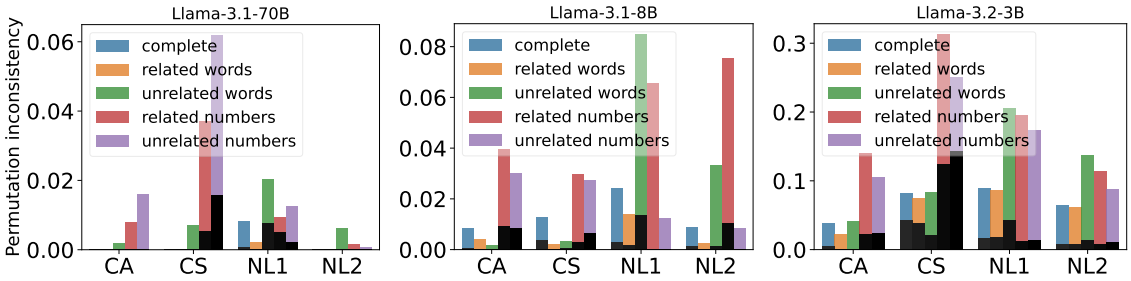

Figure 8: The fraction of order-independent queries where the answer of the LLM is not consistently correct for all the permutations of the set, by prompt type and set type for LLMs not included in the main article. Values higher than 0.0 indicate the presence of sensitivity to ordering. The black region of a bar corresponds to the fraction of consistently incorrect order-independent queries.

We used Nvidia H100 GPUs in our experiments. The evaluation took around 140 GPU-days to complete on all models, prompt schemas and sets with all permutations (60 963 840 queries all together).

The HuggingFace IDs of the LLMs we used are the following:

- meta-llama/Llama-3.2-3B-Instruct
- meta-llama/Llama-3.1-8B-Instruct
- mistralai/Mistral-Small-24B-Instruct-2501
- Qwen/Qwen2.5-32B-Instruct
- microsoft/Phi-3.5-MoE-instruct
- meta-llama/Llama-3.1-70B-Instruct
- meta-llama/Llama-3.3-70B-Instruct

B List of sets

We list the 22 sets used in our experiments in table 2. The construction of the sets was done based on human judgement, that is, based on the judgement of the authors. This involved consulting with LLMs as well, but not in a structured manner.

The **complete sets of words** are an exception, where we first came up with a pool of candidate quadruples. We then tested the candidate sets for completion: we removed one element and asked the LLMs (the ones used in the evaluation in the main paper) what the missing element is. We regarded a set to be complete if all the LLMs proposed the removed element as the missing one for every query. Queries tested removing all the elements of a given quadruple and presenting the remaining three elements in every possible order.

An example prompt to test what element the LLM considers to be missing from the set would be of the form

```
Answer with the most likely missing element from the below
set:
North, East, South
```

In this case, we require the response of all the investigated LLMs to be `West`. A single incorrect answer discarded the set from the candidate quadruples.

C Python prompt schemes in CS and CA

The features defined for the CS and CA schemas are the following:

- *init*: initialization of the set
  - shorthand
  - constructor with list
  - constructor with tuple
  - constructor with string and split

Examples for the different init features:

```
S = "North", "South", "West", "East"
S = set(["North", "South", "West", "East"])
S = set(("North", "South", "West", "East")
S = set("North South West East".split())
```

- *variable*: `S` | `my_set` (variable name for the set)
- *quotation*: `'` | `"` (quotation mark for strings)
- *tab*: how the code was tabulated
  - 2 spaces
  - 4 spaces
  - tab
- *result*: representation of the result
  - stored in a variable
  - printed out

Example CS prompts for result features. Note that the given question changes accordingly.

````
Answer with a single word. Given the Python code below, what
will be the value of the ‘result‘ variable?

“‘python
S = "North", "South", "West", "East"
result = "North" in S
“‘
````

````
Answer with a single word. Given the Python code below, what
will be the output?

“‘python
S = "North", "South", "West", "East"
print("North" in S)
“‘
````

- *logic*: structure of the code
  - *main*: all operations are executed in the global scope
  - *function*: the set containment operation is defined in a function
  - *function with typehints*: the function has typehints in the function header
  - *function with docstring*: the function has a docstring
  - *function with typehints and docstring*
- *argument order*: order of arguments if a function is present
  - set-first
  - element-first

Example CS prompts for logic features, i.e. everything is executed in the main scope or a function was defined with both typehinting and docstring.

````
Answer with a single word. Given the Python code below, what
will be the value of the ‘result‘ variable?

“‘python
S = "North", "South", "West", "East"
result = "North" in S
“‘
````

````
Answer with a single word. Given the Python code below, what
will be the value of the ‘result‘ variable?

“‘python
from typing import Any, Set

def contains(s: Set[Any], item: Any) -> bool:
    """
    Checks if an item is in a set.

    Args:
        s (set): The set to check in.
        item (any): The item to check for.
    Returns:
        bool:  True  if  the  item  is  in  the  set,  False
otherwise.
    """
    return item in s

S = "North", "South", "West", "East"
result = contains(S, "North")
“‘
````

D Additional plots

Figure 8 shows the permutation inconsistency of the remaining LLMs (those that were left out from the main paper). Here, again, we see that the LLMs show a significant inconsistency to permutation, as well as among each other. Llama-3.2-3B and 3.1-70B, for example, are surprisingly sensitive to the order of number-sets in the CS prompt type, while the 8B version is not.

Figure 9 shows the complete set of results regarding the dependence of accuracy on set-types, and Figures 10 to 12 show the complete set of scatter plots illustrating the connection between semantic relatedness and unrelatedness for reference. The plots give further support to the claims in the paper but also allow one to examine the relationship between complete and related word sets, or between CS and CA prompts.

Figures 13 to 16 include the complete results of the **Shapley analysis**, giving further support to the claims made in the main paper.

For example, in the case of the Python prompts, the initialization using the split method is an important feature with a negative effect, except for Llama 3B, where it is not that important.

The feature of not using a helper function (logic: main) mostly results in a strong negative effect, except some LLMs, like Qwen, where this feature has a slight positive effect.

The most important features are also quite variable. In general, the Python prompts show very similar brittleness properties to those of the natural language prompts.

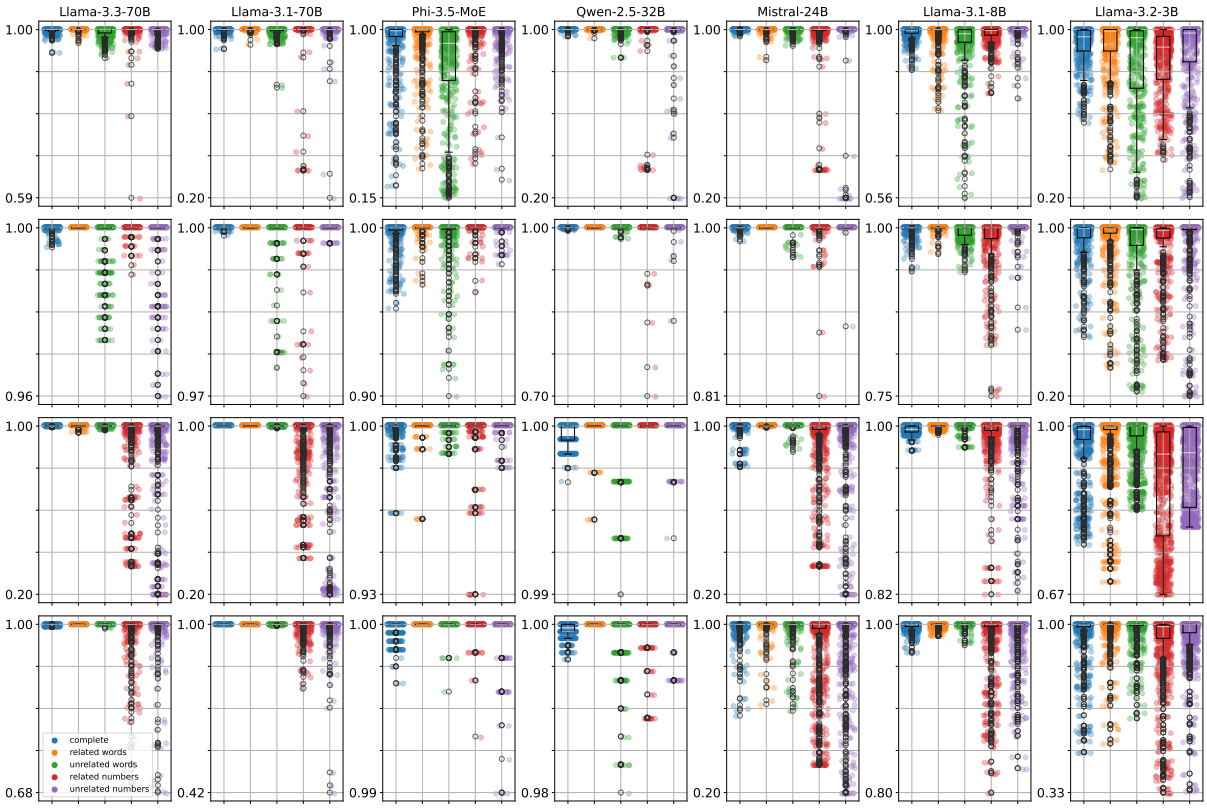

Figure 9: The average accuracy of queries with different set-types, for all the LLMs, in the NL1, NL2, CS and CA prompt categories (rows from top to bottom). Every point belongs to a fixed prompt template and represents the average of the queries that belong to the given set type. For example, for the complete set type, every point is the average accuracy of $10 \cdot (24 \cdot 4 + 6 \cdot 4 + 6 \cdot 4)$ prompts.

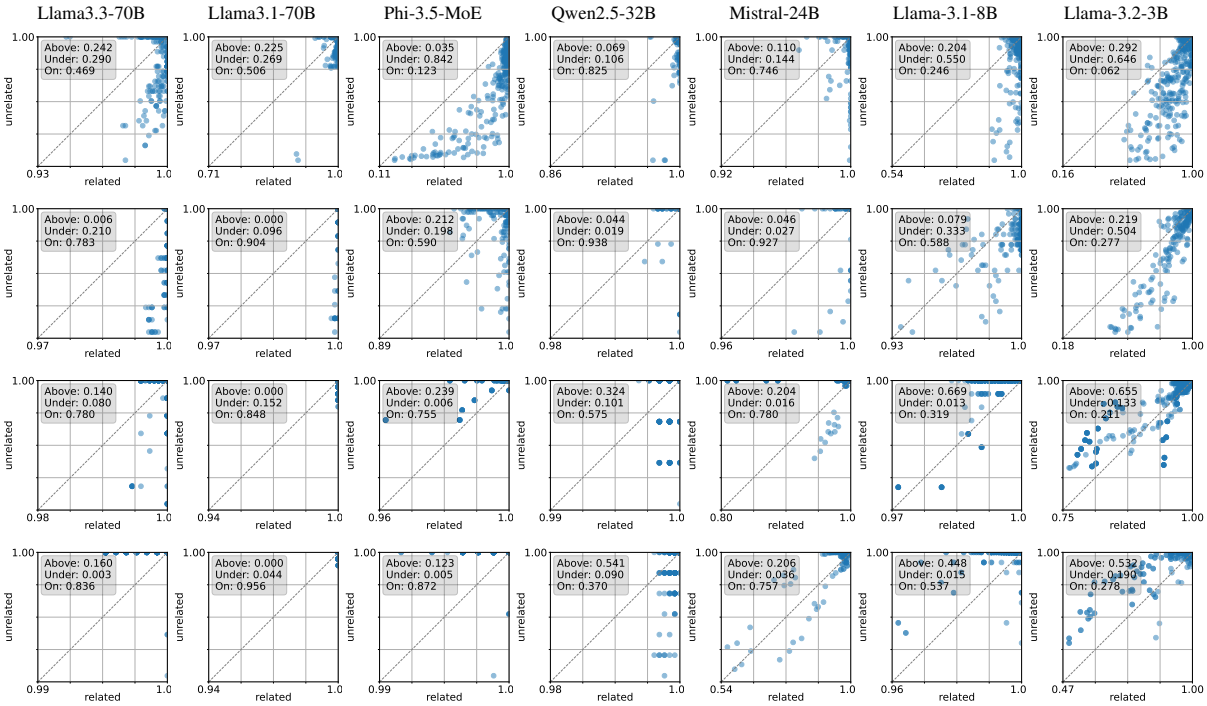

Figure 10: Scatter plots illustrating the relationship of **unrelated word sets and complete sets** in the NL1, NL2, CS and CA prompt categories (rows from top to bottom). A point belongs to the same prompt template, the two coordinates are average accuracies over queries with related and unrelated sets of the given type.

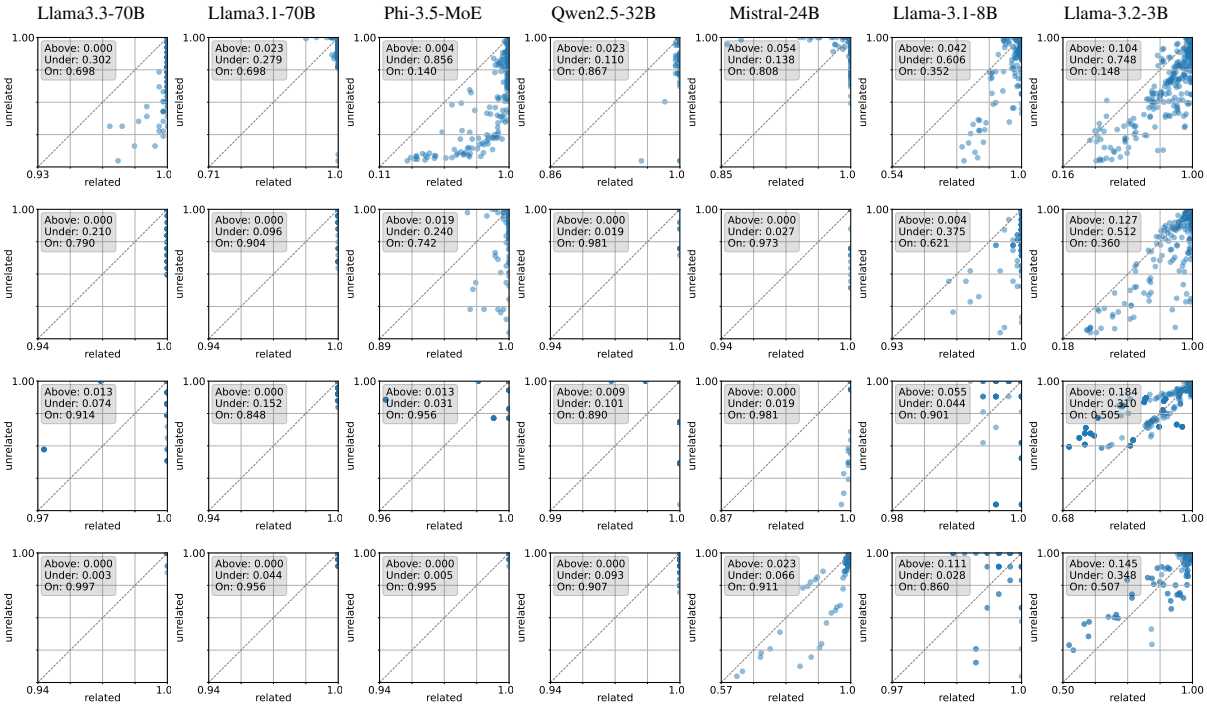

Figure 11: Scatter plots illustrating the relationship of **unrelated and related word sets** in the NL1, NL2, CS and CA prompt categories (rows from top to bottom). A point belongs to the same prompt template, the two coordinates are average accuracies over queries with related and unrelated sets of the given type.

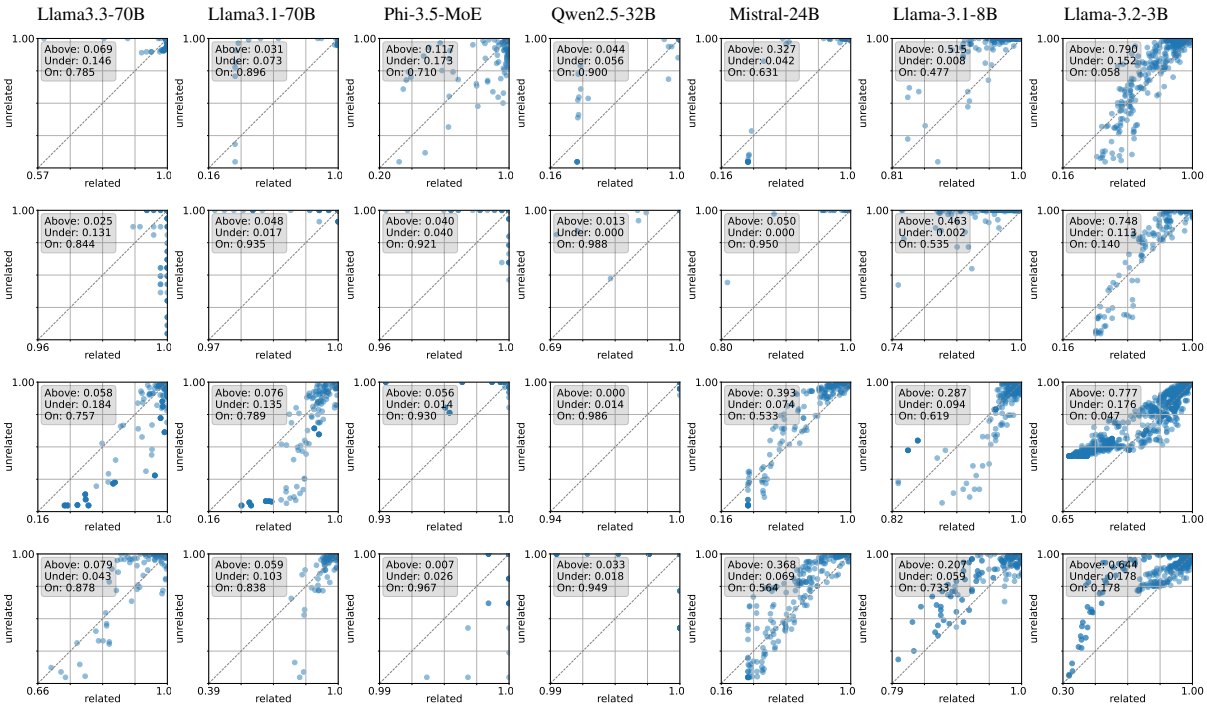

Figure 12: Scatter plots illustrating the relationship of **unrelated and related numbers sets** in the NL1, NL2, CS and CA prompt categories. A point belongs to the same prompt template, the two coordinates are average accuracies over queries with related and unrelated sets of the given type.

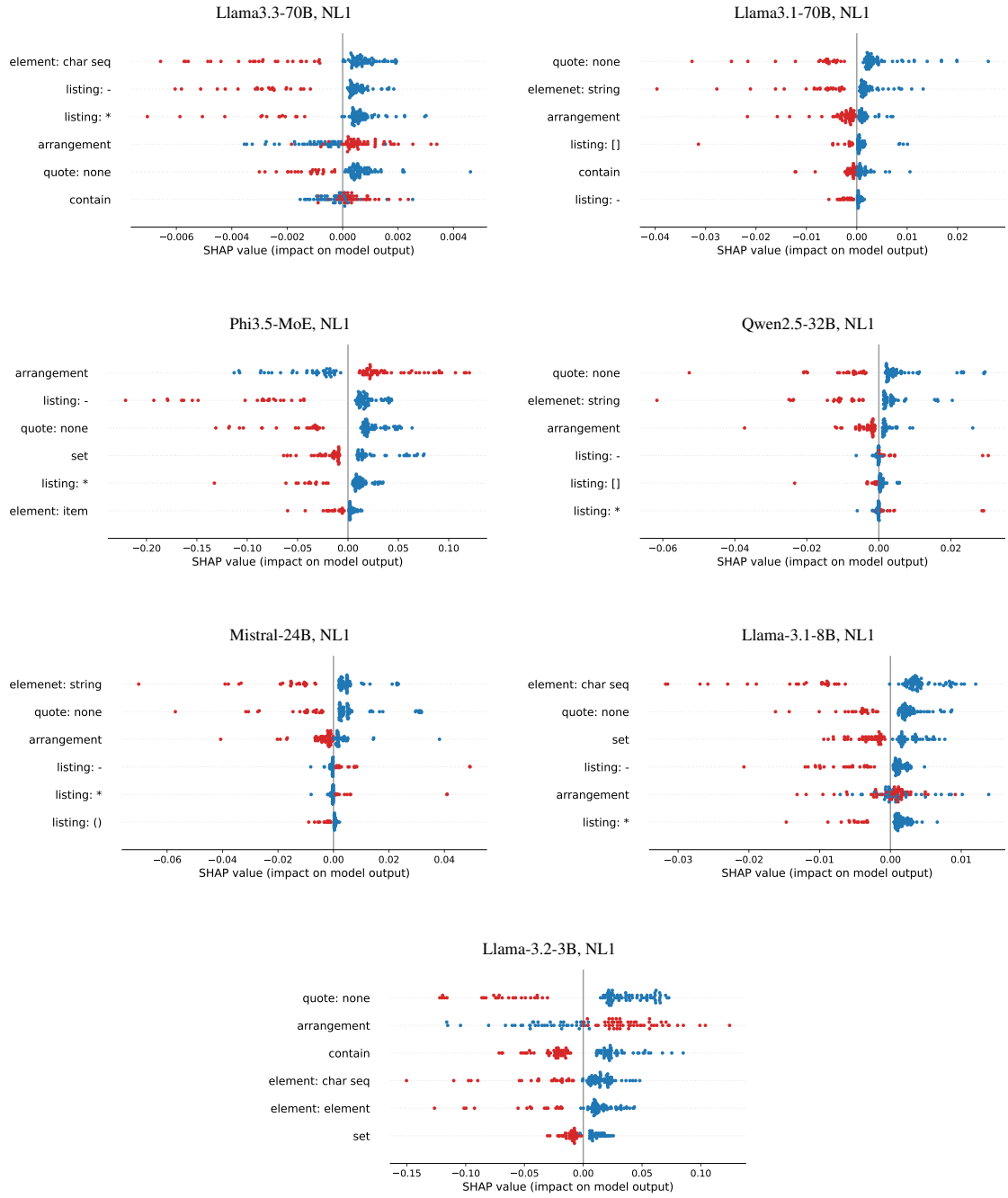

Figure 13: Top 6 Shapley values among binary features of the **NL1 prompt category**. Non-binary features are converted to one-hot representation. A dot corresponds to a prompt template and the color red indicates that the feature is present.

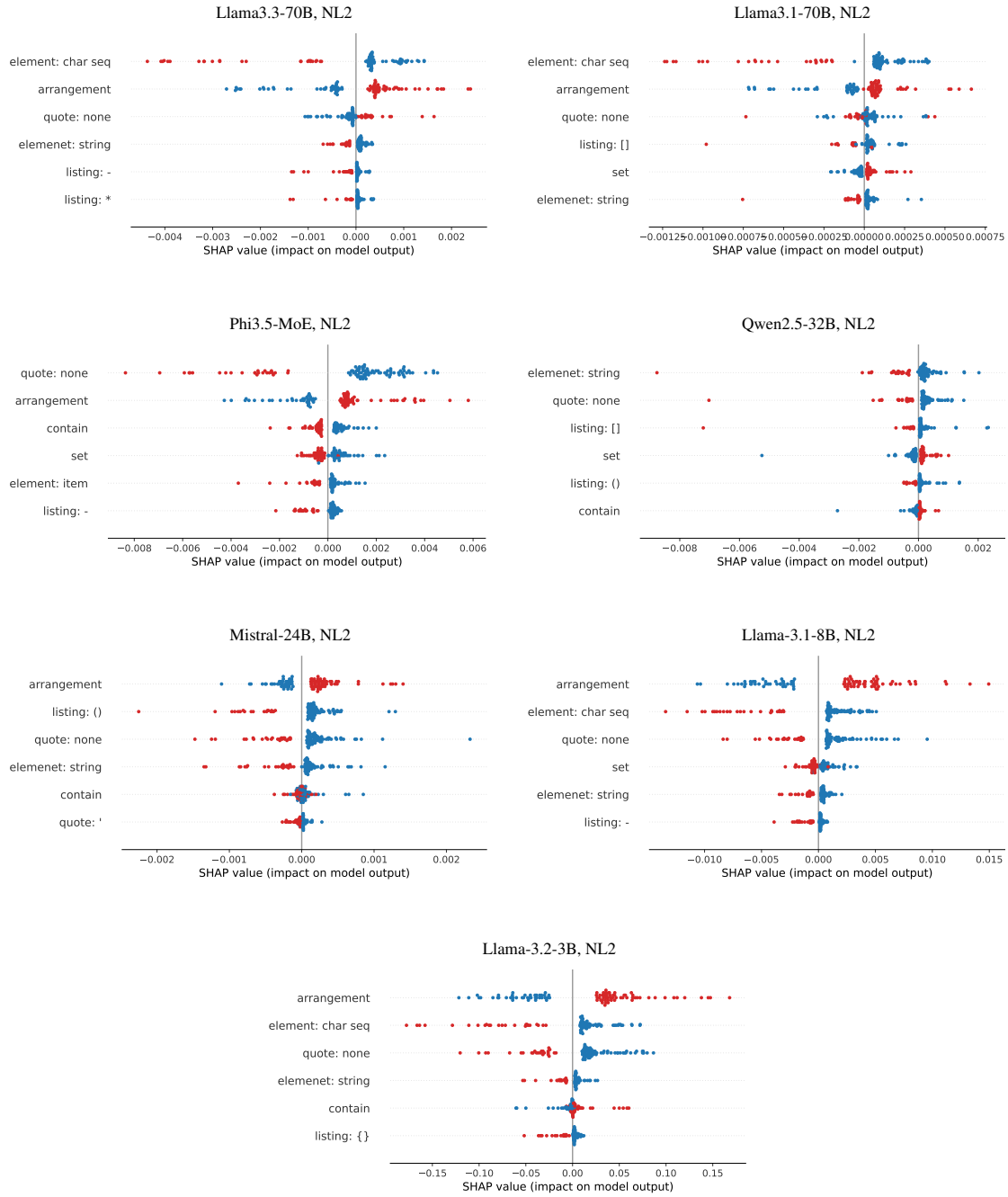

Figure 14: Top 6 Shapley values among binary features of the **NL2 prompt category**. Non-binary features are converted to one-hot representation. A dot corresponds to a prompt template and the color red indicates that the feature is present.

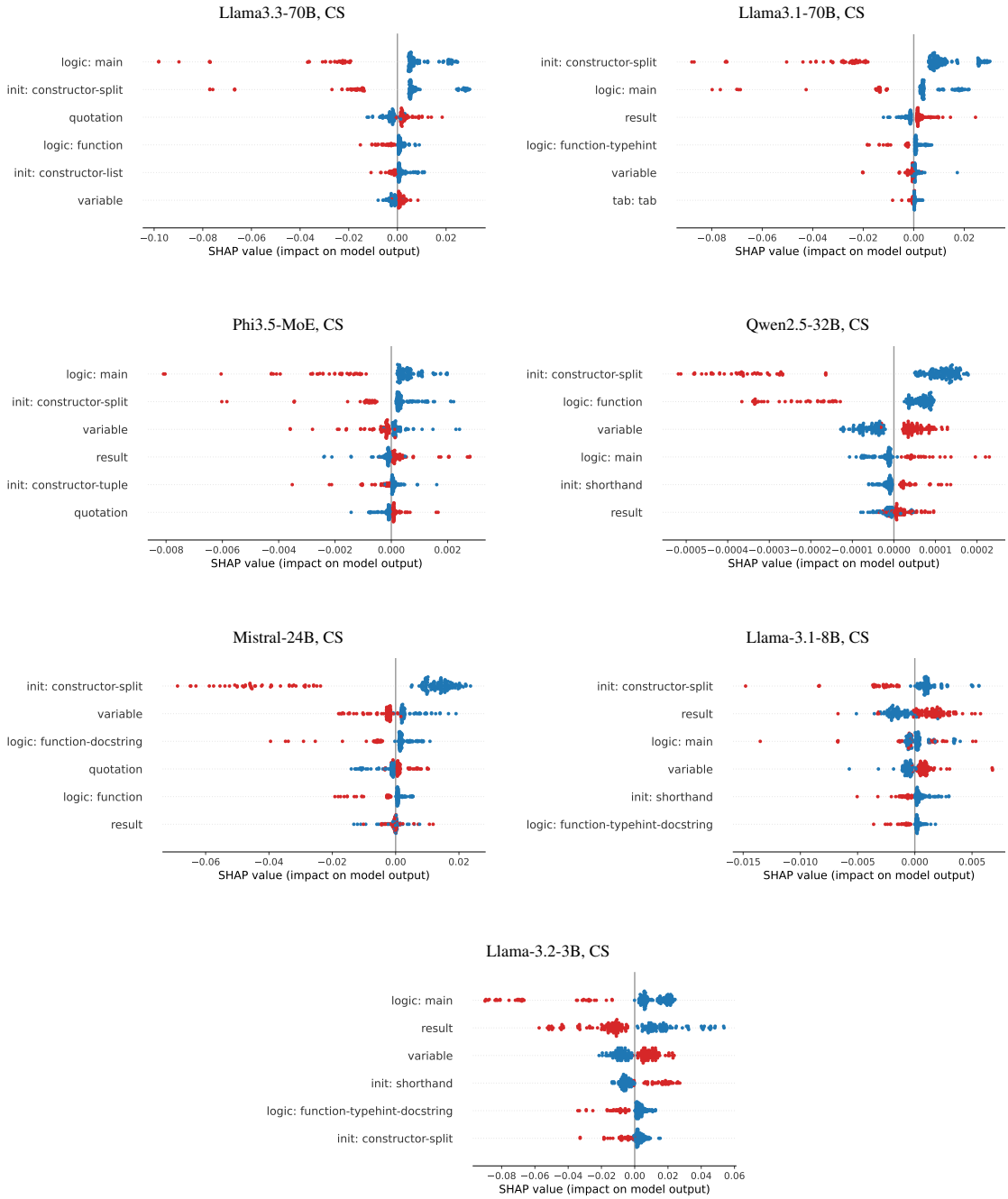

Figure 15: Top 6 Shapley values among binary features of the **CS prompt category**. Non-binary features are converted to one-hot representation. A dot corresponds to a prompt template and the color red indicates that the feature is present.

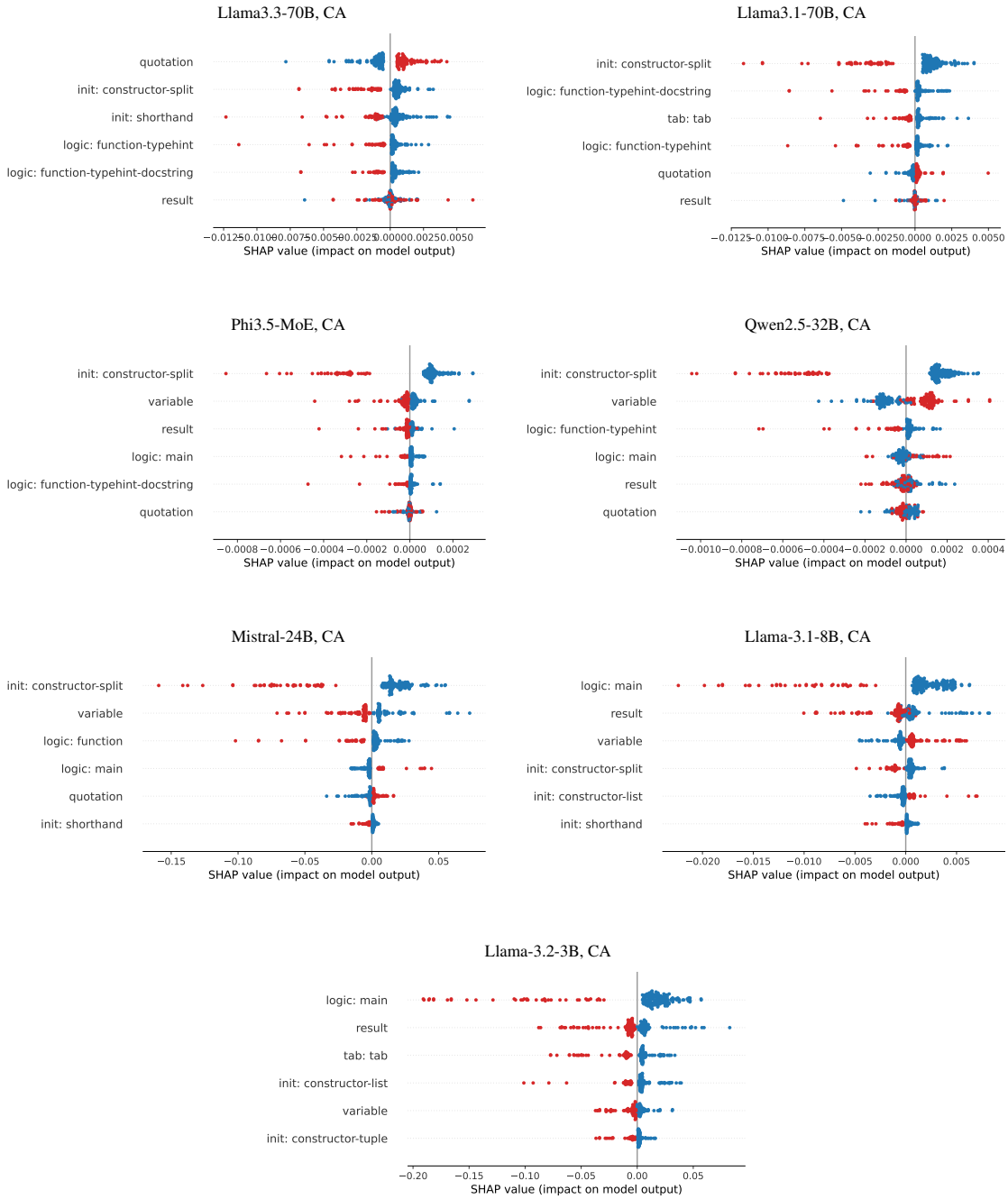

Figure 16: Top 6 Shapley values among binary features of the **CA prompt category**. Non-binary features are converted to one-hot representation. A dot corresponds to a prompt template and the color red indicates that the feature is present.