Understanding LLM Performance Degradation in Multi-Instance Processing: The Roles of Instance Count and Context Length

Anonymous ACL submission

Abstract

Users often rely on Large Language Models (LLMs) for processing documents or performing analysis over a number of instances, such as sentences. For instance, analysing the average sentiment of movie reviews requires an LLM to process the sentiment of each review individually in order to provide a final aggregated answer. While LLM performance on such individual tasks is beyond doubt, there has been little research on how LLMs perform when dealing with multi-instance inputs. In this paper, we perform an exhaustive evaluation of the ability of LLMs to handle multi-instance inputs for tasks in which they excel individually. The results show that most LLMs follow a pattern of slight performance degradation for small numbers of instances (≈20–100), followed by a performance collapse beyond larger instance counts. Crucially, our analysis shows that while context length is partially responsible for this degradation, the number of instances has a stronger effect on the final results. This finding suggests that when optimising LLM performance for multi-instance processing, attention should be paid to both context length and, in particular, instance count.

1 Introduction

LLMs have demonstrated remarkable capabilities across a wide range of NLP tasks and beyond (Wang et al., 2024). However, these capabilities have been predominantly evaluated in settings where a single instance is provided to the model at a time, which we refer to as single-instance processing (SIP). In contrast, many real-world applications, such as data analytics, document analysis, and large-scale information processing, require multi-instance processing (MIP), where a model must reason over and aggregate information from multiple inputs simultaneously (Chen et al., 2025; Rahman et al., 2025; Sun et al., 2025). Compared to SIP, MIP poses additional challenges due to its long-context nature and the need to perform repeated reasoning and aggregation over multiple instances, making it substantially more demanding for current LLMs (Bertsch et al., 2025). Ensuring reliable performance in such settings therefore requires a careful understanding of model failure modes, in order to inform the development of effective mitigation strategies for MIP.

As a motivating example, consider a non-expert user who inputs multiple movie reviews and wishes to obtain their overall sentiment distribution. Figure 1 illustrates a toy comparison between SIP and MIP: while an LLM can correctly classify the sentiment of each review in isolation, it may fail when required to process and aggregate all instances within a single prompt. Given that the model is capable of accurately handling individual instances, it is therefore crucial to understand the nature of the errors that arise when multiple instances must be processed jointly. Although alternative solutions, such as agentic designs that process instances separately or require users to manually batch instances or write code, are possible, these approaches are often impractical for non-expert users in real-world settings.

Existing work has extensively examined the challenges posed by long-context inputs, showing that model performance often degrades as context length increases, even when inputs remain within the model's nominal context window (An et al., 2025; Moon and Lim, 2025). However, in most long-context benchmarks, increasing input length is accompanied by a simultaneous increase in task complexity (Hsieh et al., 2024). This coupling makes it difficult to disentangle whether observed performance degradation arises from longer inputs per se or from increased semantic and reasoning demands. In parallel, multi-instance or batch processing settings have been explored in prior work, but primarily from an efficiency or cost-reduction perspective, and typically with relatively small batch

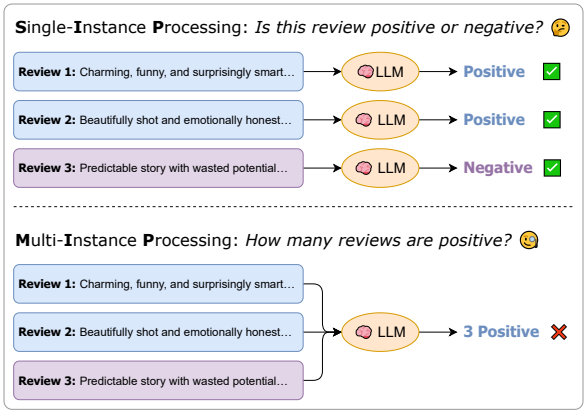

Figure 1: A toy example of SIP and MIP settings for sentiment analysis, where an LLM succeeds under SIP but fails under MIP given the same instances.

sizes (Cheng et al., 2023; Lin et al., 2024a). As a result, the effect of scaling the number of instances itself on model performance remains underexplored.

Given these gaps, we propose two research questions (RQs) to better understand LLM behaviour in MIP settings: (1) *How does LLM performance change in MIP settings, and what failure behaviours emerge?*; and (2) *What is the primary driver of performance degradation in MIP: the number of instances or the context length?*

To answer these RQs, we evaluate nine LLMs across a diverse set of eight tasks, including calculation, token-level and sentence-level classification, for which they are known to perform well when only individual instances are provided. We analyse a broad range of open-weight and closed-source LLMs, and systematically examine the limits and factors that influence their performance in MIP settings. Our results show that model performance typically degrade gradually for small instance counts before collapsing at larger scales, and that this degradation is driven more strongly by the number of instances than by context length, even when the latter is substantially increased.

2 Related Work

The practical demand for MIP is typically driven by the use of LLM-based agents for complex data analytics, such as data wrangling, exploratory analysis, and multi-file reasoning (Guo et al., 2024; Hong et al., 2025; Nam et al., 2025). In these workflows, data processing is a foundational step and often requires transforming heterogeneous inputs into intermediate representations or executable code (Lin et al., 2024b; Shankar et al., 2024). While current approaches typically adopt modular pipelines to improve reliability through task decomposition (Nam et al., 2025; Shankar et al., 2024), these systems are ultimately bounded by the LLM's ability to handle high-density information. Understanding the limits of such analytical workflows therefore requires examining LLM performance along two critical dimensions: the capacity to maintain coherence over long contexts and the ability to execute multiple concurrent tasks through batch processing.

Long Context. Recent benchmarks have examined LLM performance degradation in long-context settings, spanning retrieval-based evaluations (Hsieh et al., 2024), long-context reasoning over dispersed evidence (Kuratov et al., 2024; Vodrahalli et al., 2024; Zhang et al., 2024), and broader application-oriented suites (Yen et al., 2025). Complementary work also studies specialised regimes, including long procedural generation (Ye et al., 2025), scalable mathematical reasoning (Zhou et al., 2025), narrative understanding (Hamilton et al., 2025), and long-term conversational memory (Wu et al., 2025). Closest to our work, Bertsch et al. (2025) evaluates long-context information aggregation via many in-context instances, but varies task difficulty primarily through context length rather than isolating the effect of increasing the number of instances.

Batch Processing. Recent works study whether LLMs can answer multiple questions within a single prompt, typically motivated by reducing inference cost and balancing capacity limits (Cheng et al., 2023; Ji et al., 2025; Lin et al., 2024a). These studies consistently observe that only a small number of instances can be processed reliably before accuracy degrades. However, they do not investigate scaling behaviour beyond this regime, which arises in many practical settings where non-expert users may directly pass all instances to an LLM. Moreover, their evaluations primarily focus on independent question answering (Wang et al., 2025) or classification tasks (Gozzi and Di Maio, 2024), rather than controlled settings in which task semantics are fixed and the instance count itself is the primary source of difficulty.

3 Multi-Instance Processing

We study MIP, where an LLM is required to reason over multiple input instances within a single prompt. In contrast to retrieval-augmented genera-

tion (RAG), which typically uses only a subset of retrieved inputs to produce a final answer, MIP requires processing all provided instances. As shown in Figure 1, LLMs must iterate over all instances individually in MIP to produce intermediate results, which are then aggregated into a final answer.

An instance is defined as a single data entry x (e.g., a movie review, a user post, a sentence, or a number). Let $\mathcal{X}$ denote the set of all available instances. Given a task-specific instruction prompt t, the model m is provided with a subset $\mathcal{X}' \subseteq \mathcal{X}$ and produces a structured textual output o. The number of instances $n = |\mathcal{X}'|$ may vary across inputs. SIP is treated as a special case of MIP where $n = 1$.

3.1 Formulation

Given a task instruction prompt t and an input instance set $\mathcal{X}' = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, the model m generates an output

$$o \sim p_\theta(\cdot \mid t, \mathcal{X}').$$

A valid model output typically contains an aggregated prediction y^{agg} together with a natural language explanation r. Let $y^{\text{agg}\star}$ denote the corresponding ground-truth aggregated label. An output is considered correct when the aggregated prediction matches the ground truth, i.e., $y^{\text{agg}} = y^{\text{agg}\star}$. Otherwise, it is considered a wrong output and belongs to the set $\mathcal{W}$. The model may also produce an invalid output, in which case it fails to generate a well-formed prediction. Such outputs belong to the set $\mathcal{I}$. We define both wrong answers (i.e., valid but incorrect outputs) and invalid outputs as failures:

$$\mathcal{E}_{\text{fail}} = \mathcal{W} \cup \mathcal{I}.$$

We write $o \in \mathcal{E}_{\text{fail}}$ to denote that a model output is a failure.

3.2 Filtering for Controlled Difficulty

An important aspect of our methodology is ensuring that the evaluated tasks are simple enough for LLMs to solve when individual instances are provided (i.e., in SIP). Therefore, to control instance-level task difficulty when evaluating MIP, we construct inputs based on SIP outcomes. Let $\mathcal{X}_{\text{SIP}} \subseteq \mathcal{X}$ denote the subset of instances for which all models produce a correct prediction under the SIP setting. MIP inputs are then formed by uniformly sampling subsets $\mathcal{X}' \subseteq \mathcal{X}_{\text{SIP}}$ using fixed random seeds. This procedure ensures that failures observed in the MIP setting are not attributable to intrinsic instance difficulty or ambiguity, but instead reflect the model's ability to reason over and aggregate multiple instances, which is the primary focus of this work.

To further ensure reliable evaluation, we retain only models whose average SIP task success rate exceeds 95% and whose per-task SIP success rate exceeds 90%. We also keep only tasks for which agreement among all retained models exceeds 85%, measured as the proportion of instances that all models answer correctly prior to filtering.

3.3 Evaluation Metrics

We define an experiment as a specific evaluation configuration represented as a tuple $e = (m, \tau, \mathcal{X}')$, where m denotes a language model, τ denotes a task with its associated instruction prompt t, and $\mathcal{X}'$ denotes the input instance set. Each experiment produces a model output o_e.

We define accuracy as a binary metric:

$$\text{Acc}(e) = \begin{cases} 1, & \text{if } y^{\text{agg}} = y^{\text{agg}\star}, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathcal{D}$ denote the set of all evaluated experiments. The success rate (SR) is defined as the average accuracy across experiments:

$$\text{SR} = \frac{1}{|\mathcal{D}|} \sum_{e \in \mathcal{D}} \text{Acc}(e).$$

The invalid rate (IR) measures the fraction of experiments in which the model produces an invalid output:

$$\text{IR} = \frac{|\{\, e \in \mathcal{D} \mid o_e \in \mathcal{I} \,\}|}{|\mathcal{D}|}.$$

Finally, we report the average context length (ACL), defined as the average number of prompt tokens across experiments with valid outputs:

$$\text{ACL} = \frac{1}{|\mathcal{D}_{\text{valid}}|} \sum_{e \in \mathcal{D}_{\text{valid}}} \text{Len}(p_e),$$

where p_e denotes the prompt used in experiment e, and $\mathcal{D}_{\text{valid}} = \{e \in \mathcal{D} \mid o_e \notin \mathcal{I}\}$.

4 Experimental Setting

In this section, we describe our general experimental setting. In particular, we introduce the chosen tasks (Section 4.1), the comparison models and prompting setup (Section 4.2), and the instance filtering process (Section 4.3).

Name	Task
Arithmetic	Solve arithmetic problems & Sum of answers
Category	Classify news category & Aggregate class counts
Language	Identify language & Aggregate class counts
NER	Count "person" entities & Aggregate total counts
Parity	Detect odd or even number & Aggregate counts
Sentiment	Detect sentiment polarity & Aggregate counts
Word	Count target word "women" & Aggregate total counts
WSD	Identify "apple" word sense & Aggregate counts

Table 1: Overview of the selected tasks and their aggregation logic in the MIP setting.

4.1 Individual Tasks

We consider eight heterogeneous tasks[1] for our analysis, as summarised in Table 1. Each task is chosen such that it can be solved individually in the SIP setting by standard LLMs. When multiple instances are provided in the MIP setting, the model must additionally aggregate outputs across all instances (e.g., counting how many movie reviews are classified as positive in sentiment analysis). Detailed task descriptions and examples are provided in Appendix A.

4.2 Models and Prompting

We evaluate nine LLMs[2], including six open-weight models (*DeepSeek V3*, *gpt-oss-120b*, *gpt-oss-20b*, *Llama 3.3*, *Llama 4 Maverick*, and *Qwen3-Instruct*) and three closed-source models (*Gemini 2.5 Flash*, *GPT-5 Nano*, and *Grok 4 Fast*).

For prompting, we use a temperature of 0 and a maximum output length of 20K tokens for consistency across models. To allow limited tolerance to formatting errors, we permit up to three retries when a model produces an invalid output belonging to $\mathcal{I}$. The full set of prompting templates is provided in Appendix B.

[1]We removed three additional tasks whose SIP performance fell below our requirements.

[2]As with the task filtering, we removed two open-weights LLMs whose SIP performance did not meet our criteria.

Task	Agreement (%)	Max (%)	Min (%)
Arithmetic	89.0	99.5	93.7
Category	94.8	99.2	97.2
Language	98.8	99.6	99.2
NER	87.6	97.3	93.6
Parity	100.0	100.0	100.0
Sentiment	96.4	99.7	97.8
Word	98.3	100.0	99.4
WSD	98.6	99.4	99.1

Table 2: Task filtering results showing agreement (i.e., the percentage of instances for which all LLMs produce correct SIP predictions), as well as the maximum and minimum SIP success rates across all LLMs. The actual agreement and minimum rate for *Parity* is 99.96%.

4.3 Single-Instance Filtering

As described in Section 3.2, we ensure that each instance can be successfully solved in the SIP setting. To this end, we conduct SIP experiments on 2,500 instances for each task[3]. We report each LLM's SIP performance for each task in Appendix C.

Table 2 reports the percentage of instances retained (i.e., agreement) for each task (89%–nearly 100%), and the corresponding maximum and minimum SIP success rates across models, all exceeding 93%. This filtering retains only unambiguous instances and excludes potential annotation errors.

4.4 MIP Sampling

After single-instance filtering, for each task τ we construct MIP inputs by sampling instances from $\mathcal{X}_{\text{SIP}}$ using five different random seeds ($s \in \{1, 2, 3, 4, 5\}$). We evaluate ten MIP sample sizes $n \in \mathcal{N} = \{2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000\}$. For each (τ, n, s), we prompt each model m with the corresponding instance set, retaining only instances for which all models are correct in the SIP setting.

5 RQ1: Performance and Failure Behaviours

Our main goal is to evaluate LLM performance and failure behaviours in MIP settings, particularly as the number of instances increases.

[3]The original dataset of *Category* contains fewer than 2,500 instances, and each instance is substantially longer. We therefore use 250 instances instead of 2,500 for this task. Correspondingly, the maximum MIP sample size for *Category* is also ten times smaller than for the other tasks.

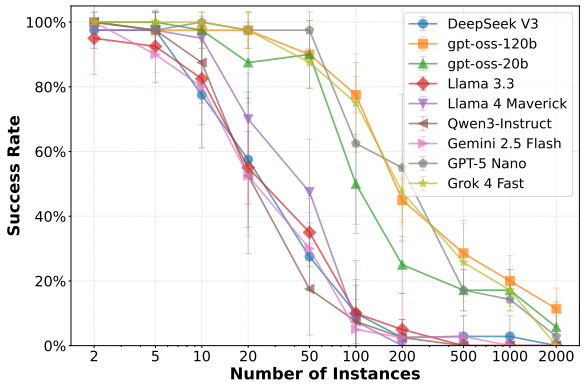

Figure 2: Model success rates (averaged across all tasks) as a function of the number of instances. Error bars indicate standard deviation across five random seeds.

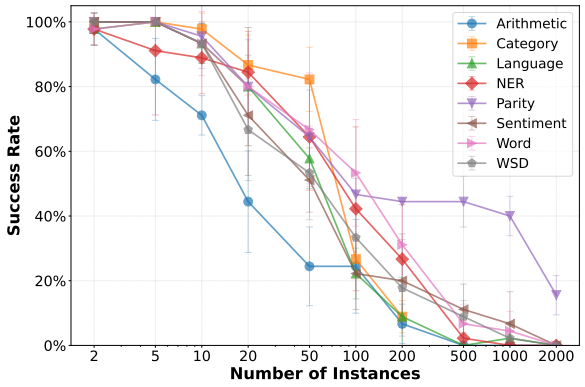

Figure 3: Task success rates (averaged across all LLMs) as a function of the number of instances. Error bars indicate standard deviation across five random seeds.

Model	Size	Success (%)	Invalid (%)
DeepSeek V3	A37B	39.0±3.6	2.9±0.6
gpt-oss-120b	117B	**68.3**±2.8	3.6±1.1
gpt-oss-20b	21B	60.8±2.5	4.9±1.2
Llama 3.3	70B	39.0±3.8	2.9±0.9
Llama 4 Maverick	17B	43.1±1.1	**0.0**±0.0
Qwen3-Instruct	A22B	37.9±3.6	1.3±0.0
Gemini 2.5 Flash	/	37.7±3.9	4.2±3.2
GPT-5 Nano	/	66.5±3.8	7.5±0.6
Grok 4 Fast	/	67.0±2.8	**0.0**±0.0

Table 3: Model success rate and invalid rate (mean±std), averaged across all tasks and instance counts. Standard deviation is computed over five random seeds. The top six models are open-weight LLMs (DeepSeek V3 is a 671B and Qwen3-Instruct is a 235B mixture-of-experts LLM.), while the bottom three are closed-source LLMs with undisclosed sizes.

5.1 Performance Analysis

Success rate drops as the number of instances increases. Figure 2 reports success rates aggregated across tasks for each model. We observe a consistent performance degradation as the number of instances increases. In particular, all models experience a pronounced drop beyond 500 instances, where success rates fall below 30%. Figure 3 shows success rates aggregated across models for each task and reveals a similar downward trend. With the exception of *Arithmetic*, all tasks achieve success rates above 50% when fewer than 50 instances are processed. Performance then deteriorates steadily as the instance count grows. Complete results by model and task are reported in Appendix D.1.

LLM comparison. Table 3 reports the success rate and invalid rate for all models. Overall, closed-source LLMs do not exhibit superior performance, possibly because the most advanced proprietary models are not included in our evaluation. Among the evaluated models, *gpt-oss-120b*, *Grok 4 Fast*, and *GPT-5 Nano* achieve the highest success rates (above 65%). Notably, only *Grok 4 Fast* produces no invalid outputs, indicating greater robustness. While all models achieve success rates above 35% on average, Table 3 and Figure 2 together show that most successful cases occur when the number of instances is below 500.

5.2 Failure Behaviours

Beyond our default setting, which requires only an aggregated answer, we introduce an additional variant for more fine-grained analysis. In this variant, models are required to produce instance-level predictions $\{y_i\}_{i=1}^{n}$ before providing the aggregated answer. Even with such instance-level predictions, which provide an explicit chain-of-thought (Wei et al., 2022), the relative performance of models remains similar to the aggregated-only setting.

As described in Section 3.1, we consider two broad categories of failures: wrong answers and invalid outputs. Wrong answers include errors at the individual-instance level, the aggregation level, or both. Invalid outputs include (1) parsing errors, where the model output cannot be reliably parsed into the expected structured format, and (2) overlong input errors, where the input exceeds the model's allowable context length. To analyse failure behaviours in greater detail, we use the instance-level variant in the following experiments.

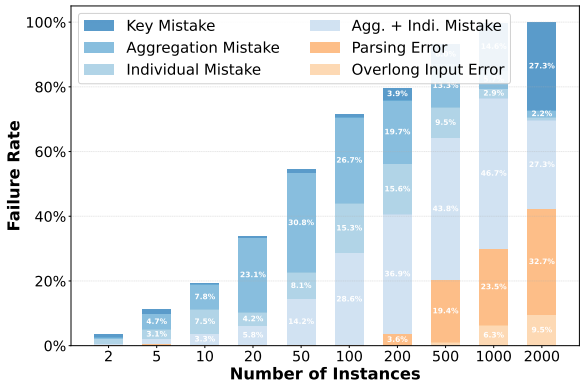

Figure 4: Breakdown of failure types. Key mistakes, aggregation mistakes, individual mistakes, and combined mistakes (Agg+Indi.) are categorised as wrong answers (blue), while parsing errors and overlong input errors are categorised as invalid outputs (orange).

Model	Wrong Exp.	Wrong Indv. Per Exp.
DeepSeek V3	29.6±2.7	34.8±13.1
gpt-oss-120b	16.2±2.9	37.5±17.4
gpt-oss-20b	15.8±2.4	11.8±5.0
Llama 3.3	35.6±2.2	53.8±26.7
Llama 4 Maverick	33.6±4.2	38.4±4.6
Qwen3-Instruct	29.4±2.1	20.8±3.1
Gemini 2.5 Flash	13.4±1.3	5.5±3.2
GPT-5 Nano	11.2±3.1	10.3±15.3
Grok 4 Fast	22.4±1.7	117.3±8.0

Table 4: Percentage of experiments with at least one incorrect instance-level prediction (Wrong Exp.) and the average number of incorrect instance-level predictions per failed experiment (mean±std). Standard deviation is computed over five random seeds.

Different failure types emerge as the number of instances increases. Figure 4 presents a stacked bar plot showing the contribution of each failure type as the instance count increases. Blue bars correspond to wrong answers, while orange bars correspond to invalid outputs. Wrong answers can occur even with as few as two instances. When the instance count exceeds 200, parsing errors increase substantially and reach nearly 30% at 2,000 instances. Overlong input errors emerge beyond 500 instances, primarily due to the *Language* task, as non-English inputs typically require more prompt tokens (Petrov et al., 2023).

Mistakes in individual instances and aggregation. To further characterise wrong answers, we define four error types: key mistakes (keys do not span from 1 to n), individual mistakes (at least one instance-level prediction is incorrect), aggregation mistakes (all instance-level predictions are correct but the aggregated answer is incorrect), and combined mistakes (both instance-level and aggregation errors occur). As shown in Figure 4, when the instance count exceeds 200, key mistakes increase markedly, accounting for approximately 25% to 45% of failures. Moreover, when aggregation mistakes and combined mistakes are considered together, aggregation remains challenging for LLMs regardless of instance-level correctness or instance count. Complete results by model and task are reported in Appendix D.2.

Model differences in making mistakes. Table 4 compares models in terms of how individual-instance mistakes are distributed across experiments. Some models tend to concentrate many individual mistakes within a small number of failed experiments, while others exhibit more frequent but sparser errors. Focusing on individual mistakes, we observe that each failed experiment of *Grok 4 Fast* typically contains many incorrect instance-level predictions. In contrast, *Gemini 2.5 Flash* more often produces failed experiments with only a small number of individual mistakes.

Self-awareness of limitations. Ideally, an LLM should recognise its own capability boundaries. A desirable behaviour is for the model to explicitly acknowledge in its reasoning r when it cannot handle a large number of instances (e.g., by suggesting batch-wise processing or explicitly stating that the instance count exceeds its capacity). Such behaviour can be reflected by the model producing predictions for only the first few instances and omitting the remaining ones. After inspecting the model outputs, we found that only 159 out of 3,465 experiments exhibit this omission, almost exclusively at instance counts of 500 or more, as expected. However, a manual analysis shows that only 27 out of these 159 experiments explicitly suggest batch-wise processing, and most models do not warn users about such limitations. *GPT-5 Nano* demonstrates this behaviour most frequently, explicitly indicating difficulty in 19 out of 28 cases. In contrast, *DeepSeek V3*, *gpt-oss-120b*, and *gpt-oss-20b* does so in fewer than 20% of its omission cases, while no such behaviour is observed for the remaining models. Notably, although *Qwen3-Instruct* and *Gemini 2.5 Flash* do not acknowledge limitations,

they also produce almost no cases of this omission.

5.3 Discussion

Overall, our findings answer RQ1 by showing that as the number of instances increases, all LLMs experience degraded performance, characterised by lower success rates and more frequent failures. In particular, when the instance count reaches 500 or higher, no LLM achieves a success rate above 40%.

Among the evaluated LLMs, *gpt-oss-120b*, *Grok 4 Fast*, and *GPT-5 Nano* exhibit the strongest overall performance. Although *Gemini 2.5 Flash* achieves relatively low success rates, its failures typically involve only one or two incorrect instances. At the same time, invalid outputs become increasingly common as the instance count grows. Notably, *GPT-5 Nano* is the only model that tends to identify when a task exceeds its capacity.

6 RQ2: Context Length vs Number of Instances

In the previous section, we analysed LLM behaviour in MIP settings as the number of instances increases. The results revealed consistent performance degradation across all models, albeit at different rates and to different extents. A natural question is whether this degradation is simply driven by increased context length, as has been observed in prior work across a range of tasks and settings (see Section 2). In this section, we analyse the effects of context length and instance count jointly, and attempt to disentangle their respective contributions.

6.1 Context Length Augmentation

To isolate the impact of context length, we design a setting in which the length of each individual instance is artificially increased without altering its original meaning.[4] For each sampled instance x, we construct a perturbed instance $x' = x + \epsilon$, where ϵ denotes injected noise. Following Hsieh et al. (2024), we define ϵ as the string "– IRRELEVANT CONTEXT:" followed by seven repetitions of the sentence "The grass is green. The sky is blue. The sun is yellow. Here we go. There and back again.". This choice is motivated by prior work showing that even irrelevant context can degrade model performance (Shi et al., 2023; Yang et al., 2025). After noise injection, the average length of each instance (measured in the number of prompt tokens) more than doubles, increasing from approximately 136 tokens in the default setting to 314 tokens in the artificially augmented setting.

[4]We exclude *Parity* from this experiment because each instance contains only a single number. For consistency with context length constraints, we also cap the maximum sample size at $n = 1000$ in this setting, as discussed below.

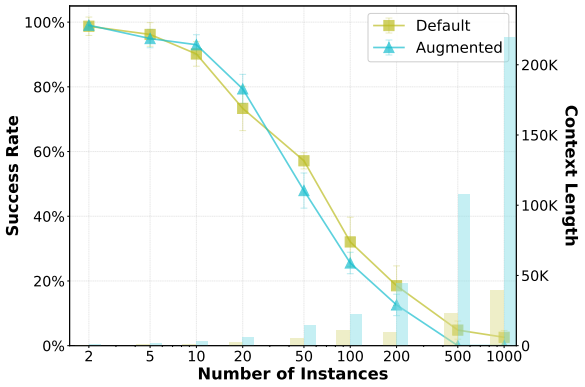

Figure 5: Success rate (lines) and total prompt token length (bars) in the artificial length setting as the number of instances increases. Error bars indicate standard deviation across five random seeds.

Results. Figure 5 compares average performance and context length across all tasks and models between the default and artificially augmented settings. When the number of instances is held constant, the success rates of the two settings remain broadly similar, despite the average context length being more than twice as large in the augmented setting. This indicates that artificially increasing the length of individual instances does not substantially impact performance, suggesting that context length alone is not the primary driver of performance degradation in MIP settings.

6.2 Correlation Analysis

Motivated by the above findings, we conduct a correlation analysis to assess the respective contributions of instance count and context length to overall model performance. Recall that in our default setting, instance samples are constructed using five random seeds. While this random sampling introduces some variation in individual instance lengths, the average prompt length becomes increasingly similar across samples as the number of instances grows. To obtain a wider range of context lengths for correlation analysis, we augment the default samples with two additional variants drawn from $\mathcal{X}_{\text{SIP}}$: a *long* set, consisting of the longest instances in each dataset, and a *short* set, consisting of the shortest instances. This design enables a clearer separation of the effects of context length by introducing greater variation in total input length.

Number of Instances	Success rate and Context length	
	Correlation	P-Value
2	0.051	0.1 – 0.5
5	0.048	> 0.5
10	0.047	> 0.5
20	-0.047	> 0.5
50	-0.021	> 0.5
100	0.007	> 0.5
200	-0.026	> 0.5
500	0.002	> 0.5
1000	-0.059	0.1 – 0.5

Table 5: Correlation between success rate and prompt token length when the number of instances is fixed.

Correlation with instance count and total context length. As an initial analysis, we compute the correlation between success rate and each factor independently: the number of instances and the total context length. Both exhibit strong negative correlations, indicating that performance decreases as either quantity increases. However, the number of instances shows a notably stronger relationship, with Spearman correlations of -0.58 and -0.37, respectively. In both cases, the corresponding p-values are below 0.001, indicating that the correlations are highly unlikely to arise by chance.

Correlation conditioned on the number of instances. The two factors are inherently related, as total context length grows with the number of instances. To disentangle their effects, we additionally compute correlations while holding the number of instances fixed and examining variation only in context length. As shown in Table 5, the resulting correlations between context length and success rate are substantially weaker, with values ranging between -0.1 and 0.1, and p-values consistently above 0.1 (often exceeding 0.5). These results indicate that context length alone has limited explanatory power in this setting and strongly suggest that the performance degradation observed in the Section 6.1 is primarily driven by the number of instances rather than total input length.

6.3 Discussion

Based on the above results, we conclude that the number of instances is a more dominant factor than context length in determining model success rates. When the number of instances is held fixed, the effect of context length appears to be negligible according to our correlation analysis. LLMs have been shown to struggle when required to perform many repeated operations (Son et al., 2024; Fu et al., 2024), which is precisely what MIP entails as the instance count increases. This behaviour contrasts with retrieval-augmented generation (RAG) settings, where models primarily need to identify relevant contexts rather than process all inputs exhaustively. In MIP settings, LLMs must process each instance individually and aggregate the resulting outputs. While prior work has shown that LLMs can handle increasingly long contexts (Liu et al., 2025), our findings suggest that improving reliability in MIP settings may require training strategies that explicitly target multi-instance reasoning and aggregation.

7 Conclusion

In this paper, we presented a comprehensive evaluation of LLMs in MIP settings, that is, tasks that require aggregation of information from multiple instances to produce a final answer. The results show that LLMs are generally able to solve tasks involving a small number of instances, but begin to make mistakes as the number of instances increases. While the errors are initially small, this has important implications in user trustworthiness, since models are able to consistently solve the task when only a single instance is given. As the number of instances further increases, models become unable to solve the tasks, in most cases without being aware of their own limitations.

Crucially, our experiments with respect to context length highlight the importance of reasoning at the instance-count level, rather than focusing solely on context length as is commonly done. This has implications for how context should be processed in batches, for example when developing data science agents. Instead of relying only on context length for batching decisions, it is advisable to take the number of instances into account in MIP settings. More generally, these results have broader implications for how models should be trained to handle MIP effectively, especially in settings where accuracy is paramount.

Limitations

This work focuses on diagnosing failure modes of LLMs in multi-instance processing (MIP) settings, rather than proposing or validating concrete solutions. While our controlled evaluation reveals consistent performance degradation as the number

of instances increases, we do not evaluate mitigation strategies such as task decomposition, external tool use, verification, or agentic designs. As a result, this paper should be interpreted primarily as an empirical characterisation of model behaviour rather than as a prescription for improving MIP reliability.

Our experiments emphasise exact aggregation tasks (e.g., counting, summation, exact class frequencies). Although these tasks are common in analytics-style applications, they may overemphasise brittleness in settings where approximate or semantic aggregation would suffice. The extent to which our findings generalise to softer aggregation objectives (e.g., majority voting, summarisation, or trend identification) therefore remains an open question.

Despite our efforts to disentangle instance count from context length, these two factors are not entirely independent in practice. While correlation and controlled noise-injection analyses suggest that instance count is the dominant driver of degradation, more fine-grained causal analyses (e.g., controlled computational complexity or attention-level diagnostics) are left for future work.

Finally, our study does not include model-internal interpretability analyses such as attention patterns or hidden-state dynamics, nor does it evaluate training-time interventions. In addition, our experiments are English-centric and limited to a set of selected LLMs, which do not include the latest proprietary models due to time and budget constraints. Future work could explore cross-lingual MIP behaviour and architectural or training modifications explicitly designed for multi-instance reasoning.

Ethical Considerations

This work presents an empirical evaluation of LLMs in multi-instance processing settings and does not involve the collection of new data, interaction with human subjects, or the deployment of models in real-world decision-making systems. All datasets used in our experiments are publicly available and open-source, and were accessed and processed in accordance with their original licences and intended research use. Also, we do not claim that any particular model or provider is inherently unsafe or unsuitable. Rather, our results reflect general limitations of current LLMs under specific experimental conditions. We hope that this work contributes to more responsible deployment of LLM-based systems by encouraging practitioners to consider instance-level reliability, aggregation strategies, and potential failure modes when designing real-world applications.

References

Chenxin An, Jun Zhang, Ming Zhong, Lei Li, Shansan Gong, Yao Luo, Jingjing Xu, and Lingpeng Kong. 2025. Why does the effective context length of llms fall short? In *The Thirteenth International Conference on Learning Representations*.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R Gormley. 2025. Oolong: Evaluating long context reasoning and aggregation capabilities. *arXiv preprint arXiv:2511.02817*.

Ke Chen, Peiran Wang, Yaoning Yu, Xianyang Zhan, and Haohan Wang. 2025. Large language model-based data science agent: A survey. *arXiv preprint arXiv:2508.02744*.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch prompting: Efficient inference with large language model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore. Association for Computational Linguistics.

Alexis Conneau and Douwe Kiela. 2018. SentEval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tairan Fu, Raquel Ferrando, Javier Conde, Carlos Arriaga, and Pedro Reviriego. 2024. Why do large language models (llms) struggle to count letters? *arXiv preprint arXiv:2412.18626*.

Manuel Gozzi and Federico Di Maio. 2024. Comparative analysis of prompt strategies for large language models: Single-task vs. multitask prompts. *Electronics*, 13(23):4712.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd international conference on Machine learning*, pages 377–384.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. Ds-agent: automated data science by empowering large language models

with case-based reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 16813–16848.

Sil Hamilton, Rebecca MM Hicke, Matthew Wilkens, and David Mimno. 2025. Too long, didn't model: Decomposing llm long-context understanding with novels. *arXiv preprint arXiv:2505.14925*.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Ceyao Zhang, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Robert Tang, Xiangtao Lu, and 9 others. 2025. Data interpreter: An LLM agent for data science. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19796–19821, Vienna, Austria. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Zhaoxuan Ji, Xinlu Wang, Zhaojing Luo, Zhongle Xie, and Meihui Zhang. 2025. Optimized batch prompting for cost-effective llms. *Proceedings of the VLDB Endowment*, 18(7):2172–2184.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. 2024a. Batchprompt: Accomplish more with less. In *The Twelfth International Conference on Learning Representations*.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeigham, Aditya G Parameswaran, and Eugene Wu. 2024b. Towards accurate and efficient document analytics with large language models. *arXiv preprint arXiv:2405.04674*.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, and 1 others. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Daniel Loureiro, Kiamehr Rezaee, Mohammad Taher Pilehvar, and Jose Camacho-Collados. 2021. Analysis and evaluation of language models for word sense disambiguation. *Computational Linguistics*, 47(2):387–443.

Hyeonseok Moon and Heuiseok Lim. 2025. Needlechain: Measuring intact long-context reasoning capability of large language models. *arXiv preprint arXiv:2507.22411*.

Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, and Tomas Pfister. 2025. Ds-star: Data science agent via iterative planning and verification. *arXiv preprint arXiv:2509.21825*.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990.

Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164, Florence, Italy. Association for Computational Linguistics.

Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. 2025. Llm-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G Parameswaran, and Eugene Wu. 2024. Docetl: Agentic query rewriting and evaluation for complex document processing. *arXiv preprint arXiv:2410.12189*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Guijin Son, SangWon Baek, Sangdae Nam, Ilgyun Jeong, and Seungone Kim. 2024. Multi-task inference: Can large language models follow multiple instructions at once? In *Proceedings of the 62nd Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 5606–5627, Bangkok, Thailand. Association for Computational Linguistics.

Maojun Sun, Ruijian Han, Binyan Jiang, Houduo Qi, Defeng Sun, Yancheng Yuan, and Jian Huang. 2025. A survey on large language model-based agents for statistics and data science. *The American Statistician*, pages 1–14.

Martin Thoma. 2018. The wili benchmark dataset for written language identification. *arXiv preprint arXiv:1801.07779*.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, and 1 others. 2024. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Zhengxiang Wang, Jordan Kodner, and Owen Rambow. 2025. Exploring limitations of LLM capabilities with multi-problem evaluation. In *The Sixth Workshop on Insights from Negative Results in NLP*, pages 121–140, Albuquerque, New Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Yang Wang, and Liangming Pan. 2025. How is LLM reasoning distracted by irrelevant context? an analysis using a controlled benchmark. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13329–13347, Suzhou, China. Association for Computational Linguistics.

Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. 2025. Longproc: Benchmarking long-context language models on long procedural generation. *arXiv preprint arXiv:2501.05414*.

Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2025. Helmet: How to evaluate long-context models effectively and thoroughly. In *The Thirteenth International Conference on Learning Representations*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and 1 others. 2024. ∞Bench: Extending long context evaluation beyond 100k tokens. *arXiv preprint arXiv:2402.13718*.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. 2025. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*.

A Task and Data Source

Here we introduce the details for each task and their data source.

A.1 Arithmetic

This task is about **arithmetic** calculation, which uses questions from Mathematics Dataset (Saxton et al., 2019), including only easy addition or subtraction questions from its training split. The average word count for this task is 4.7.

Example Input:

1. What is the difference between -2 and 251860?
2. -9,259,432 + 1
3. What is 1,141.09 less than 1?

SIP Ground Truth Output:

1. 251,862
2. -9,259,431
3. -1,140.09

MIP Question: Solve all the provided arithmetic questions and calculate the sum of all answers.

MIP Ground Truth Output: -9,008,709.09

A.2 Category

This task is about news **category** classification, where a news can belong to "tech" or "business" or "entertainment" or "politics" or "sport". The dataset is BBC News (Greene and Cunningham, 2006), where we use its training split. The average word count for this task is 371.4.

Example input:

1. german business confidence slides german business confidence fell in february knocking hopes of a speedy recovery in europe s largest economy...
2. bbc poll indicates economic gloom citizens in a majority of nations surveyed in a bbc world service poll believe the world economy is worsening...
3. lifestyle governs mobile choice faster better or funkier hardware alone is not going to help phone firms sell more handsets research suggests...

SIP Ground Truth Output:

1. business
2. business
3. tech

MIP Question: Count how many of the provided news articles belong to the "tech" category.

MIP Ground Truth Output: 1

A.3 Language

This task is about **language** identification, where a paragraph can belong to "English" or "Chinese" or "Persian" or "Spanish". The dataset is WiLI-2018 (Thoma, 2018). The average word count for this task is 55.8.

Example Input:

1. Nordahl Road is a station served by North County Transit District's SPRINTER light rail line...
2. En Navidad de 1974, poco después de que interpretó la canción en francés película Papillon (Toi qui Regarde la Mer)...
3. A talk by Takis Fotopoulos about the Internationalization of the Capitalist Market Economy and the project of Inclusive Democracy...

SIP Ground Truth Output:

1. English
2. Spanish
3. English

MIP Question: Count how many paragraphs are in English.

MIP Ground Truth Output: 2

A.4 NER

The task is about **n**amed **e**ntity **r**ecognition, which uses data from WikiANN (Rahimi et al., 2019). The average word count for this task is 16.0.

Example Input:

1. we love everything about the fence .
2. i want to hook up with that girl paige in the brown leather jacket .
3. in addition , there is a reduction of 22,101 mmbtu which is the difference between the scada values (best available) that anita showed on the february 29th storage sheet and the " official " february 29th values that gary wilson received from mips .

SIP Ground Truth Output:

1. 0
2. 1
3. 2

MIP Question: Count occurrences of the entity 'PERSON' in all sentences.

MIP Ground Truth Output: 3

A.5 Parity

The task is about **parity** classification (i.e., identify if a number is "odd" or "even"), where we use synthetic data generated by ourselves. The average word count for this task is 1 since only a single number is provided.

Example Input:

1. 18010
2. 10160
3. 89449

SIP Ground Truth Output:

1. even
2. even
3. odd

MIP Question: Count how many of the provided numbers are odd.

MIP Ground Truth Output: 1

A.6 Sentiment

The task is about **sentiment** analysis, where a movie review can belong to "positive" or "negative". The dataset is Sentiment Treebank (Socher et al., 2013), where we only use the "most" positive and negative reviews to avoid ambiguity. The average word count for this task is 18.7.

Example Input:

1. High Crimes is a cinematic misdemeanor , a routine crime thriller remarkable only for its lack of logic and misuse of two fine actors , Morgan Freeman and Ashley Judd .
2. One of the worst movies of the year .
3. A mix of gritty realism , crisp storytelling and radiant compassion that effortlessly draws you in .

SIP Ground Truth Output:

1. negative
2. negative
3. positive

MIP Question: Count how many of the provided movie reviews are positive.

MIP Ground Truth Output: 1

A.7 Word

The task is about tweets **word** occurrence (i.e., count a target word's occurrences in given tweets). The dataset is TweetEval (Barbieri et al., 2020), where we use its stance detection subset. The average word count for this task is 17.3.

Example Input:

1. IF FEMINISTS WERE HONEST "I want a worldwide matriarchal dictatorship with all men enslaved to women" #GamerGate #SemST
2. What the fuck do women even do? I mean seriously they're just useless other than sex. #womensrights #Feminist #SemST
3. DEAR FEMINISTS Start asking for accountability from man-haters instead of shielding them for convenient concealment. #SemST

SIP Ground Truth Output:

1. 1
2. 2
3. 0

MIP Question: Count occurrences of the word "women" in all tweets.

MIP Ground Truth Output: 3

A.8 WSD

The task is about **w**ord **s**ense **d**isambiguation, where a target word "apple" is required to be distinguished as meaning either "company" or "fruit" based on its context. The dataset is CoarseWSD-20 (Loureiro et al., 2021), where we use its "apple" subset. The average word count for this task is 31.4.

Example Input:

1. both seasons are available for download from apple 's itunes store .
2. in klayman ii , the plaintiffs sued the same government defendants and in addition , facebook , yahoo! , google , microsoft , youtube , aol , paltalk , skype , sprint , at&t , apple again alleging the bulk metadata collection violates the first , fourth and fifth amendment and constitutes divulgence of communication records in violation of section 2702 of stored communications act .
3. description alongside dried pears the filling also contains raisin , walnut and other dried fruit such as apple or figs .

SIP Ground Truth Output:

1. company
2. company
3. fruit

MIP Question: Count how many paragraphs contain the word "apple" referring to the company (Apple Inc.), not the fruit.

MIP Ground Truth Output: 2

A.9 Excluded Tasks

Beyond the tasks mentioned above, we have three additional tasks that have been filtered out due to unsatisfactory SIP performance:

- **B**igram **Shift** detection: from SentEval (Conneau and Kiela, 2018), which checks whether a bigram in a sentence has been shifted, with binary outcomes (i.e., shifted or not).
- **Subj**ect **Num**ber identification: from SentEval (Conneau and Kiela, 2018), which checks whether the subject of a sentence is "plural" or "singular".
- **Voice** classification: from Universal Dependencies (Silveira et al., 2014), which checks whether a sentence is in the "active" or "passive" voice. Since no ground-truth labels are available, we use rule-based approaches to annotate the dataset.

B Prompt Template

Here we present the prompt templates we use in our experiments. We example inputs in Appendix A for illustration.

B.1 Default Setting

B.1.1 Arithmetic

```
Task: Solve all the provided arithmetic
↪  questions and calculate the sum of
↪  all answers.

Instructions:
- Calculate the answer for each
↪  arithmetic operation
- Sum all individual answers
- Provide exact values without
↪  unnecessary trailing zeros (e.g.,
↪  "5" not "5.0")
- Prefix negatives with '-' (e.g.,
↪  "-42")
- For decimal results, keep only
↪  necessary decimal places (e.g.,
↪  "3.14" not "3.140")
- If you only receive one question, the
↪  sum is just its answer

Response format:
Return a JSON object with:
- "reasoning": briefly explain your
↪  calculation process
- "answer": sum of all answers as a
↪  string

Example:
{"reasoning": "your approach here",
↪  "answer": "42"}

=== Here are the arithmetic questions
↪  ===
Question 1: What is the difference
↪  between -2 and 251860?
Question 2: -9259432 + 1
Question 3: What is 1141.09 less than 1?
...
=== End of arithmetic questions ===
```

B.1.2 Category

```
Task: Count how many of the provided news articles belong to the 'tech' category.

Background:
- Each news article belongs to one of 5 categories:
* business
* entertainment
* politics
* sport
* tech
- You need to classify each article based on its content and context

Instructions:
- Read each news article carefully
- Identify which category each article belongs to based on the content
- Count how many articles belong to the 'tech' category
- Do not count articles from other categories (business, entertainment, politics, sport)
- If you only receive one article, return 1 if it's tech, else return 0

Response format:
Return a JSON object with:
- "reasoning": briefly explain your approach to classifying news categories and how you counted
- "answer": integer count of tech news articles

Example:
{"reasoning": "your approach here", "answer": 42}

=== Here are the news articles ===
Article 1: german business confidence slides german business confidence fell in february knocking hopes of a speedy recovery in europe s largest economy...
Article 2: bbc poll indicates economic gloom citizens in a majority of nations surveyed in a bbc world service poll believe the world economy is worsening...
Article 3: lifestyle  governs mobile choice  faster  better or funkier hardware alone is not going to help phone firms sell more handsets research suggests...
...
=== End of news articles ===
```

B.1.3 Language

```
Task: Count how many of the provided paragraphs are written in English.

Background:
The paragraphs are written in one of four languages:
- English (label 0)
- Chinese (label 1)
- Persian (label 2)
- Spanish (label 3)

Instructions:
- Read each paragraph carefully
- Identify the language of each paragraph
- Count how many paragraphs are written in English
- Do not count paragraphs in other languages
- If you only receive one paragraph, return 1 if it's English, else return 0

Response format:
Return a JSON object with:
- "reasoning": briefly explain your approach to identifying English paragraphs
- "answer": integer count of English paragraphs

Example:
{"reasoning": "your approach here", "answer": 5}

=== Here are the paragraphs ===
Paragraph 1: Nordahl Road is a station served by North County Transit District's SPRINTER light rail line...
Paragraph 2: En Navidad de 1974, poco después de que interpretó la canción en francés película Papillon (Toi qui Regarde la Mer)...
```

```
Paragraph 3: A talk by Takis Fotopoulos
↪ about the Internationalization of
↪ the Capitalist Market Economy and
↪ the project of Inclusive
↪ Democracy...
...
=== End of paragraphs ===
```

B.1.4 NER

```
Task: Count how many times the entity
↪ "PERSON" appears across all provided
↪ sentences.

Background:
- An entity may consist of multiple
↪ words that form a contiguous
↪ fragment in the text
- You need to first identify entities in
↪ the sentence (named entity
↪ recognition), then count them
- Two entities may appear consecutively
↪ without punctuation or words between
↪ them
- No entity overlaps occur (each word
↪ belongs to at most one entity)
- Some words do not belong to any entity

Entity Definition:
- PERSON: names of people, real or
↪ fictional, but not nominals

Instructions:
- Identify all PERSON entities across
↪ all sentences
- Count the total number of PERSON
↪ entity mentions (not unique
↪ entities, but total occurrences)
- Each distinct mention counts as one
↪ occurrence, even if it refers to the
↪ same person

Response format:
Return a JSON object with:
- "reasoning": briefly explain how you
↪ identified the entities and counted
↪ them
- "answer": integer count of PERSON
↪ entities

Example:
{"reasoning": "your approach here",
↪ "answer": 5}

=== Here are the sentences ===
Sentence 1: we love everything about the
↪ fence .
Sentence 2: i want to hook up with that
↪ girl paige in the brown leather
↪ jacket .
Sentence 3: in addition , there is a
↪ reduction of 22,101 mmbtu which is
↪ the difference between the scada
↪ values ( best available ) that anita
↪ showed on the february 29th storage
↪ sheet and the " official " february
↪ 29th values that gary wilson
↪ received from mips .
...
=== End of sentences ===
```

B.1.5 Parity

```
Task: Count how many of the provided
↪ numbers are odd.

Background:
- An odd number is an integer that is
↪ not evenly divisible by 2
- Odd numbers end in 1, 3, 5, 7, or 9
- An even number is an integer that is
↪ evenly divisible by 2
- Even numbers end in 0, 2, 4, 6, or 8

Instructions:
- Check each number to determine if it
↪ is odd or even
- Count how many numbers are odd
- Do not count even numbers
- If you only receive one number, return
↪ 1 if it's odd, else return 0

Response format:
Return a JSON object with:
- "reasoning": briefly explain your
↪ approach to identifying odd numbers
- "answer": integer count of odd numbers

Example:
{"reasoning": "your approach here",
↪ "answer": 3}

=== Here are the numbers ===
Number 1: 18010
Number 2: 10160
Number 3: 89449
...
```

```
=== End of numbers ===
```

B.1.6 Sentiment

```
Task: Count how many of the provided
↪  movie reviews are positive.

Instructions:
- Each review has a sentiment: positive
↪  or negative
- Count only the reviews with positive
↪  sentiment
- Return the total count of positive
↪  reviews
- If you only receive one review, return
↪  1 if it's positive, else return 0

Response format:
Return a JSON object with:
- "reasoning": briefly explain how you
↪  solved this task
- "answer": integer count of positive
↪  reviews

Example:
{"reasoning": "your approach here",
↪  "answer": 42}

=== Here are the movie reviews ===
Review 1: High Crimes is a cinematic
↪  misdemeanor , a routine crime
↪  thriller remarkable only for its
↪  lack of logic and misuse of two fine
↪  actors , Morgan Freeman and Ashley
↪  Judd .
Review 2: One of the worst movies of the
↪  year .
Review 3: A mix of gritty realism ,
↪  crisp storytelling and radiant
↪  compassion that effortlessly draws
↪  you in .
...
=== End of movie reviews ===
```

B.1.7 Word

```
Task: Count how many times the word
↪  "women" appears in the provided
↪  tweets.

Instructions:
- Search is case-insensitive (e.g.,
↪  "women", "Women", "WOMEN" all count)
- Count occurrences that include the
↪  substring "women" (e.g., "women",
↪  "womens", "women's", "womenfolk")
- Do not count forms that lack the
↪  substring "women" (e.g., "woman",
↪  "womankind")
- Count all occurrences across all
↪  tweets, not just unique tweets
- If you only receive one tweet, just
↪  return the occurrence for that tweet

Response format:
Return a JSON object with:
- "reasoning": briefly explain how you
↪  counted the matches
- "answer": integer total count

Example:
{"reasoning": "your approach here",
↪  "answer": 42}

=== Here are the tweets ===
Tweet 1: IF FEMINISTS WERE HONEST   "I
↪  want a worldwide matriarchal
↪  dictatorship with all men enslaved
↪  to women"   \#GamerGate \#SemST
Tweet 2: What the fuck do women even do?
↪  I mean seriously they're just
↪  useless other than sex.
↪  \#womensrights \#Feminist \#SemST
Tweet 3: DEAR FEMINISTS  Start asking
↪  for accountability from man-haters
↪  instead of shielding them for
↪  convenient concealment.  \#SemST
...
=== End of tweets ===
```

B.1.8 WSD

```
Task: Count how many paragraphs contain
↪  the word "apple" referring to the
↪  company (Apple Inc.), not the fruit.

Background:
- Each paragraph contains exactly one
↪  occurrence of the word "apple"
↪  (case-insensitive, as a complete
↪  word, not as part of another word)
- This "apple" can mean either:
* The company: Apple Inc., the
↪  technology company
* The fruit: the edible fruit that grows
↪  on apple trees
```

```
- You need to determine the meaning of
↪  "apple" in each paragraph based on
↪  context

Instructions:
- Read each paragraph carefully
- Identify whether "apple" refers to the
↪  company or the fruit based on
↪  contextual clues
- Count how many paragraphs where
↪  "apple" means the company (Apple
↪  Inc.)
- Do not count paragraphs where "apple"
↪  means the fruit
- If you only receive one paragraph,
↪  return 1 if it means the company,
↪  else return 0

Response format:
Return a JSON object with:
- "reasoning": briefly explain your
↪  approach to disambiguating the word
↪  sense and how you counted
- "answer": integer count of paragraphs
↪  where "apple" means the company

Example:
{"reasoning": "your approach here",
↪  "answer": 42}

=== Here are the paragraphs ===
Paragraph 1: both seasons are available
↪  for download from apple 's itunes
↪  store .
Paragraph 2: in klayman ii , the
↪  plaintiffs sued the same government
↪  defendants and in addition ,
↪  facebook , yahoo! , google ,
↪  microsoft , youtube , aol , paltalk
↪  , skype , sprint , at\&t , apple
↪  again alleging the bulk metadata
↪  collection violates the first ,
↪  fourth and fifth amendment and
↪  constitutes divulgence of
↪  communication records in violation
↪  of section 2702 of stored
↪  communications act .
Paragraph 3: description alongside dried
↪  pears the filling also contains
↪  raisin , walnut and other dried
↪  fruit such as apple or figs .
...
=== End of paragraphs ===
```

B.2 Instance-Level Setting

B.2.1 Arithmetic

```
Task: Solve each of the provided
↪  arithmetic questions and calculate
↪  the sum of all answers.

Instructions:
- Calculate the answer for each
↪  arithmetic question
- Use 1-based indexing for question
↪  numbers ("1", "2", "3", . . . )
- Provide exact values without
↪  unnecessary trailing zeros (e.g.,
↪  "5" not "5.0")
- Prefix negatives with '-' (e.g.,
↪  "-42")
- For decimal results, keep only
↪  necessary decimal places (e.g.,
↪  "3.14" not "3.140")
- Sum all individual answers

Response format:
Return a JSON object with:
- One key per question: "1", "2", "3",
↪  . . . , mapping to the answer as a
↪  string
- "sum": sum of all answers as a string
- "reasoning": brief explanation of your
↪  approach

Example for 3 questions:
{"1": "8", "2": "-30", "3": "5.6",
↪  "sum": "-16.4", "reasoning": "your
↪  approach here"}

=== Here are the arithmetic questions
↪  ===
Question 1: What is the difference
↪  between -2 and 251860?
Question 2: -9259432 + 1
Question 3: What is 1141.09 less than 1?
...
=== End of arithmetic questions ===
```

B.2.2 Category

```
Task: For each news article, classify it
↪  into one of 5 categories, then
↪  provide total counts for each
↪  category.
```

```
Background:
- Each news article belongs to one of 5
↪  categories:
* business (label 0)
* entertainment (label 1)
* politics (label 2)
* sport (label 3)
* tech (label 4)
- You need to classify each article
↪  based on its content and context

Instructions:
- Read each news article carefully and
↪  classify it into the most
↪  appropriate category
- For each article, assign:
* 0 if it's business news
* 1 if it's entertainment news
* 2 if it's politics news
* 3 if it's sport news
* 4 if it's tech news
- Count the total number of articles for
↪  each category
- Provide classification for each
↪  article along with summary counts

Response format:
Return a JSON object with:
- One key per article: "1", "2", "3",
↪  . . . , mapping to the category label
↪  (0-4)
- "business": integer count of articles
↪  classified as business
- "entertainment": integer count of
↪  articles classified as entertainment
- "politics": integer count of articles
↪  classified as politics
- "sport": integer count of articles
↪  classified as sport
- "tech": integer count of articles
↪  classified as tech
- "reasoning": brief explanation of your
↪  approach to news classification

Example for 3 articles:
{"1": 4, "2": 0, "3": 3, "business": 1,
↪  "entertainment": 0, "politics": 0,
↪  "sport": 1, "tech": 1, "reasoning":
↪  "your approach here"}

=== Here are the news articles ===
Article 1: german business confidence
↪  slides german business confidence
↪  fell in february knocking hopes of a
↪  speedy recovery in europe s largest
↪  economy...
Article 2: bbc poll indicates economic
↪  gloom citizens in a majority of
↪  nations surveyed in a bbc world
↪  service poll believe the world
↪  economy is worsening...
Article 3: lifestyle  governs mobile
↪  choice  faster  better or funkier
↪  hardware alone is not going to help
↪  phone firms sell more handsets
↪  research suggests...
...
=== End of news articles ===
```

B.2.3 Language

```
Task: For each paragraph, identify which
↪  language it is written in, then
↪  provide summary counts for all
↪  categories.

Background:
The paragraphs are written in one of
↪  four languages:
- English (label 0)
- Chinese (label 1)
- Persian (label 2)
- Spanish (label 3)

Instructions:
- Read each paragraph carefully and
↪  identify its language
- Classify each paragraph using the
↪  labels:
* 0 = English
* 1 = Chinese
* 2 = Persian
* 3 = Spanish
- Provide the classification for each
↪  individual paragraph
- Also provide summary counts for all
↪  four language categories

Response format:
Return a JSON object with:
- One key per paragraph: "1", "2", "3",
↪  . . . , mapping to the classification
↪  (0, 1, 2, or 3)
- "english": integer count of paragraphs
↪  classified as English
```

```
- "chinese": integer count of paragraphs
↪  classified as Chinese
- "persian": integer count of paragraphs
↪  classified as Persian
- "spanish": integer count of paragraphs
↪  classified as Spanish
- "reasoning": brief explanation of your
↪  approach

Example for 5 paragraphs:
{"1": 0, "2": 1, "3": 2, "4": 3, "5": 0,
↪  "english": 2, "chinese": 1,
↪  "persian": 1, "spanish": 1,
↪  "reasoning": "your approach here"}

=== Here are the paragraphs ===
Paragraph 1: Nordahl Road is a station
↪  served by North County Transit
↪  District's SPRINTER light rail
↪  line...
Paragraph 2: En Navidad de 1974, poco
↪  después de que interpretó la canción
↪  en francés película Papillon (Toi
↪  qui Regarde la Mer)...
Paragraph 3: A talk by Takis Fotopoulos
↪  about the Internationalization of
↪  the Capitalist Market Economy and
↪  the project of Inclusive
↪  Democracy...
...
=== End of paragraphs ===
```

B.2.4 NER

```
Task: Count how many times the entity
↪  "PERSON" appears in each sentence
↪  and provide the total count.

Background:
- An entity may consist of multiple
↪  words that form a contiguous
↪  fragment in the text
- You need to first identify entities in
↪  each sentence (named entity
↪  recognition), then count them
- Two entities may appear consecutively
↪  without punctuation or words between
↪  them
- No entity overlaps occur (each word
↪  belongs to at most one entity)
- Some words do not belong to any entity

Entity Definition:
- PERSON: names of people, real or
↪  fictional, but not nominals

Instructions:
- Identify all PERSON entities in each
↪  sentence
- Count the number of PERSON entity
↪  mentions in each sentence separately
- Provide the count for each sentence
↪  along with the total count across
↪  all sentences
- Each distinct mention counts as one
↪  occurrence, even if it refers to the
↪  same person

Response format:
Return a JSON object with:
- One key per sentence: "1", "2", "3",
↪  . . . , mapping to the integer count
↪  of PERSON mentions in that sentence
- "total": total count of PERSON
↪  entities across all sentences
- "reasoning": brief explanation of how
↪  you identified and counted the
↪  PERSON entities

Example:
{"1": 0, "2": 2, "3": 1, "total": 3,
↪  "reasoning": "your approach here"}

=== Here are the sentences ===
Sentence 1: we love everything about the
↪  fence .
Sentence 2: i want to hook up with that
↪  girl paige in the brown leather
↪  jacket .
Sentence 3: in addition , there is a
↪  reduction of 22,101 mmbtu which is
↪  the difference between the scada
↪  values ( best available ) that anita
↪  showed on the february 29th storage
↪  sheet and the " official " february
↪  29th values that gary wilson
↪  received from mips .
...
=== End of sentences ===
```

B.2.5 Parity

```
Task: For each number, identify whether
↪  it is odd or even, then provide
↪  summary counts for both categories.
```

```
Background:
- An odd number is an integer that is not evenly divisible by 2
- Odd numbers end in 1, 3, 5, 7, or 9 (label 1)
- An even number is an integer that is evenly divisible by 2
- Even numbers end in 0, 2, 4, 6, or 8 (label 0)

Instructions:
- Check each number to determine if it is odd or even
- Classify each number using the labels:
* 1 = odd
* 0 = even
- Provide the classification for each individual number
- Also provide summary counts for both odd and even categories

Response format:
Return a JSON object with:
- One key per number: "1", "2", "3", . . . , mapping to the classification (0 or 1)
- "odd": integer count of numbers classified as odd
- "even": integer count of numbers classified as even
- "reasoning": brief explanation of your approach

Example for 5 numbers:
{"1": 0, "2": 1, "3": 0, "4": 1, "5": 1, "odd": 3, "even": 2, "reasoning": "your approach here"}

=== Here are the numbers ===
Number 1: 18010
Number 2: 10160
Number 3: 89449
...
=== End of numbers ===
```

B.2.6 Sentiment

```
Task: For each movie review, classify whether it is positive or negative, then provide summary counts.

Instructions:
- Classify each review as either:
- 0 = negative sentiment
- 1 = positive sentiment
- Provide the classification for each individual review
- Also provide summary counts for negative and positive reviews

Response format:
Return a JSON object with:
- One key per review: "1", "2", "3", . . . , mapping to the classification (0 or 1)
- "negative": integer count of reviews classified as negative
- "positive": integer count of reviews classified as positive
- "reasoning": brief explanation of your approach to classification

Example for 3 reviews:
{"1": 1, "2": 0, "3": 1, "negative": 1, "positive": 2, "reasoning": "your approach here"}

=== Here are the movie reviews ===
Review 1: High Crimes is a cinematic misdemeanor , a routine crime thriller remarkable only for its lack of logic and misuse of two fine actors , Morgan Freeman and Ashley Judd .
Review 2: One of the worst movies of the year .
Review 3: A mix of gritty realism , crisp storytelling and radiant compassion that effortlessly draws you in .
...
=== End of movie reviews ===
```

B.2.7 Word

```
Task: For each tweet, count how many times the word "women" appears, then provide the total count.

Instructions:
- Search is case-insensitive (e.g., "women", "Women", "WOMEN" all count)
- Count occurrences that include the substring "women" (e.g., "women", "womens", "women's", "womenfolk")
- Do not count forms that lack the substring "women" (e.g., "woman", "womankind")
```

```
- Count occurrences in each tweet
↪  separately
- Provide the per-tweet counts plus the
↪  overall total

Response format:
Return a JSON object with:
- One key per tweet: "1", "2", "3",
↪  . . . , mapping to the count of
↪  "women" in that tweet (integer)
- "total": integer representing the
↪  total count of "women" across all
↪  tweets
- "reasoning": brief explanation of your
↪  counting approach

Example for 3 tweets:
{"1": 2, "2": 0, "3": 1, "total": 3,
↪  "reasoning": "your approach here"}

=== Here are the tweets ===
Tweet 1: IF FEMINISTS WERE HONEST   "I
↪  want a worldwide matriarchal
↪  dictatorship with all men enslaved
↪  to women"   \#GamerGate \#SemST
Tweet 2: What the fuck do women even do?
↪  I mean seriously they're just
↪  useless other than sex.
↪  \#womensrights \#Feminist \#SemST
Tweet 3: DEAR FEMINISTS  Start asking
↪  for accountability from man-haters
↪  instead of shielding them for
↪  convenient concealment.  \#SemST
...
=== End of tweets ===
```

B.2.8 WSD

```
Task: For each paragraph, identify
↪  whether the word "apple" refers to
↪  the company or the fruit, then
↪  provide total counts.

Background:
- Each paragraph contains exactly one
↪  occurrence of the word "apple"
↪  (case-insensitive, as a complete
↪  word, not as part of another word)
- This "apple" can mean either:
* The company (label 0): Apple Inc., the
↪  technology company
* The fruit (label 1): the edible fruit
↪  that grows on apple trees
- You need to determine the meaning of
↪  "apple" in each paragraph based on
↪  context

Instructions:
- Read each paragraph carefully and
↪  classify the meaning of "apple"
- For each paragraph, assign:
* 0 if "apple" means the company (Apple
↪  Inc.)
* 1 if "apple" means the fruit
- Count the total number of paragraphs
↪  for each category
- Provide classification for each
↪  paragraph along with summary counts

Response format:
Return a JSON object with:
- One key per paragraph: "1", "2", "3",
↪  . . . , mapping to either 0 (company)
↪  or 1 (fruit)
- "company": integer count of paragraphs
↪  where "apple" means the company
- "fruit": integer count of paragraphs
↪  where "apple" means the fruit
- "reasoning": brief explanation of your
↪  approach to word sense
↪  disambiguation

Example for 3 paragraphs:
{"1": 0, "2": 1, "3": 0, "company": 2,
↪  "fruit": 1, "reasoning": "your
↪  approach here"}

=== Here are the paragraphs ===
Paragraph 1: both seasons are available
↪  for download from apple 's itunes
↪  store .
Paragraph 2: in klayman ii , the
↪  plaintiffs sued the same government
↪  defendants and in addition ,
↪  facebook , yahoo! , google ,
↪  microsoft , youtube , aol , paltalk
↪  , skype , sprint , at\&t , apple
↪  again alleging the bulk metadata
↪  collection violates the first ,
↪  fourth and fifth amendment and
↪  constitutes divulgence of
↪  communication records in violation
↪  of section 2702 of stored
↪  communications act .
```

Task	Agreement (%)
BShift	69.3
SubjNum	82.5
Voice	78.3

Table 6: Task filtering across tasks that have been removed.

```
Paragraph 3: description alongside dried
↪  pears the filling also contains
↪  raisin , walnut and other dried
↪  fruit such as apple or figs .
...
=== End of paragraphs ===
```

C Single Instance Filtering Result

Table 6 reports the agreements for the tasks that are removed, while Table 7 shows the average SIP success rate for each model and task. Although *BShift* achieves a relatively high SIP success rate for the selected models, its agreement across models is low.

D Experimental Results

Here we present the success rate and failure breakdown for each model and task.

D.1 Success Rate

D.1.1 Task success rate for models

Figure 6 and Figure 7 show task success rate for each model.

D.1.2 Model success rate for tasks

Figure 8 and Figure 9 show model success rate for each task.

D.2 Failure Breakdown

D.2.1 Failure breakdown for models

Figure 10 and Figure 11 show failure breakdown for models, averaged across all tasks.

D.2.2 Failure breakdown for tasks

Figure 12 and Figure 13 show failure breakdown for tasks, averaged across all models.

Model	Arithmetic	Language	NER	News	Parity	BShift	SubjNum	Sentiment	Tweets	Voice	WSD	Average
DeepSeek V3	98.3	99.2	95.8	97.6	100.0	92.7	88.3	99.3	99.6	89.5	99.4	96.3
gpt-oss-120b	97.2	99.5	96.2	97.6	100.0	92.6	89.4	98.5	100.0	91.4	99.3	96.5
gpt-oss-20b	97.7	99.5	96.2	97.2	100.0	90.4	89.6	97.8	100.0	90.6	99.1	96.2
Llama 3.3	93.7	99.5	93.6	98.4	100.0	90.9	87.4	99.7	99.5	86.9	99.2	95.3
Llama 4 Maverick	96.7	99.3	94.3	99.2	100.0	90.9	88.9	99.4	99.9	91.7	99.4	96.3
Llama 4 Scout	93.4	97.9	93.9	97.2	100.0	86.9	87.3	99.0	99.5	88.7	99.1	94.8
Mistral NeMo	72.0	96.2	78.1	96.8	75.3	62.9	75.2	83.8	60.3	72.7	98.4	79.2
Qwen3-Instruct	99.5	99.4	97.3	99.2	100.0	91.3	88.9	99.5	99.7	92.8	99.3	97.0
Gemini 2.5 Flash	96.2	99.6	96.5	98.0	100.0	92.8	88.9	99.0	99.4	92.0	99.4	96.5
GPT-5 Nano	98.0	99.5	96.5	97.2	100.0	93.0	89.0	99.2	100.0	91.3	99.2	96.6
Grok 4 Fast	97.9	99.5	97.2	98.8	100.0	95.7	88.6	99.5	100.0	90.2	99.4	97.0

Table 7: SIP success rate (%) across tasks. Open-weight models are listed first, followed by closed-source models. Underlined rows and columns indicate the tasks and LLMs that are excluded because they do not satisfy the filtering criteria described in Section 3.2.

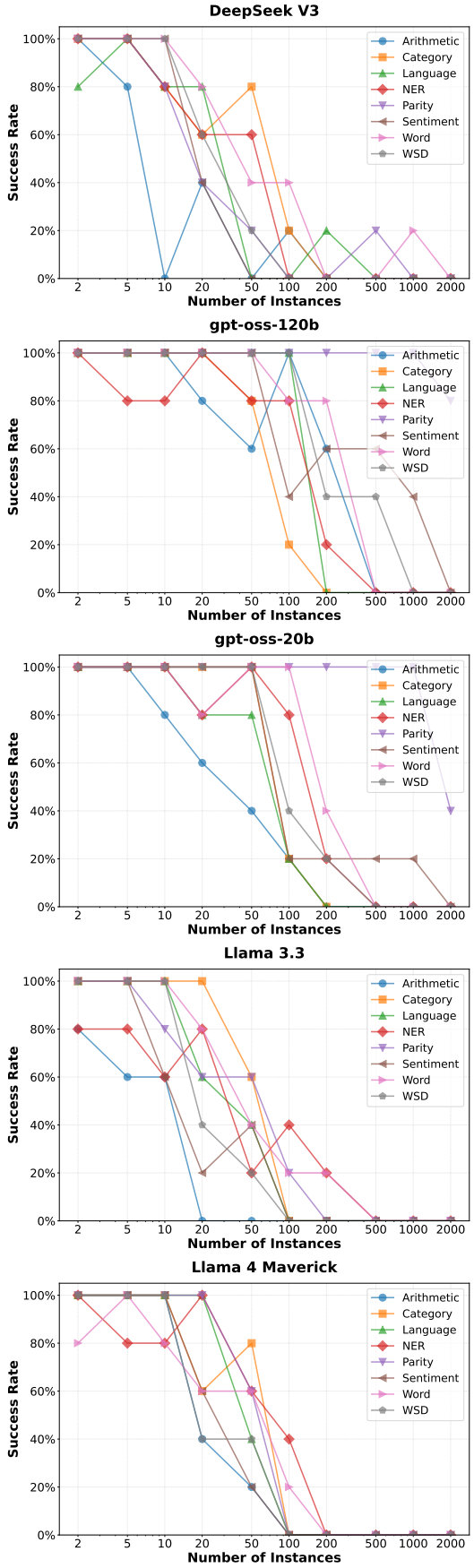

Figure 6: Success rate of models.

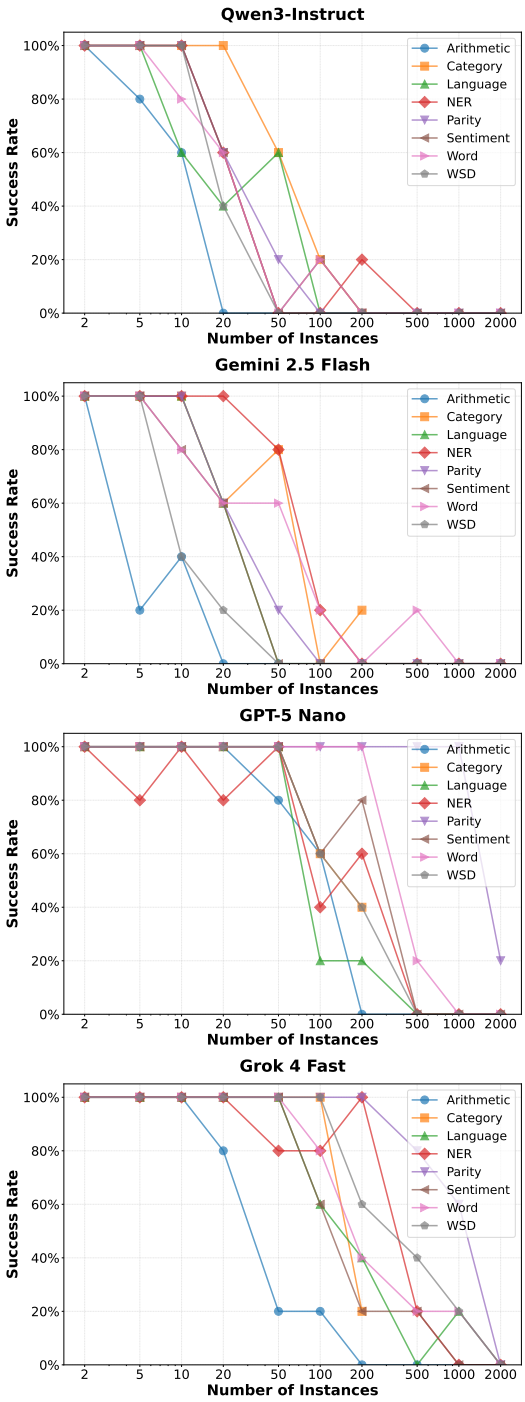

Figure 7: Success rate of models.

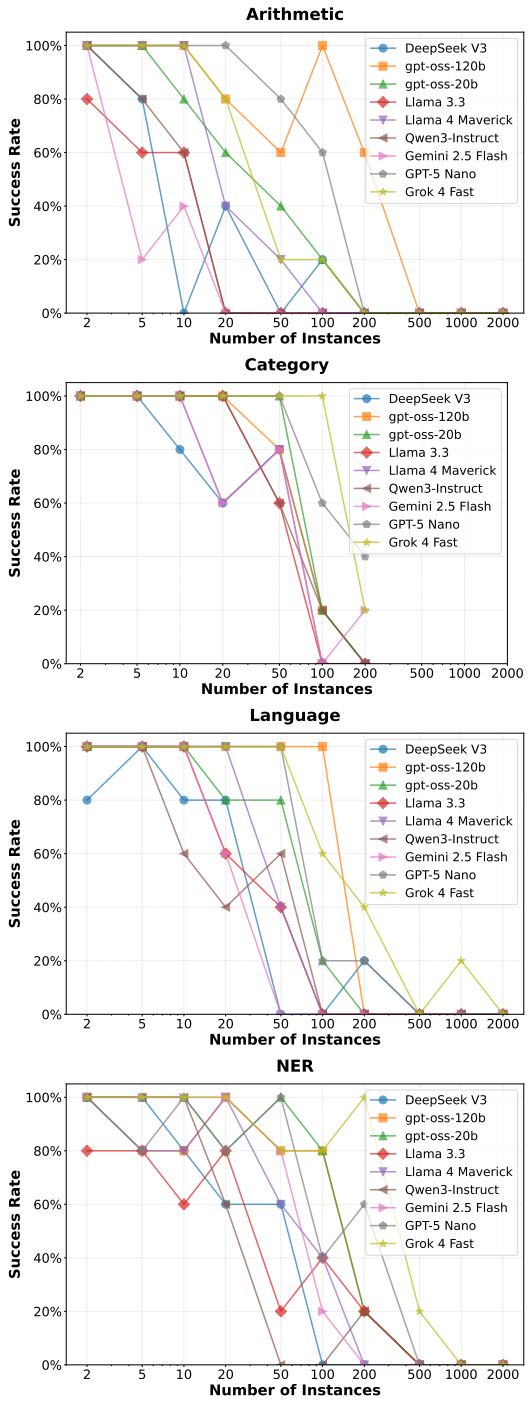

Figure 8: Model success rate for tasks.

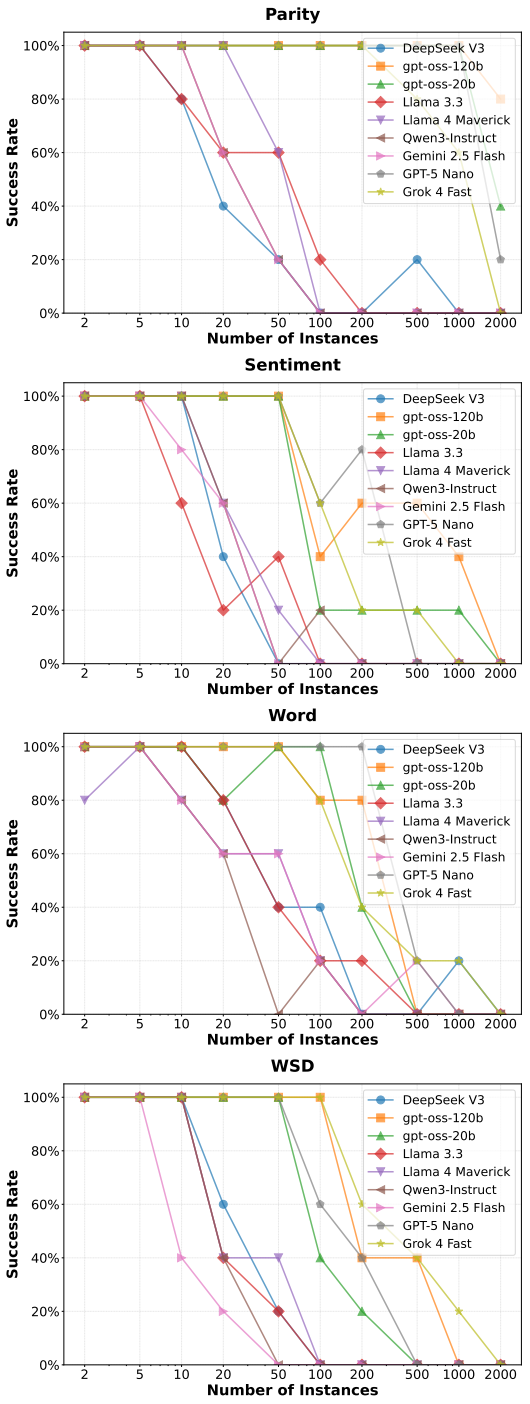

Figure 9: Model success rate for tasks.

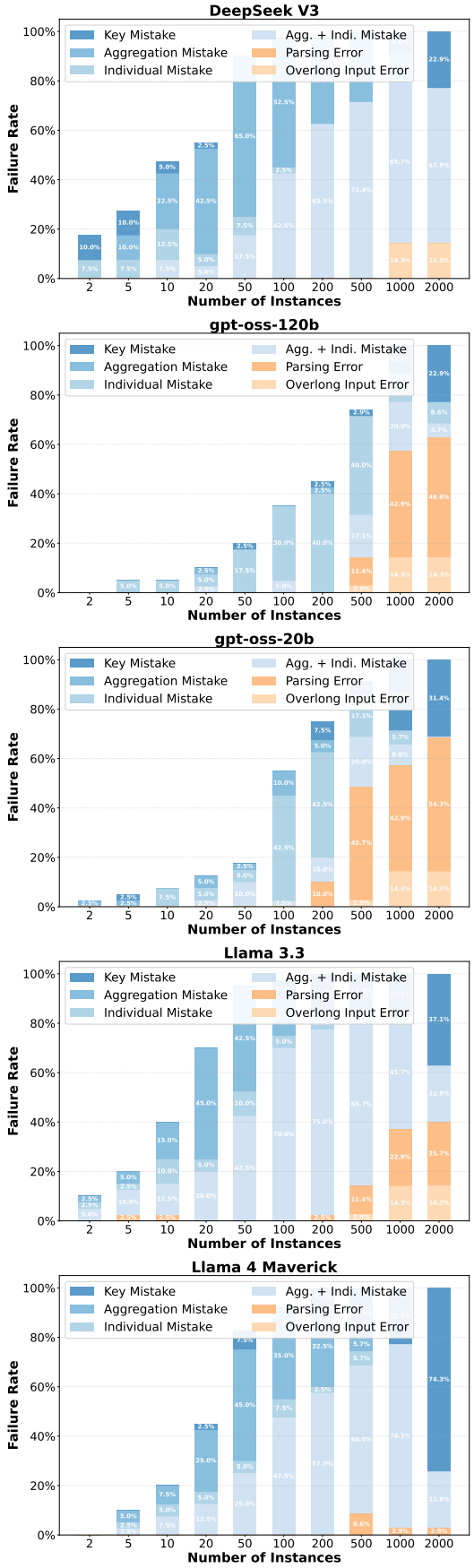

Figure 10: Failure breakdown for models.

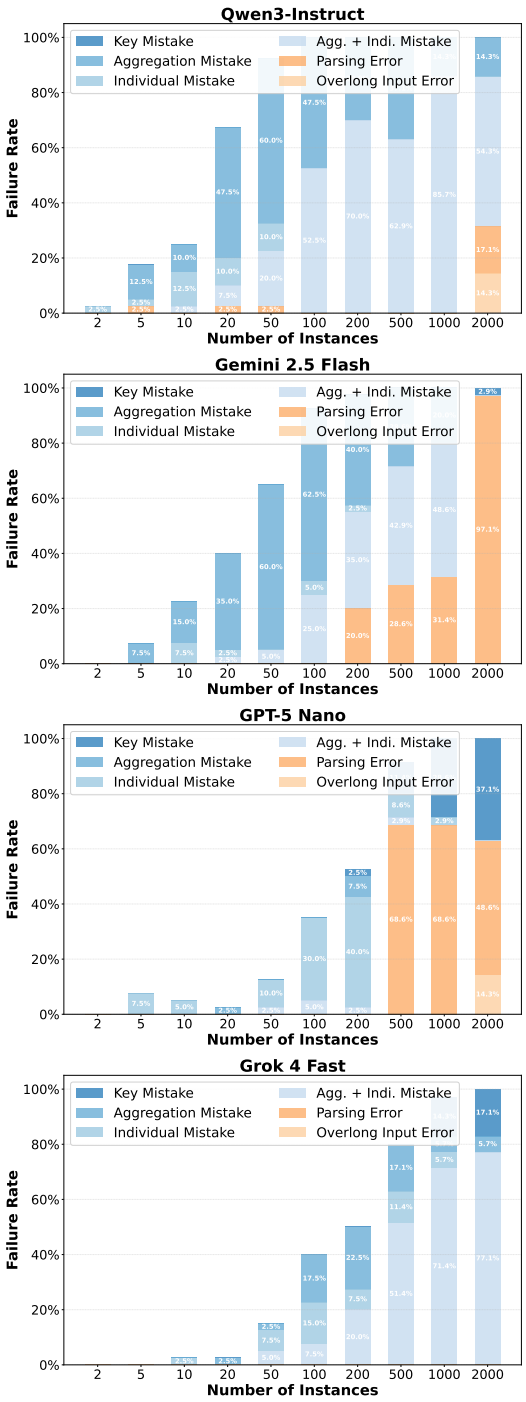

Figure 11: Failure breakdown for models.

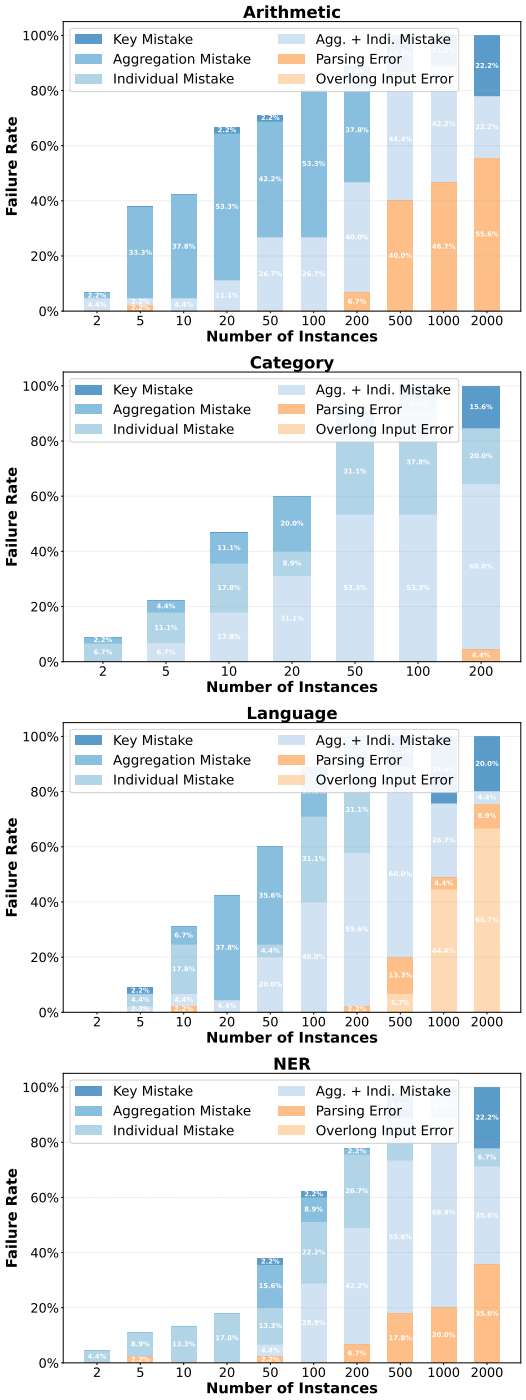

Figure 12: Failure breakdown for tasks.

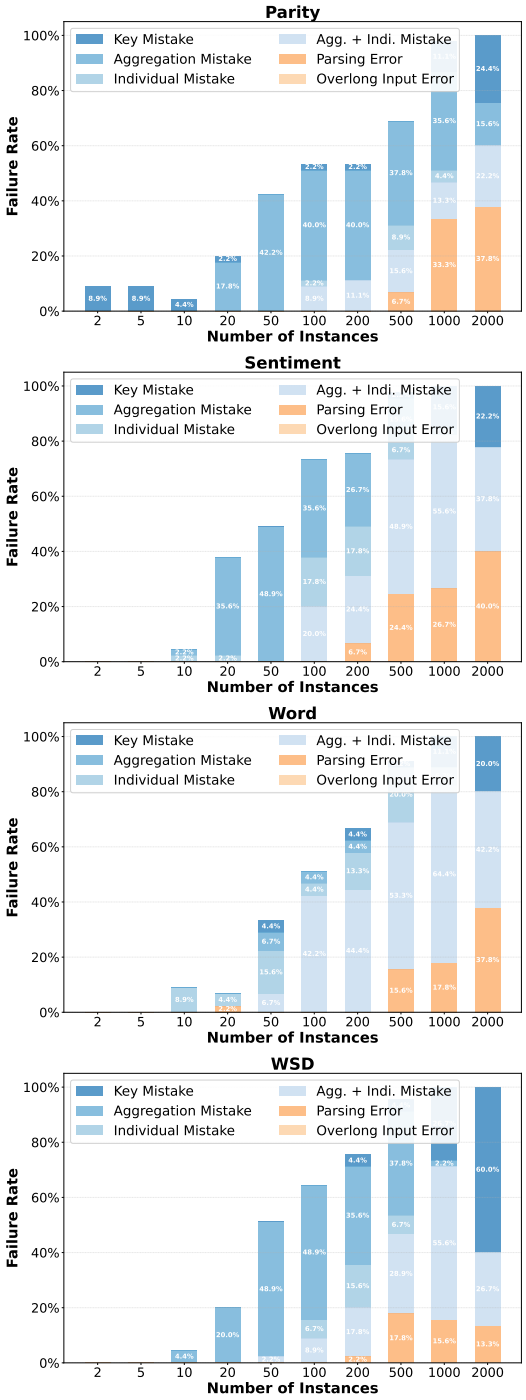

Figure 13: Failure breakdown for tasks.