Attention Overflow: Language Model Input Blur during Long-Context Missing Items Identification

Damien Sileo

Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 - CRIStAL, F-59000 Lille, France

damien.sileo@inria.fr

Abstract

Large language models (LLMs) can suggest missing elements from items listed in a prompt, which can be used for list completion or similar item recommendation. However, their performance degrades when they are exposed to too many items, as they start to suggest items already included in the input list. This occurs at around 100 items for mid-2024 flagship LLMs. We evaluate this phenomenon on both synthetic problems (e.g., finding missing numbers in a given range of shuffled integers) and realistic movie recommendation scenarios. We refer to this issue as *attention overflow*, as avoiding repetition requires attending to all items simultaneously. Although iterative loops can mitigate this problem, their costs increase with the repetition rate, affecting the language models' ability to derive novelty from lengthy inputs.

1 Introduction

Large language models (LLMs) boast ever-growing context windows, enabling new potential applications. However, the theoretical context length is not a sufficient indication of a model's real performance with a given input size (Liu et al., 2024). Multiple benchmarks have been proposed to stress-test the actual ability of language models to reason over long contexts. These tasks either involve pure retrieval or a form of reasoning requiring the identification of a few relevant portions from a large context.

We question the effective context window of language models from an opposing angle: asking them to provide the only relevant elements that are *not* in a large input. We formulate this as a missing item prediction task. Missing item prediction has multiple applications, notably in conversational recommendation, where users can provide a list of items (e.g. movies) they liked and ask for new suggestions. This task involves a form of inductive reasoning, in contrast to the deductive reasoning typically explored in long context stress tests. More importantly, it requires comparing a representation to the whole input, and we notice that this is difficult for current LLMs, which leads to the prediction of items already in the input (repetition).

Missing item prediction is also relevant when models are asked to generate long lists. We observed repetitions in this scenario[1], but we focus on the movie recommendation use case, where users provide the movies they have watched, and we also create synthetic examples, notably number ranges with a missing element. We quantify the repetition phenomenon with existing off-the-shelf language models and investigate whether fine-tuning can easily address this problem. The created datasets are publicly available[2].

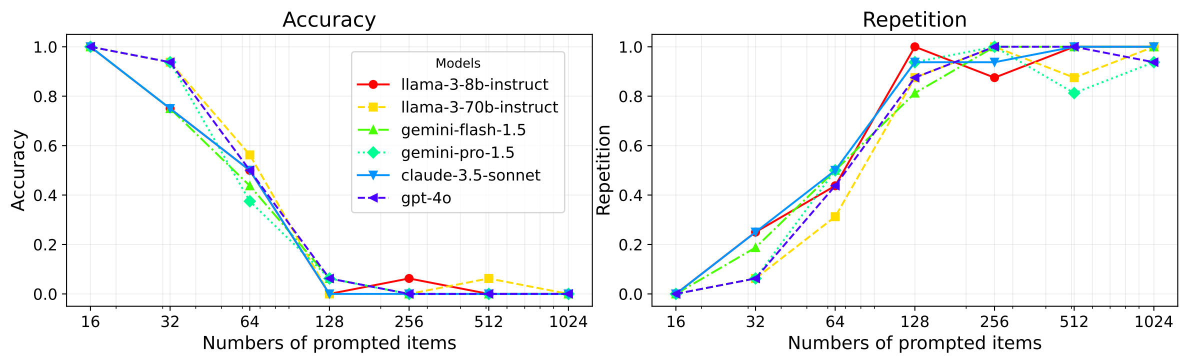
2 Related work

Repetitions in language modeling We study a form of repetitions, a well-identified problem in language models (Keskar et al., 2019), which can sometimes lead to text degeneration, where models repeat the same token indefinitely (Fu et al., 2021). Repetition penalties were proposed to alleviate this issue (Keskar et al., 2019), but they operate at the token level and cannot scale to large contexts where all tokens are already represented. Repetitions also exist in more subtle ways, as Chiang and Lee (2024) showed that chain-of-thought reasoning contains redundant content.

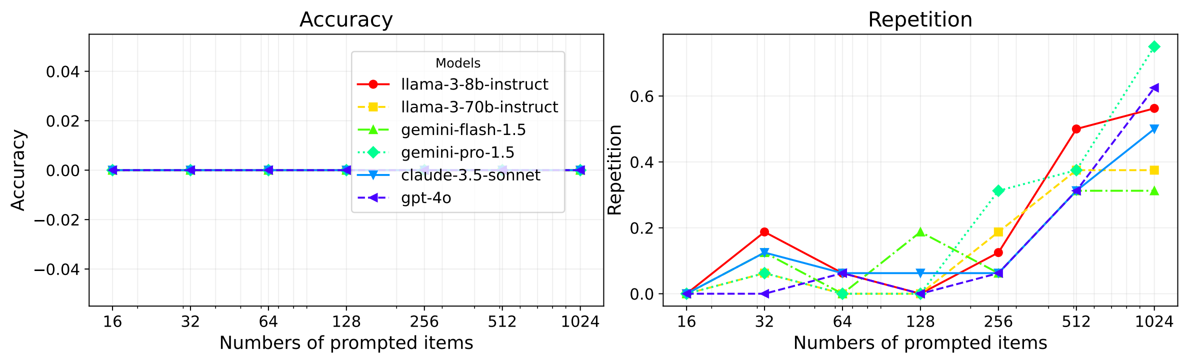
LLM context length stress tests Our work is also related to context window stress testing and language modeling-based recommendation. Previous work has studied the ability of attention mechanisms to identify what is present in long contexts, but not what is missing. The Long-Range Arena

[1]For example, asking Claude Sonnet 3.5 200 movies released in 2022 leads to numerous repetitions: [artifact]

[2] code: Included in reasoning_core (Lacombe et al., 2025) as a subtask. data:HF-datasets



(a) Zero-shot missing number prediction



(b) Zero-shot missing movie prediction

Figure 1: Zero-shot test accuracy and repetition rate with increasing itemset sizes.

(Tay et al., 2021) provides the first systematic analysis of the long-range processing capabilities of text encoders, focusing primarily on algorithmic reasoning and retrieval tasks. BABILong (Kuratov et al., 2024) uses bAbi reasoning tasks (Weston et al., 2016) and interleaves relevant text with irrelevant input. FlenQA (Levy et al., 2024) applies a similar process to the RuleTaker (Clark et al., 2020) deductive logical reasoning task. Ruler (Hsieh et al., 2024) uses simple algorithmic/retrieval tasks.

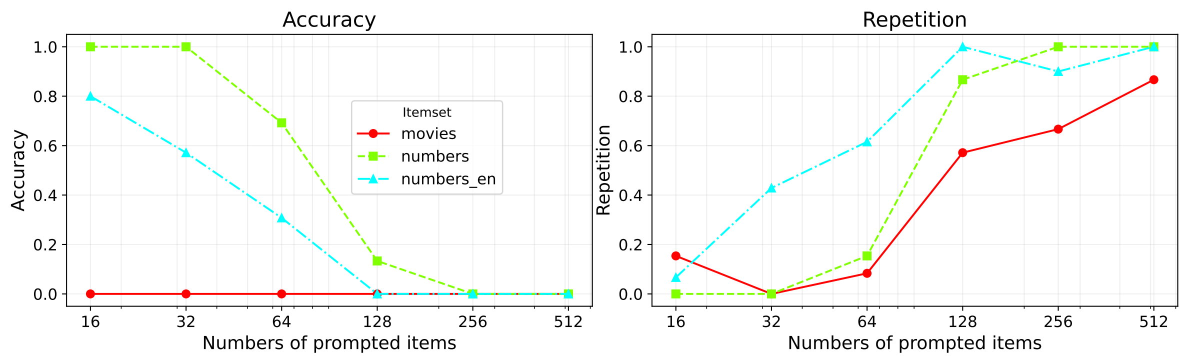
Concurrent work by Fu et al. (2025) introduces AbsenceBench, which also investigates the difficulty LLMs face with missing information. However, their task formulation is distinct: they provide the model with both an original document and a modified version, asking it to identify the removed elements. This frames the problem as a direct comparison or "diffing" task. In contrast, our missing item prediction task requires an inductive leap where the model must generate a novel item while verifying its absence from a single provided context. While both studies diagnose a failure in exhaustive reasoning, they approach it from complementary angles: AbsenceBench from explicit comparison and our work from constrained generation.

Recommendation with LLMs Our study is also related to LLM usage for collaborative filtering (Sileo et al., 2022), where users enumerate a list of items to communicate their tastes. LLMs can also be used in content-based recommendations, where users explicitly mention what they are looking for (Wu et al., 2023). Here, we do not address the fine-grained relevance of the recommendations (providing an item that users do not already know). Repetition is also related to the novelty metric in recommender systems evaluation (Vargas and Castells, 2011).

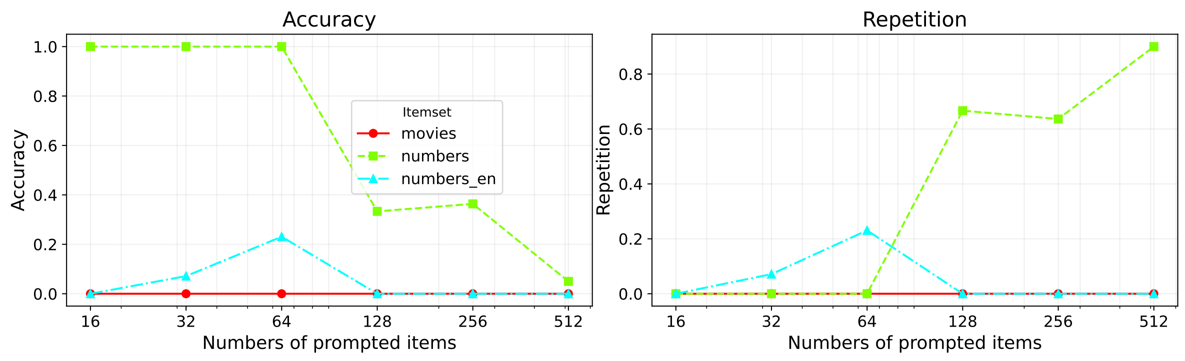
3 Missing item prediction

We formalize the task of missing item prediction as follows: Given a set X (randomly shuffled) of N elements, guess the element y that is missing in X. This is technically an induction task that can be underdetermined but we can construct relatively easy X, y pairs with easily identifiable itemsets $\mathcal{S}$ (numbers from 0 to 1024, letters, chemical elements...) and randomly removing one element y from $\mathcal{S}$ to get X. We can use two evaluation metrics:

Accuracy the rate at which a language model returns the expected missing element.



(a) Llama-3 zero-shot missing number prediction on multiple domains



(b) Llama-3 fine-tuned on missing number prediction

Figure 2: Llama-3-8B-Instruct Accuracy on various itemsets with increasing itemset sizes, without any fine-tuning (a) and after fine-tuning on the numbers itemset.

Repetition rate the rate at which a language model returns an element that is already in X.

Repetitions are always mistakes. For easily identifiable sets, ideal behavior is perfect accuracy and no repetition. But even in cases where the structure of $\mathcal{S}$ is under-determined, language models performing missing item prediction should not repeat elements from X.

To construct an example of the missing item prediction task, we select an itemset $\mathcal{S}$, select a random element y, and present a scrambled version of $X = \mathcal{S} \setminus \{y\}$ in a prompt explicitly asking the model to guess a missing element. We provide the following itemsets:

Movies We select a user from the MovieLens 1M dataset who watched more than 2048 movies.

Numbers Numbers in numerical form (1...1024). We exclude set extrema from the choice of y for numerical itemsets.

Numbers-english We use the same numbers but converted in English using the num2word library[3].

An example with the Numbers itemset of size 8 is QUESTION: *Find the missing element in 5, 7, 1, 3, 6, 8, 4.* ANSWER: 2.

[3]https://github.com/savoirfairelinux/num2words

4 Experiments

We use the same prompt template for all models:

```
Guess the missing item from this list: {X}. Directly answer with only one item. Item format should match the list format. Provide no explanation. Answer format: "{item}."
```

To construct this prompt template, we iterated on Llama-3-8B-Instruct with the numbers itemset validation data until we obtained a satisfactory output format. We normalize the outputs with punctuation removal and lowercasing to compute repetition rate and accuracy, and perform exact matches to compute accuracy and repetition rate.

We use powers of 2 starting from 16 as itemset sizes. This ensures that there are enough items to guess the itemset structure. We generate 200 train examples and 100/100 validation test examples per itemset size and itemset type.

4.1 Zero-shot evaluation

We evaluate off-the-shelf instruction-tuned language models via OpenRouter API. We evaluate Llama3-Instruct 8B and 70B, Gemini 1.5 Flash and Pro, GPT-4o, and Claude 3.5 Sonnet with the default hyperparameters.

Figure 1 shows the evolution of Accuracy and Repetition metrics with different itemsets sizes for numeric numbers and movies missing item prediction tasks. Most language models solve the missing number prediction task with relatively high accuracy with less than 128 items. Increasing model size improves accuracy, as Gemini Pro and Llama-3-70B outperform their smaller counterparts. However, the repetition rates shoot up and the accuracy decreases in all models after 256 items.

We cannot interpret the low accuracy of the movie item prediction tasks as a failure because the models can predict relevant movies that are not y. However, we can interpret the growing repetition rate as a failure, which can frustrate users who could expect better recommendations as they provide more examples, which limits the accuracy of conversational recommender systems that do not filter their output to prevent repetitions.

4.2 Fine-tuning

We now investigate whether fine-tuning can easily address this issue. We fine-tune Llama-3 Instruct 8B using Unsloth default configuration [4] (4bit quantization, LoRA (Dettmers et al., 2024) with dimension 16, 1 epoch with a learning rate of 2e-4). We fine-tune on 500 numeric items of size below 256 and evaluate on the test set in-domain and out-of-domain. Figure 2 shows that fine-tuning improves missing item prediction on in-domain data, but does not generalize to larger itemsets nor to different domains, which might indicate a limit of current attention architectures that may not be solved with data only.

4.3 Contrastive evaluation

We also evaluated the ability of LLama-3-8B-Instruct to tell whether an element is present or not in the list by randomly sampling either the missing element or a random element from a prompt.

```
{X}. Is "{item}" in the previous list? Provide no explanation, directly answer with only "Yes." or "No."
```

Figure 3 shows the evolution of accuracy with growing itemset sizes. Llama-3-8B-Instruct maintains 75% accuracy below 1024 items[5]. This shows that once the item is explicitly present in the query, the model is much better at identifying it. These results are lower than the Needle in a Haystack evaluation scores of Llama-3 (Zhang et al., 2024), which is due to the high similarity between items. This suggests that context-length stress testing is harder when many prompt elements are similar to each other, which makes existing (Kuratov et al., 2024) problem lengthening strategies too easy to circumvent.

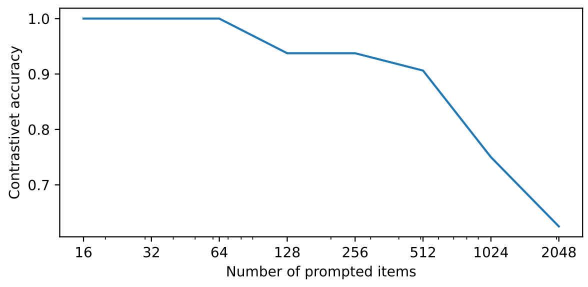


Figure 3: Zero-shot contrastive accuracy with Lllama-3-8B-Instruct on the Numbers itemset.

5 Analysis

To solve the missing item prediction problem, a model must generate a plausible candidate and verify its absence from the input list. Our contrastive experiments show that verifying a single item is much easier than generating a novel one. We hypothesize an *attention overflow*: as the list grows, the mechanism for verifying absence against all items simultaneously becomes overloaded. We visualize this with the contrastive task (Appendix A). When an item is present, attention focuses on it. When absent, attention diffuses across the input, as if searching for a match. This supports the idea that generating a novel item requires a distributed verification that fails with long lists, causing repetitions. This issue appears fundamental, affecting models with standard (8k) and extended context windows alike, pointing to an architectural bottleneck in handling exhaustive, non-local comparisons rather than a side-effect of context extension techniques.

[4] https://colab.research.google.com/drive/135ced7oHytdxu3N2DNe1Z0kqjyYIkDXp?usp=sharing

[5] All examples fit in the 8K context window of Llama 3.

6 Conclusion

We introduce a new missing item prediction dataset and show that repetitions occur in plausible recommendation tasks and synthetic list completion. Our findings highlight a limitation in the ability of current LLMs to check for exhaustivenessss. Our examples show that language models can repeat context elements when asked to produce novel content from a long list, with issues arising from as few as a hundred items. This contrasts with other long-context benchmarks, where failures appear at much larger scales. We attribute this to an *attention overflow*, where the model fails to compare a generated candidate against all input items simultaneously. While in-model solutions are challenging, a practical mitigation could involve offloading verification to an external code interpreter. Our dataset is publicly available to support future work on this problem.

Limitations

Our study has several limitations. The range of itemsets and models could be expanded for broader generalizability. Our fine-tuning experiments were limited, and more systematic prompt engineering might yield different results. We also lack a human baseline for comparison; however, the task's difficulty for unaided humans would strongly depend on factors such as time and available tools. The movie recommendation task simplifies real-world scenarios. While we provide initial evidence from attention visualizations (Appendix A), a more granular analysis of specific heads and layers is needed. Addressing these limitations would provide a more comprehensive understanding of the attention overflow phenomenon.

Acknowledgments

This work was supported by the French National Research Agency (ANR) under grant ANR-24-CE23-4637 (JCJC Adada). We also thank the anonymous reviewers for their insightful comments.

References

Cheng-Han Chiang and Hung-yi Lee. 2024. Over-reasoning and redundant calculation of large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–169, St. Julian's, Malta. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization. Main track.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Harvey Yiyun Fu, Aryan Shrivastava, Jared Moore, Peter West, Chenhao Tan, and Ari Holtzman. 2025. Absencebench: Language models can't tell what's missing. *arXiv preprint arXiv:2506.11440*.

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 14, pages 12848–12856.

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? *arXiv preprint arXiv:2404.06654*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yuri 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. *arXiv preprint arXiv:2406.10149*.

Valentin Lacombe, Valentin Quesnel, and Damien Sileo. 2025. Reasoning core: A scalable rl environment for llm symbolic reasoning. *arXiv preprint arXiv:2509.18083*.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Damien Sileo, Wout Vossen, and Robbe Raymaekers. 2022. Zero-shot recommendation as language modeling. In *Advances in Information Retrieval*, pages 223–230, Cham. Springer International Publishing.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Saúl Vargas and Pablo Castells. 2011. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 109–116.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2023. A survey on large language models for recommendation. *arXiv preprint arXiv:2305.19860*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

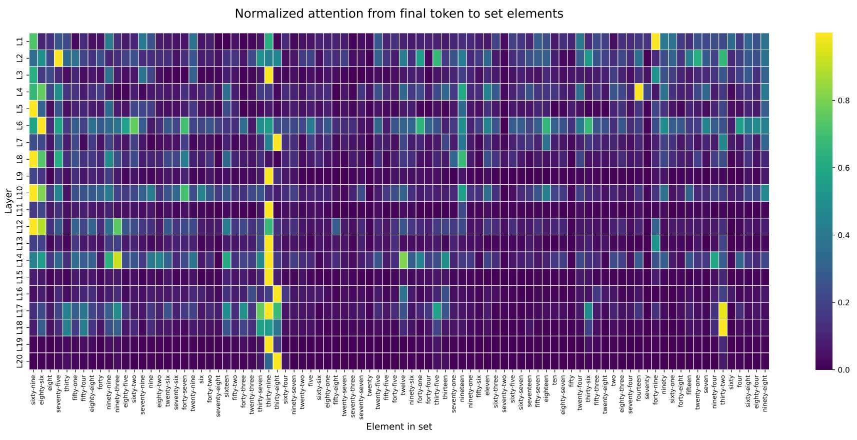
Peitian Zhang, Ninglu Shao, Zheng Liu, Shitao Xiao, Hongjin Qian, Qiwei Ye, and Zhicheng Dou. 2024. Extending llama-3's context ten-fold overnight. *arXiv preprint arXiv:2404.19553*.

A Attention Visualization

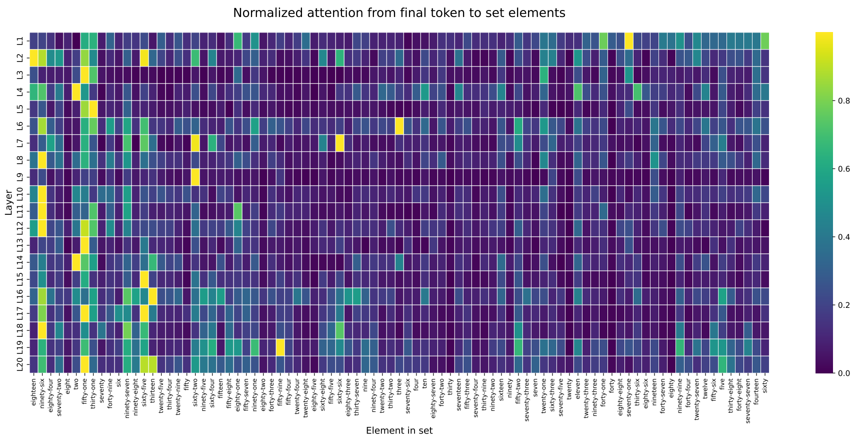
To better understand the *attention overflow* phenomenon, we visualize the heads-wise averaged attention from the input item tokens to the final answer tokens during the contrastive evaluation task with Qwen 3 4B (Yang et al., 2025), which we chose because of its small size. We change the prompt to force the attention scan to occur within a single token. The prompt in Section 4.3 has multiple tokens between the set and the queried token, so that the computation can be distributed over multiple steps, which makes it much harder to visualize the attention in an interpretable manner.

```
You will receive a set S followed by a number N. Directly output Yes if N is in S, and No if not. S: {set}. N: {word}
```

Figure 4 shows the attention patterns for lists of 100 and 99 numbers:



(a) Attention when the queried item (`thirty-nine`) is **present** in the input list. Attention is sharply focused on the correct item.



(b) Attention when the queried item (`sixty-one`) is **absent** from the input list. Attention is diffuse, spreading across multiple locations as the model searches for the item. This distributed search is more complex and prone to failure than focused retrieval.

Figure 4: Average attention from input list tokens to the answer token ("Yes" or "No") in the contrastive task. We exclude boundary elements from the visualization as they naturally attract more attention. We also exclude first and last layers from the visualization.

As shown, when the item is present (Figure 4a), the model's attention is highly localized to that specific item in context. However, when the item is absent (Figure 4b), attention becomes diffuse and spreads out over many items in the list. This suggests that the model is performing a broad, less efficient search across the context. We hypothesize that the generative task (finding the missing item) requires this diffuse, all-items-at-once verification, which becomes intractable as the list size increases, leading to overflow and repetition.