How Language Model Hallucinations Can Snowball

Muru Zhang[♡] **Ofir Press**[♡] **William Merrill**[♠] **Alisa Liu**[♡] **Noah A. Smith**[♡♣]
[♡]Paul G. Allen School of Computer Science and Engineering, University of Washington
[♠]New York University
[♣]Allen Institute for Artificial Intelligence
nanami17@cs.washington.edu

Abstract

A major risk of using language models in practical applications is their tendency to hallucinate incorrect statements. Hallucinations are often attributed to knowledge gaps in LMs, but we hypothesize that in some cases, when justifying previously generated hallucinations, LMs output false claims that they can separately *recognize* as incorrect. We construct three question-answering datasets where ChatGPT and GPT-4 often state an incorrect answer and offer an explanation with at least one incorrect claim. Crucially, we find that ChatGPT and GPT-4 can identify 67% and 87% of their own mistakes, respectively. We refer to this phenomenon as hallucination snowballing: an LM over-commits to early mistakes, leading to more mistakes that it otherwise would not make.[1]

1 Introduction

Language models are increasingly being deployed to interface with humans in open-ended information-seeking and problem-solving settings. Despite their diverse capabilities and extreme fluency, a major open challenge is that LMs still hallucinate by making up facts or citing sources that do not exist (Maynez et al., 2020; Liu et al., 2023, i.a.), often while sounding extremely plausible.

Hallucination is commonly attributed to knowledge gaps in LMs (Zheng et al., 2023), motivating mitigation strategies through retrieval over knowledge bases (Lewis et al., 2020; Shuster et al., 2021; Peng et al., 2023) But, do LMs *only* hallucinate when they do not "know" a fact? We present a setting where LMs often generate hallucinations that they immediately recognize as wrong when presented in isolation. Specifically, after an LM answers a question incorrectly, it usually justifies that answer by making incorrect assertions that it separately acknowledges as incorrect (Figure 1).

[1]Our data and code are available at: https://github.com/Nanami18/Snowballed_Hallucination

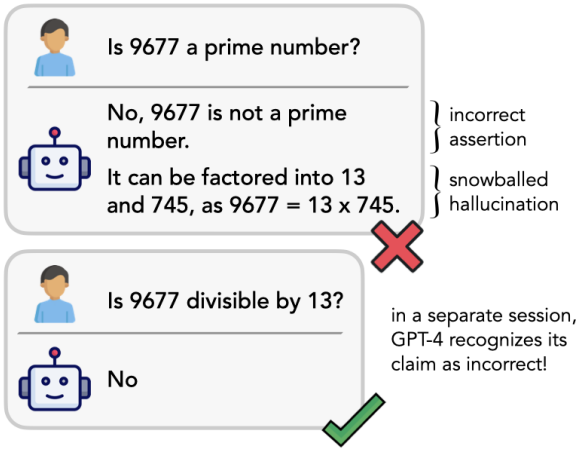


Figure 1: GPT-4 mistakenly claims that 9677 is not prime, followed by an incorrect explanation that $13 \times 745 = 9677$. We refer to this factorization as a *snowballed hallucination*, as GPT-4 appears to "know" that 13 is not a factor of 9677 when asked separately. Thus, hallucinations are not necessarily reflective of a model's lack of "knowledge," but may be induced by an earlier hallucination for consistency.

To study this behavior empirically, we automatically construct three question-answering (QA) datasets. These datasets span different domains: determining whether a number is prime, whether there is a U.S. senator satisfying two given constraints, and whether two cities are connected given a set of flights between cities. Empirically, we find that ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) commit to an answer within the first token (`Yes`/`No`) over 95% of the time; these answers are often incorrect, and then followed by an incorrect explanation. Yet, when presented with the incorrect explanation alone, we find that the LM is likely able to recognize it as incorrect.

We refer to this phenomenon as **hallucination snowballing**. We hypothesize that LMs produce snowballed hallucinations for consistency with earlier hallucinations (rather than due to a "knowledge gap" in the model), as they recognize the snow-

Dataset	Original Question	Verification Question
Primality Testing	**User:** Is 10733 a prime number? **GPT-4:** No... It can be factored into 3×3577.	**User:** Is 10733 divisible by 3? Answer with either Yes or No. **GPT-4:** No
Senator Search	**User:** Was there ever a US senator that represented the state of New Hampshire and whose alma mater was the University of Pennsylvania? **GPT-4:** Yes... His name was John P. Hale	**User:** Was John P. Hale's alma mater University of Pennsylvania? **GPT-4:** No... [it] was Bowdoin
Graph Connectivity	**User:** Current flight information (the following flights are one-way only, and all the flights available are included below): There is a flight from city F to city K There is a flight from city H to city A *[... 10 other rules cut for space ...]* Question: Is there a series of flights that goes from city B to city E? **GPT-4:** Yes... the route is as follows: ... City K to City G...	**User:** *[...flight information given in the context...]* Based on the above flight information, is City K to City G a valid flight? **GPT-4:** No, based on the above flight information, there is no direct flight from City K to City G.

Table 1: **Original Question**: An example from each dataset and GPT-4's (incorrect) response, with the snowballed hallucination underlined. **Verification Question**: The corresponding query used to probe whether GPT-4 recognizes its claim is incorrect, with its refutation of the hallucination underlined.

balled hallucination is incorrect when presented in isolation (i.e., in a separate interaction session).

While prompting strategies that encourage the LM to reason before stating an answer improve accuracy on the task, our work points to the broader issue that conditioning on faulty context leads LMs to produce extremely simple mistakes that they wouldn't otherwise make. Indeed, when prompting with "*Let's think step by step*" (Kojima et al., 2023), snowballed hallucinations still occur in 95% of cases where the model fails to answer correctly. We observe that sometimes even when "*Let's think step by step*" does lead to the right answer, it uses invalid reasoning chains.

In this paper, we demonstrate the phenomenon of hallucination snowballing by leveraging recent LMs' tendency to state and justify their answers. Rather than over-committing to its previously generated context, we believe that LMs should acknowledge their initial mistake, and then revise their answer. We have indeed observed GPT-4 doing this in a limited number of cases; amplifying this behavior would be beneficial, as well as developing new methods in which LMs can backtrack.

2 Why do we expect hallucination snowballing?

In this section, we explain why we hypothesize that LMs are susceptible to hallucination snowballing. We predict that snowballing will occur on questions with two key properties:

1. **Initial committal:** The prompt leads the LM to first state an answer (*before* outputting the explanation). This applies to many yes/no questions.
2. **Inherently sequential:** Transformers cannot find the answer within one timestep because of their limited reasoning abilities within one timestep.

We now discuss how these properties may lead to snowballed hallucination.

Initial committal. In English and many other languages, speakers often say the final Yes/No answers to questions before explaining their answer. We therefore hypothesize that LMs and especially instruction-tuned LMs (Wei et al., 2021; Sanh et al., 2021; Ouyang et al., 2022; Wang et al., 2022) will reflect this answer format where the answer comes before the explanation. Indeed, on our datasets (presented in §3.1), we observe that GPT-4 and ChatGPT immediately commit to an answer to the question: the first token is `Yes` or `No` 95.67% and 98.40% of the time for GPT-4 and ChatGPT respectively. In the remaining cases, the model often commits to an answer within the first few tokens of the response (e.g., *"There is no record of a U.S. Senator..."*). Crucially, once the LM generates `Yes` or `No`, that token remains in the context, and coherence would require commitment to that choice through the subsequent justification. Thus, the model pro-

duces an answer to a complex question in a *single* timestep, and it then continues by generating an explanation for that answer, which inevitably will be incorrect.

Inherently sequential. Furthermore, transformers cannot solve inherently sequential reasoning problems like primality testing or graph connectivity within a single timestep,[2] as documented in recent theoretical results (Merrill and Sabharwal, 2023).[3] Our graph connectivity and primality datasets are concrete instantiations of these problems. Because the transformer must use one step to answer a question that requires multiple timesteps to answer correctly, it will necessarily sometimes commit to an incorrect answer. We hypothesize that this leads the LM to hallucinate supporting incorrect facts that it otherwise would not generate.

3 Experiments

We design three QA datasets with the properties described in §2 to probe hallucination snowballing, and evaluate ChatGPT and GPT-4. We first check whether the LM returns the correct answer to the given question, and we show that when the model returns the wrong answer, it frequently provides an incorrect explanation for that wrong answer. We automatically extract the incorrect claim in the explanation and ask the same LM to check whether its claim is correct. See Table 1 for a representative example from each dataset.

3.1 Datasets

We design three QA datasets, each containing 500 yes/no questions that we expect are not answerable by transformers in one timestep. To aid evaluation, the questions are designed so that an incorrect answer would be justified with easily verifiable claims.

For each dataset, we fix one specific label for all examples, so that if the model chooses the incorrect answer (e.g., that 9677 is not prime), it would produce a specific claim to support it (e.g., an incorrect factorization). This enables us to systematically examine model-written justifications for incorrect answers.

Primality testing For this dataset, we query the primality of 500 randomly chosen primes between 1,000 and 20,000; the correct answer is always `Yes`. When the model answers incorrectly, we expect it to justify its answer with an incorrect factorization.

Senator search This dataset consists of 500 questions of the form "*Was there ever a US senator that represented the state of x and whose alma mater was y?*" where x is a U.S. state and y is a U.S. college. For these questions, the correct answer is always `No`. When the model answers incorrectly, we expect it to falsely claim that a particular senator both represented x and attended y.

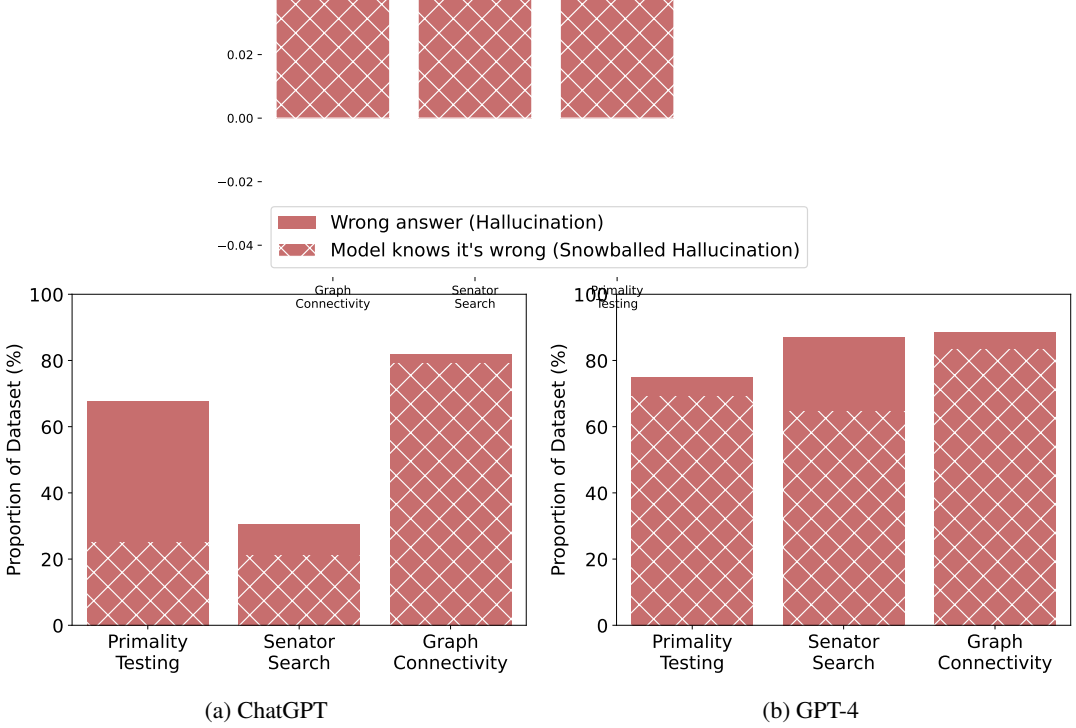
To create the dataset we consider all U.S. states and a manually constructed list of twelve popular U.S. colleges (see §A for the full list); for each possible pair, we generate a question following the template, and manually remove pairs where the answer is `Yes`.

Graph connectivity For each of the 500 questions in this dataset, we present 12 flights among 14 cities, and ask if there is a sequence of flights from a particular city to another. The problem always corresponds to the same underlying directed graph structure (see §A.1), where flights are edges and cities are nodes. For each instance in the dataset, we randomly assign letters from the English alphabet to name the nodes. To formulate the query, we sample a source city s and destination city t in different subgraphs, with the additional constraint that s corresponds to a source node, and t a leaf node, so that 1-step heuristics cannot be used to solve the problem.

We formulate the problem as a flight-finding question in natural language so that it sounds more natural: in the prompt, we list the twelve flights ("*There is a flight from city F to city K; there is a flight from city G to city N, ...*"), followed by the question *"Is there a series of flights... from s to t?"*. Note the correct answer is always `No`. When the model answers incorrectly, we expect it to justify its answer with a flight that does not exist.

[2] Technically, this holds only for inputs above a certain hardness level, i.e., the size of the prime number for primality testing, or the size of the graph for graph connectivity.

[3] Merrill and Sabharwal (2023) show that, with a single generation step, bounded-precision transformers cannot solve any problem outside the complexity class TC^0, which corresponds to a highly parallelizable subclass of both L (log-space) and P (polynomial-time). Graph connectivity is an L-complete problem, which means it cannot be in TC^0 unless $\mathsf{TC}^0 = \mathsf{L}$, i.e., all of L can be parallelized to a surprisingly high degree. Primality testing was shown to be in P (Agrawal et al., 2004) but cannot be in TC^0 unless it is also in L; i.e., any n can be factored with $O(\log \log n)$ bits of overhead. In summary, unless standard complexity-theoretic conjectures are false, graph connectivity and primality testing are outside TC^0 and thus are too inherentially sequential for transformers to solve in a single generation (cf. Merrill and Sabharwal, 2023).



(a) ChatGPT (b) GPT-4

Figure 2: Percentage of hallucination and percentage of snowballed hallucination (both calculated with respect to the entire dataset) for ChatGPT and GPT-4. The precise numbers for this plot are available in Table 6 and Table 7 in the Appendix.

3.2 Inference Setup

Language models. We run all experiments on ChatGPT (`gpt-3.5-turbo`) and GPT-4 with greedy decoding.

Our experiments are *zero-shot* (i.e., we do not show the model any example QA pairs in the prompt). We focus on the model behavior under the direct prompt (see §A for full examples), which is the most common way users interact with LMs. See §4 for experiments with the zero-shot chain-of-thought style prompting method.

For each dataset, we perform a two-stage evaluation. First, we evaluate the model's accuracy (i.e., how many of the questions it answers correctly). When either models is *incorrect*, empirically it *always* generates a justification. In the second stage, we assess whether the model can identify the incorrect step in the explanation.

For a given question, we evaluate the model's response by examining whether the output begins with either `Yes` or `No`. In cases where the response does not fall into these categories, we manually determine the answer conveyed by the model.

3.3 LM Recognition of Snowballed Hallucinations

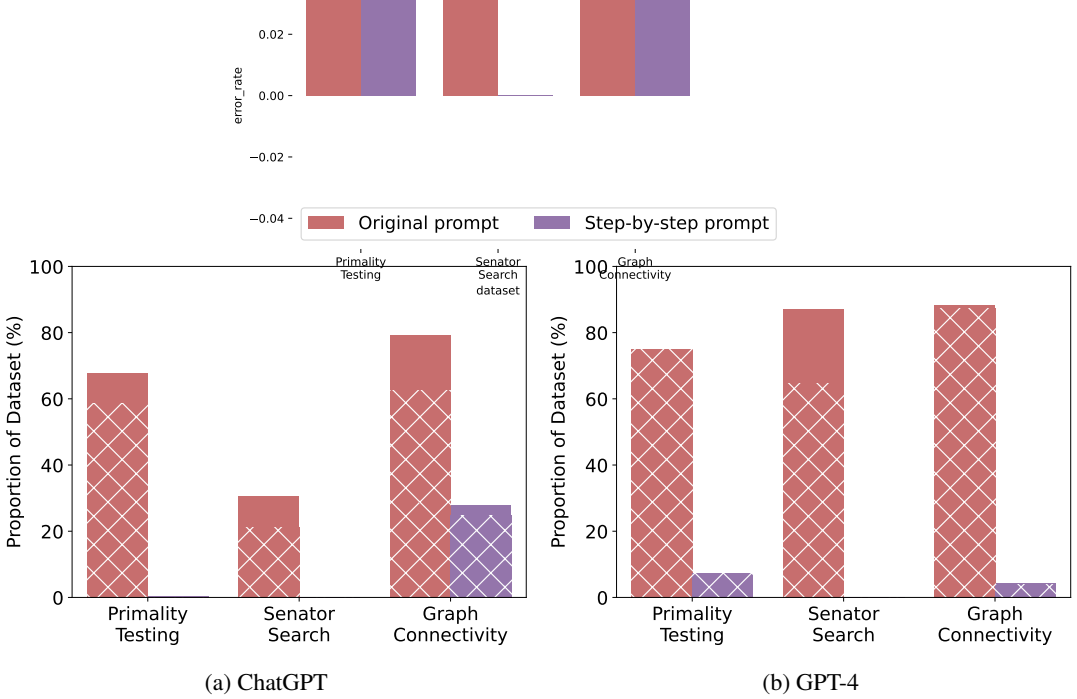
We probe whether LMs recognize their snowballed hallucinations by verifying the model's incorrect claims in the output against the model itself. Note that our recognition procedure relies on heuristics gained from manual examination of the model output, and these heuristics might not work on other models (e.g., a different model might not provide factors when supporting the claim that a number is not prime).

Graph Connectivity For each sample where the model thinks there is a series of connecting flights (where answer starts with `Yes`), we manually extract the list of flights from the model's output and identify the invalid or discontinuous flights.

We then, in a new session, ask the model to verify whether the extracted flights are valid based on the flight information, and if consecutive flights are indeed connected. We manually assess the verification output to check if the model correctly detects the error. See Appendix Table 3 for how we prompt the model and an example of successful verification.

Primality Testing For each sample where the model answers that the number is not prime, we extract the factors the model uses to justify it. The extraction is done by putting the output in the context and asking *"What are the factors proposed in the above text? List them out."* We use ChatGPT for extraction with one-shot demonstration (for its fast inference speed); we manually checked 30 examples and found that it can always extract the correct factors.

We then, in a new session, ask the model to verify each extracted factor individually. See Appendix Table 4 for an example of successful verification.



(a) ChatGPT (b) GPT-4

Figure 3: Error rate and snowballed hallucination rate (hatch pattern) for ChatGPT and GPT-4, when using the original prompt versus *"Let's think step by step"*. See Appendix Table 8 and Table 9 for the exact numbers.

Senator Search For each sample where the model thinks there is such senator, we extract the name of the senator the model uses to justify the existence, by putting the output in the context and asking *"What is the senator mentioned in the above text? Just give the name"*. Again, we use ChatGPT and manually observed perfect extraction on 30 examples.

We then, in a new session, ask the model if that senator's alma mater is the college in the question and has represented the state in the question. See Appendix Table 5 for an example of successful detection.

3.4 Results

Question-answering accuracy Figure 2 shows that both ChatGPT and GPT-4 experience very low accuracy across the board. With the exception of ChatGPT on the **Senator Search** dataset, all models achieve less than 50% accuracy.(See Appendix Table 6 for a breakdown of the error rate by dataset.) We observe that GPT-4 performs worse than ChatGPT across all datasets despite popularly being considered superior to ChatGPT (OpenAI, 2023). While ChatGPT has an average accuracy of 39.87%, GPT-4 has only 16.6%.

Hallucination detection Here, we check whether the model can identify that the incorrect claim is wrong when it is presented alone. As shown in Figure 2, ChatGPT detects 67.37% of incorrect claims in explanations (i.e., snowballed hallucinations), and GPT-4 detects 87.03%. Notice that when the model fails the verification (an example in Appendix Table 12), we do not consider it a snowballed hallucination.

Overall, we find that ChatGPT and GPT-4 are both extremely susceptible to hallucination snowballing, leading to extremely simple mistakes.

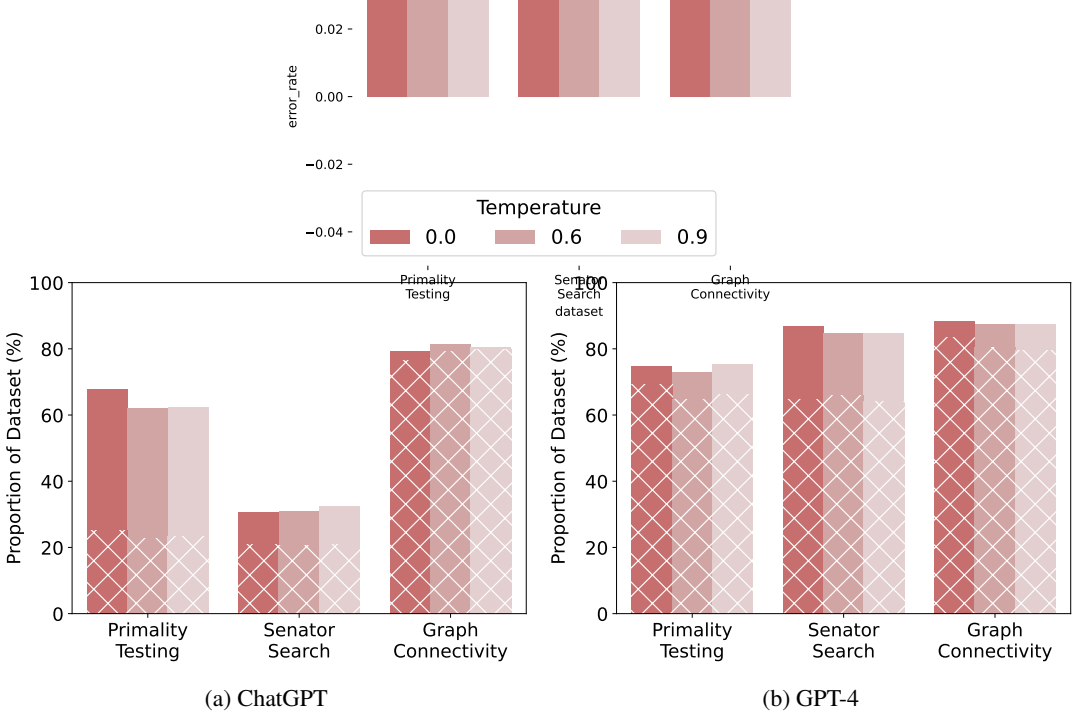
4 Can we prevent snowball hallucinations?

We hypothesize that hallucination snowballing occurs because LMs are trained to model continuations consistent with their current context (the given prompt and prior outputs). Although a fix to the fundamental problem might require more than just inference-time modification, in this section we study the effectiveness of two inference strategies in alleviating hallucination snowballing: prompting (§4.1) and decoding or training methods (§4.2).

4.1 Engineering Better Prompts

In this section, we examine the effectiveness of better prompts on preventing snowballed hallucination by using a different zero-shot prompt that encourages the model to generate the reasoning chain before the answer. Since the outputs generated under these prompts are less structured, we manually inspect them to determine correctness and the presence of snowballed hallucinations.

For each task, we append *"Let's think step-by-step"* at the end of the original question (shown in Table 1). As shown in Figure 3, the model can solve the **Senator Search** task perfectly, achieve $\leq$10% error rate on **Primality Testing**, and $\leq$30% on **Graph Connectivity**. Despite the large improve-



(a) ChatGPT (b) GPT-4

Figure 4: Error rate and snowballed hallucination rate (hatch pattern) from ChatGPT and GPT-4, when using different values for temperature at decoding-time. See Appendix Table 10 and Table 11 for the exact numbers.

ment in accuracy, we identify a potential issue: the model sometimes hallucinate while outputting the reasoning chain, which causes snowballed hallucination in future steps. For example, in the below output,

> *[....previous steps omitted]*
>
> *Step 3: From city E, we have three options: a flight to city N, a flight to city B, or a flight to city C.*
>
> *Step 4: The only option that could potentially lead us to city M is the flight from city E to city C.*
>
> *[....rest of the output omitted]*

ChatGPT incorrectly states that there are three options in the step 3 (there are only two), inducing the snowballed hallucination *"or a flight to city C"* (ChatGPT can verify that E → C is not a valid flight in a separate session). As shown in Figure 3, GPT-4 still has a high overall snowballed hallucination rate at 94.90% averaged across tasks, and ChatGPT also obtains a similarly high snowballed hallucination rate.

Finally, while our experiments have focused on simple multi-step problems that are suitable for breaking down step-by-step, we hypothesize that hallucination snowballing appears in open-ended text generation more broadly, where one mistake in the generation triggers more (Arora et al., 2022). In these cases, better prompting would neither be able to anticipate nor fix these mistakes.

4.2 Algorithmic Corrections

Increasing the temperature During decoding, the temperature t controls the sharpness of the output distribution, with higher t spreading probability mass away from the model's most likely prediction for each next word. Our experiments in §3 used greedy decoding, which is equivalent to $t = 0$. At $t = 0.6$ and $t = 0.9$, both error rates and snowballed hallucination rate remain similarly high, in both GPT-4 and ChatGPT (Figure 4).

Top-k and nucleus sampling Using sampling methods such as top-k sampling or nucleus sampling (Holtzman et al., 2020) would not help since they only *narrow* the range of tokens to be considered, and thus can only *increase* the probability that the model will immediately commit to an answer.

Beam search The argument for hallucination snowballs in §2 relies on the fact that, once a model generates some tokens committing to an answer, they remain in the context and influence later generations. One potential way around this is *beam search*, i.e., maintaining a beam of high-probability sequences at each timestep rather than a single sequence. In principle, if some sequences in the beam after the initial token do not commit to an answer (or commit to the right answer), their continuations may eventually have higher probability than those that initially commit incorrectly and later produce incorrect reasoning as a result. If so, beam search would solve the snowball hallucination problem. Unfortunately, we cannot test the effect of beam search on hallucination snowballs because the Ope-

nAI API does not support beam search.

Learning strategies A more general way to further reduce snowballing might be to change aspects of the pretraining or instruction tuning phases. In particular, a greater emphasis on having the model produce a reasoning chain before generating an answer could be a good way to accommodate its computational limitations and avoid committing to wrong answers that force hallucinations.

In addition, we hypothesize that finetuning on data with backtracking might improve a model's performance on the tasks we present. This could be accomplished by, for example, giving a question, followed by a wrong solution, and then issuing a phrase like *"Sorry, that was incorrect"* before giving the correct solution. This solution is related to the *"Review your previous answer and find problems with your answer."* prompt from Kim et al. (2023).

5 Related Work

Hallucinations Hallucination in text generation is a well-studied problem (Rohrbach et al., 2018; Maynez et al., 2020; Raunak et al., 2021, i.a.) that has recently become more prominent due to ChatGPT's tendency to produce plausible-sounding falsehoods. Hallucinations are often attributed to knowledge gaps in LMs (Zheng et al., 2023), and several works have shown the promise of using retrieval over knowledge bases to mitigate them (Lewis et al., 2020; Shuster et al., 2021; Peng et al., 2023). Our work demonstrates hallucination can be induced from context, thus motivating further mitigation techniques.

Hallucination snowballing is likely the result of *exposure bias*: LMs were only exposed to gold history during training, but during inference, conditions on possibly erroneous previous predictions. Prior work linked this to compounding hallucinations in machine translation (Wang and Sennrich, 2020) and open-ended text generation (Arora et al., 2022). We go beyond demonstrating error propagation by showing that the propagated errors (which we call snowballed hallucinations) are recognized by the LM itself.

Our observations are related to previous findings that LMs hallucinate when given questions that contain false presuppositions (e.g., "*Which linguist invented the lightbulb?*"; Kim et al., 2021, 2022) or that are otherwise misleading (e.g., "*Who really caused 9/11?*"; Lin et al., 2022), in that faulty context misguides the LM. However, our work differs in that our questions are not intentionally misleading, showing that this failure mode may be triggered even on innocent information-seeking queries to the LM.

LM (in)consistency Our work adds to a growing body of work demonstrating the extent to which LMs are inconsistent across different prompts on the same issue. For instance, allowing an LM to generate intermediate steps (Nye et al., 2021; Wei et al., 2022; Press et al., 2022) enables it to reach a different answer than it otherwise would. Other work has shown that simply prepending "*Professor Smith was given the following instructions*" to a prompt can improve performance, despite providing no valuable information about the problem itself (Lin et al., 2022).

6 Conclusion

We define the phenomenon of hallucination snowballing and demonstrate its prevalence in generations from state-of-the-art models, leading to hallucinations on simple facts that wouldn't otherwise occur. Our findings point to the risk of training language models that prioritize fluency and coherence indiscriminatively at the expense of factuality, and we encourage future work to study remedial actions at all levels of model development.

Limitations

We focus on hallucination snowballing in the context of question answering in English, and we do not explore it on other tasks, such as summarization or code generation.

In addition, we only conduct experiments on two proprietary models, namely ChatGPT and GPT-4, due to their state-of-the-art performance on many benchmarks (OpenAI, 2023). Due to the limitations of the APIs for these models, we do not have access to the probability distributions they output and do not have the ability to finetune them. This restricts our ability to explore potential mitigation strategies. Having access to the output distributions would allow us to investigate mitigating the snowballing hallucination issue using alternative sampling methods such as beam search. Having the ability to finetune the model would allow us to explore whether instruction tuning with different annotations could lead to better handling of the questions we use to instigate hallucination snowballing.

Acknowledgements

We thank Sofia Serrano, Yizhong Wang, Yanai Elazar, Michael Hu and Richard Yuanzhe Pang for their valuable feedback and fruitful discussions. While writing this paper, Ofir Press was a visitor at New York University's Center for Data Science, hosted by Kyunghyun Cho.

References

Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. 2004. Primes is in p. *Annals of Mathematics*, 160:781–793. Godel Prize, Fulkerson Prize.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Cheung. 2022. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 700–710, Dublin, Ireland. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks.

Najoung Kim, Phu Mon Htut, Sam Bowman, and Jackson Petty. 2022. (qa)2: Question answering with questionable assumptions. *ArXiv*, abs/2212.10003.

Najoung Kim, Ellie Pavlick, Burcu Karagol Ayan, and Deepak Ramachandran. 2021. Which linguist invented the lightbulb? presupposition verification for question-answering. In *Annual Meeting of the Association for Computational Linguistics*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

William Merrill and Ashish Sabharwal. 2023. The parallelism tradeoff: Limitations of log-precision transformers.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models.

Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt. 2021. The curious case of hallucinations in neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1172–1183, Online. Association for Computational Linguistics.

Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. 2018. Object hallucination in image captioning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4035–4045, Brussels, Belgium. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin

Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2021. Multitask prompted training enables zero-shot task generalization.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Why does chatgpt fall short in answering questions faithfully?

A Dataset Details

A.1 Graph Connectivity

In this dataset, the list of flights can be represented by a directed graph. We generated the flight information to ensure all the graphs share a specific connection pattern, with the node names randomly chosen among the 26 letters in the English alphabet. For an illustration of the underlying graph structure, see Figure 5.

A.2 Senator search

The twelve colleges used in the datasets are: MIT, University of Chicago, Johns Hopkins University, California Institute of Technology, Duke University, Northwestern University, Dartmouth College, Brown University, Vanderbilt University, Rice University, University of Washington. We constructed this list by taking a list of top universities in the U.S. and excluding from it universities which also appeared on The U.S. News & World Report's list of Top 10 Colleges for Members of Congress.

B Additional Results

We provide the detail breakdown of the question-answering accuracy in Table 6 and the hallucination detection accuracy in Table 7.

Graph Connectivity: Current flight information (the following flights are one-way only, and all the flights available are included below):

There is a flight from city F to city K
There is a flight from city H to city A
There is a flight from city B to city F
There is a flight from city N to city H
There is a flight from city B to city M
There is a flight from city N to city G
There is a flight from city M to city J
There is a flight from city G to city L
There is a flight from city H to city E
There is a flight from city G to city C
There is a flight from city M to city I
There is a flight from city F to city D

Question: Is there a series of flights that goes from city B to city E?

Table 2: Input example for the Graph Connectivity task. Figure 5 describes the underlying graph structure for this example. We use the same graph structure in all examples, and vary the source and target cities and the labels of the nodes. The cities are literally called "City A", "City B", and so on.

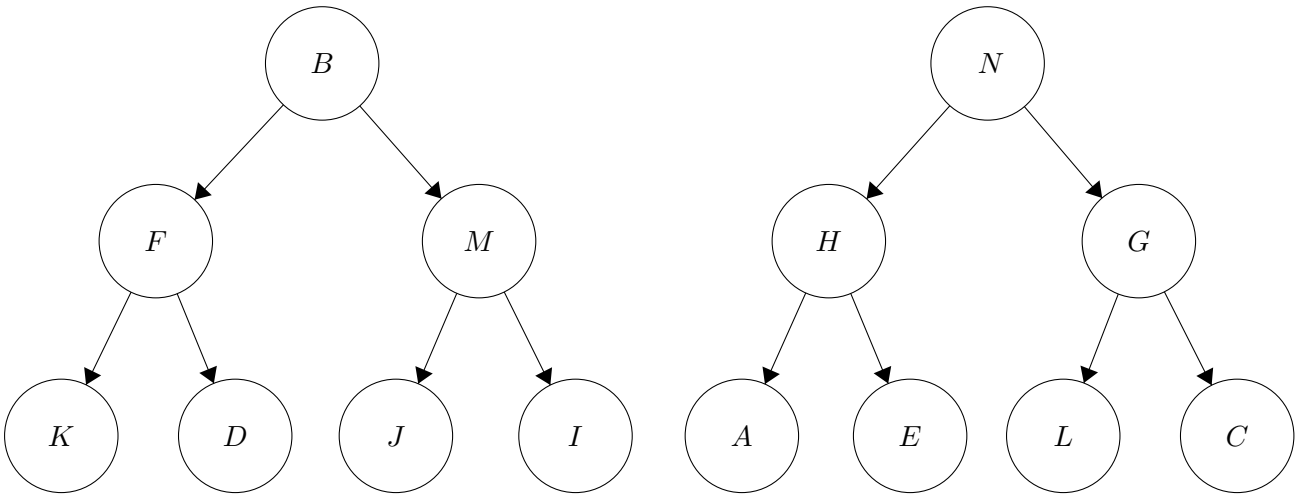


Figure 5: The underlying graph structure corresponds to example in Table 2.

Input: Current flight information (the following flights are one-way only, and all the flights available are included below): There is a flight from city G to city B There is a flight from city H to city K There is a flight from city L to city M There is a flight from city F to city H There is a flight from city G to city J There is a flight from city B to city I There is a flight from city L to city A There is a flight from city H to city N There is a flight from city B to city D There is a flight from city J to city C Question: Is there a series of flights that goes from city F to city I?
Output: Yes, there is a series of flights that goes from city F to city I. The route is as follows: 1. City F to City H 2. City H to City K 3. City K to City G 4. City G to City B 5. City B to City I
Verification Question: Current flight information (the following flights are one-way only, and all the flights available are included below): There is a flight from city G to city B There is a flight from city H to city K *[... 10 other rules cut for space ...]* Based on the above flight information, is City K to City G a valid flight?
Verification Answer: No, based on the above flight information, there is no direct flight from City K to City G.

Table 3: Graph Connectivity dataset verification example, with GPT-4.

Input: Is 9791 a prime number?
Output: No, 9791 is not a prime number. It can be factored into 13×751.
Verification Q1: Is 9791 divisible by 13? Answer with either Yes or No.
Verification A1: No
Verification Q2: Is 9791 divisible by 751? Answer with either Yes or No.
Verification A2: No

Table 4: Primality Testing dataset verification example, with GPT-4.

Input: Was there ever a US senator that represented the state of New Hampshire and whose alma mater was University of Pennsylvania?
Output: Yes, there was a U.S. Senator who represented the state of New Hampshire and whose alma mater was the University of Pennsylvania. His name is John P. Hale, who served as a Senator from 1847 to 1853 and again from 1855 to 1865. He graduated from Bowdoin College in 1827 and later attended the University of Pennsylvania Law School.
Verification Q1: Has John P. Hale represented the state of New Hampshire as US senator?
Verification A1: Yes, John P. Hale represented the state of New Hampshire as a U.S. Senator. He served in the Senate from March 4, 1847, to March 3, 1853, and again from March 4, 1855, to March 3, 1865.
Verification Q2: Was John P. Hale's alma mater University of Pennsylvania?
Verification A2: No, John P. Hale's alma mater was Bowdoin College in Brunswick, Maine.

Table 5: Senator Alma Mater dataset verification example, with GPT-4.

	Task			Average
Model	Graph Connectivity	Primality Testing	Senator Search	
ChatGPT	410/500 (82.0%)	339/500 (67.8%)	153/500 (30.6%)	60.13%
GPT-4	442/500 (88.4%)	374/500 (74.8%)	435/500 (87.0%)	83.40%

Table 6: Number of **mistakes** out of the number of samples, the percentage here is the error rate.

	Task			Average
Model	Graph Connectivity	Primality Testing	Senator Search	
ChatGPT	396/410 (96.6%)	125/339 (36.9%)	98/153 (68.6%)	67.37%
GPT-4	417/442 (94.3%)	346/374 (92.5%)	323/435 (74.3%)	87.03%

Table 7: Number of snowballed hallucination out of number of hallucination generated in the original output.

	Task			Average
Model	Graph Connectivity	Primality Testing	Senator Search	
ChatGPT	139/500 (27.8%)	2/500 (0.4%)	0/500 (0.0%)	9.40%
GPT-4	21/500 (4.2%)	37/500 (7.4%)	0/500 (0.0%)	3.87%

Table 8: Number of **mistakes** out of the number of samples, the percentage here is the error rate, using "Let's think step by step" prompt.

	Task			Average
Model	Graph Connectivity	Primality Testing	Senator Search	
ChatGPT	123/139 (88.5%)	0/2 (0%)	0/0 (N/A)	44.25%
GPT-4	20/21 (95.2%)	35/37 (94.6%)	0/0 (N/A)	94.90%

Table 9: Number of snowballed hallucination out of number of hallucination generated in the original output, using "Let's think step by step" prompt.

Model	Graph	Prime	Senator	Average
ChatGPT ($t = 0.0$)	410/500 (82.0%)	339/500 (67.8%)	153/500 (30.6%)	60.13%
ChatGPT ($t = 0.6$)	407/500 (81.4%)	310/500 (63.2%)	155/500 (31.0%)	58.53%
ChatGPT ($t = 0.9$)	403/500 (80.6%)	312/500 (62.4%)	163/500 (32.6%)	58.53%
GPT-4 ($t = 0.0$)	442/500 (88.4%)	374/500 (74.8%)	435/500 (87.0%)	83.40%
GPT-4 ($t = 0.6$)	438/500 (87.6%)	365/500 (75.4%)	423/500 (84.6%)	82.53%
GPT-4 ($t = 0.9$)	437/500 (87.4%)	377/500 (73.0%)	423/500 (84.6%)	81.67%

Table 10: Number of **mistakes** out of the number of samples, the percentage here is the error rate, with different temperature setups

Model	Graph	Prime	Senator	Average
ChatGPT ($t = 0.0$)	396/410 (96.6%)	125/339 (36.9%)	98/153 (68.6%)	67.37%
ChatGPT ($t = 0.6$)	396/407 (97.3%)	113/310 (36.5%)	103/155 (66.5%)	66.77%
ChatGPT ($t = 0.9$)	399/402 (99.3)%	116/312 (37.2%)	104/163 (63.8%)	66.77%
GPT-4 ($t = 0.0$)	417/442 (94.3%)	346/374 (92.5%)	323/435 (74.3%)	87.03%
GPT-4 ($t = 0.6$)	402/438 (91.8%)	324/365 (88.8%)	329/423 (77.8%)	86.13%
GPT-4 ($t = 0.9$)	398/437 (91.1%)	331/377 (87.8%)	320/423 (75.7%)	84.87%

Table 11: Number of **mistakes** out of the number of samples, the percentage here is the error rate, with different temperature setups

Input: Was there ever a US senator that represented the state of North Carolina and whose alma mater was Dartmouth College?
Output: Yes, there was a U.S. Senator who represented the state of North Carolina and whose alma mater was Dartmouth College. His name was Willis Smith, and he served as a Senator from 1950 to 1953.
Verification Q1: Has Willis Smith represented the state of North Carolina as a US senator?
Verification A1: Yes, Willis Smith represented the state of North Carolina as a U.S. Senator. He served from 1950 until his death in 1953.
Verification Q2: Was Willis Smith's alma mater Dartmouth College?
Verification A2: Yes, Willis Smith's alma mater was Dartmouth College. He graduated from there in 1917.

Table 12: An example of GPT-4 failing the verification test for Senator Search task. Willis Smith was a senator from North Carolina, but he has never attended Dartmouth College.