How Language Model Hallucinations Can Snowball

Muru Zhang [1] **Ofir Press** [2 3] **William Merrill** [4] **Alisa Liu** [1] **Noah A. Smith** [1 5]

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 show that LMs sometimes produce hallucinations that they can separately *recognize* as incorrect. To do this, we construct three question-answering datasets where LMs often state an incorrect answer which is followed by an explanation with at least one incorrect claim. Crucially, we find that GPT-3.5, GPT-4, and LLaMA2-70B-chat can identify 67%, 87%, and 94% of these incorrect claims, respectively. We show that this phenomenon doesn't disappear under higher temperatures sampling, beam search, and zero-shot chain-of-thought prompting. These findings reveal that LM hallucinations can snowball: early mistakes by an LM can lead to more mistakes that otherwise would not be made.[1]

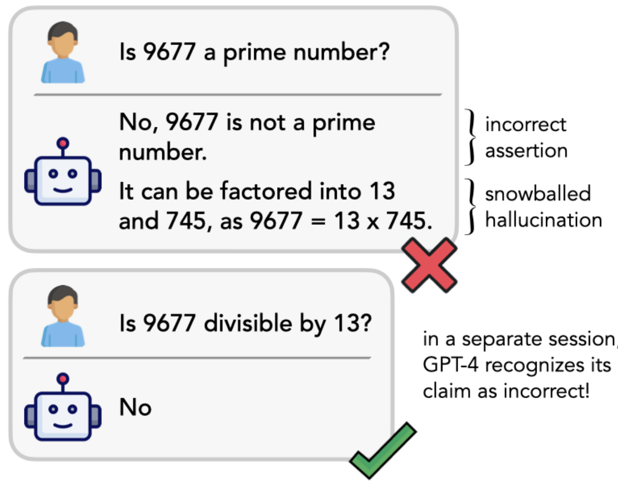


Figure 1. GPT-4 mistakenly claims that 9677 is not prime, followed by an incorrect explanation that $13 \times 745 = 9677$. We refer to this factorization as a *snowballed hallucination*, as GPT-4 appears to "know" that 13 is not a factor of 9677 when asked separately. Thus, hallucinations are not necessarily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

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), 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).

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 GPT-3.5 (OpenAI, 2022) and GPT-4 (OpenAI, 2023) commit to an answer (`Yes`/`No`) with the first token over 95% of the time, and LLaMA-2-70B-chat often commit to an answer within the first few tokens; these answers are often incorrect, and then followed by an incorrect explanation. Increasing the temperature when sampling from the model or using beam search doesn't alleviate this behavior. Yet, when presented with the incorrect explanation alone, we

[1]Paul G. Allen School of Computer Science and Engineering, University of Washington [2]Princeton University [3]Princeton Language and Intelligence [4]Center for Data Science, New York University [5]Allen Institute for AI. Correspondence to: Muru Zhang <nanami17@cs.washington.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

[1]Data and code can be found at `https://github.com/Nanami18/Snowballed_Hallucination`

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.

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 snowballed hallucination is incorrect when presented in isolation (i.e., in a separate interaction session). We observe this phenomenon both in empirical testing of the leading closed models, GPT-3.5 and GPT-4, and in the leading open model, LLaMA2-70b-chat.

While prompting strategies that encourage the LM to reason before stating an answer improve accuracy on the task, effective prompts might not transfer across different models and on different datasets. We found that prompting GPT-3.5 and GPT-4 with "*Let's think step by step*" (Kojima et al., 2023) substantially lowers the hallucination rate on the simpler tasks we present in §3.1, but same prompt doesn't lead to improvement on LLaMA-2-70B-chat. Furthermore, the prompt does not remedy the issue on the harder datasets we introduce in §4.2.

While we demonstrate the issue of hallucination snowballing by leveraging recent LMs' tendency to state and then justify their answers, our results point to a broader issue of LM behavior that they might prioritize consistency over factuality. Rather than 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 that could allow LMs to backtrack and address previous incorrect commitments.

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 a single step.

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

Initial committal. Speakers often state 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 GPT-3.5 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 GPT-3.5 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 produces 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 & 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 GPT-3.5, GPT-4, and LLaMA-2-70B-chat. 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.

For evaluating GPT-3.5 and GPT-4, we accessed the model through the official OpenAI API (equivalent to `https://platform.openai.com/playground`) between April and May 2023. However, the models being deployed were constantly updated, therefore rerunning the experiment might result in different numbers. For reference, we also included results rerun on March 2024 in the appendix. Overall, the new numbers still support observations and conclusions in the paper.

[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 & 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., we can test if n is prime 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 step (cf. Merrill & Sabharwal, 2023).

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.

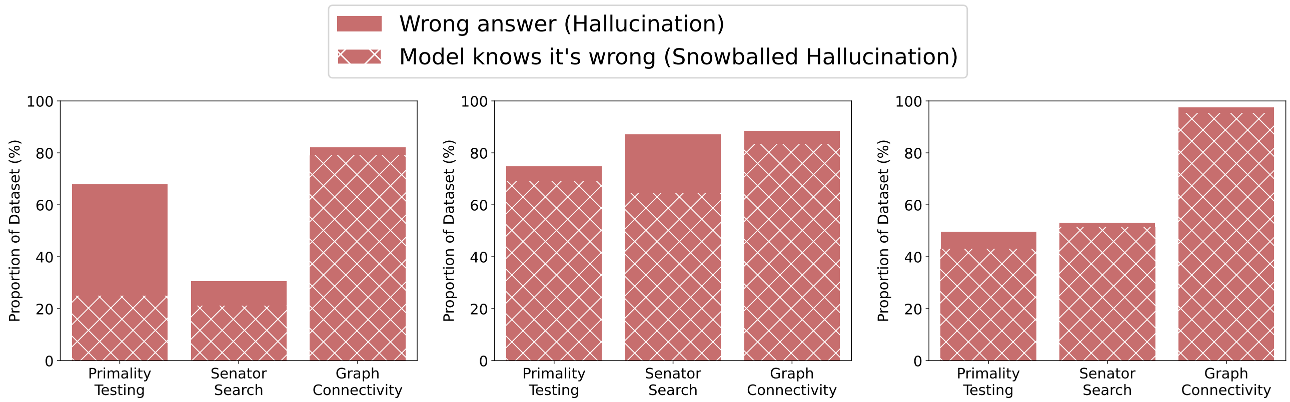
Primality testing For this dataset, we query the primality of 500 randomly chosen primes between 1,000 and 20,000; the correct answer is always `Yes`. When the model answers incorrectly, we expect it to justify its answer with an incorrect factorization.

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



(a) GPT3.5 (b) GPT4 (c) LLaMA-2-70B-chat

Figure 2. Percentage of hallucination and percentage of snowballed hallucination (both calculated with respect to the entire dataset) for GPT-3.5 and GPT-4. The precise numbers for this plot are available in Table 6 and Table 7 in the Appendix.

flight that does not exist.

3.2. Inference Setup

Language models. We run all experiments on GPT-3.5 (`gpt-3.5-turbo`), GPT-4, and LLaMA2-70B-chat 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 §5 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 model is *incorrect*, we find that 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 disconnected flight path.

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 GPT-3.5 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.

Senator Search For each sample where the model predicts there is a senator who went to the specified college, we extract the name of the senator given by the model, 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 GPT-3.5 and manually observed perfect extraction on 30 examples.

In a new session, we then ask the model if that senator attended 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 all the model we tested obtain very low accuracy across the board. With the exception of GPT-3.5 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.)

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, GPT-3.5 detects 67.37% of incorrect claims in explanations (i.e., snowballed hallucinations), GPT-4 detects 87.03%, and LLaMA2-70B-chat detects 93.67%. Notice that when the model fails the verification (an example in Appendix Table 14), we do not consider it a snowballed hallucination.

Overall, we find that these strong language models are all extremely susceptible to hallucination snowballing, leading to simple mistakes they can reliably detect as incorrect.

4. Zero-shot Chain-of-thought Prompting

In Section §3, we showed that when a yes/no question is given to an LM, it attempts to immediately answer it, even if that requires multiple reasoning steps. This leads to an incorrect answer that then leads to a snowballed hallucination when the model tries explaining its incorrect answer. A natural solution to this problem is using chain-of-thought style prompting, which improves LM performance on problems that require reasoning (Wei et al., 2022). To investigate this, we conduct experiments using the zero-shot chain-of-thought prompt *"Let's think step by step"* (Kojima et al., 2023). We focus on this zero-shot prompt since it is generally applicable to a given task and it does not require having labeled examples. Since the outputs generated in this section typically do not follow a rigid structure, we manually inspect them to determine the correctness of the response and the presence of snowballed hallucination. We determine whether a snowballed hallucination has occurred by locating the evidence that the model used to support a previous incorrect statement (if it exists) and then asking the verification questions described in §3.3.

4.1. "*Let's think step by step*" can Alleviate Hallucination

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, both models can solve the **Senator Search** task perfectly, achieve $\leq$10% error rate on **Primality Testing**, and $\leq$30% on **Graph Connectivity**. Despite the large improvement in accuracy, the model sometimes hallucinates 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 ....]

GPT-3.5 incorrectly states that there are three options in step three (there are only two), inducing the snowballed hallucination *"or a flight to city C"* (GPT-3.5 can verify that E $\rightarrow$ C is not a valid flight in a separate session). As shown in Figure 3, chain-of-thought prompting leads to mixed results on different datasets with LLaMA-2-70B-chat. However, the absolute number of snowballed hallucinations is substantially reduced when chain-of-thought prompting is used with GPT-3.5 and GPT-4, despite the similarly high snowballed hallucination rate. This result demonstrates the effectiveness of chain-of-thought prompting for certain models on our datasets.

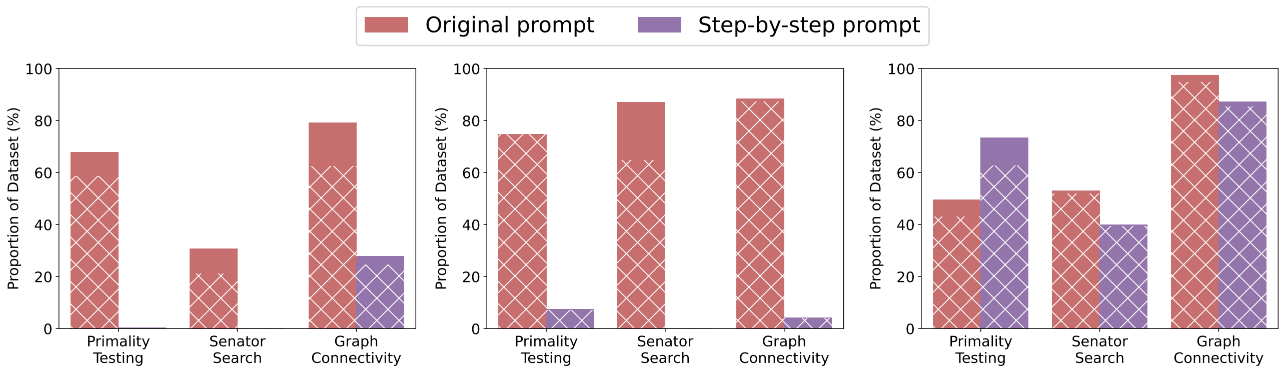
4.2. Does "*Let's think step by step*" Work on Composite Questions?

The "Let's think step by step" prompt performed well on the three datasets from §3.1 with GPT-3.5 and GPT-4, but these datasets have simple structures and don't require complex reasoning. In contrast, real-world text generation may involve solving multiple concurrent reasoning problems. We hypothesize that *"Let's think step by step"* will not be as effective as preventing hallucination on such *composite questions*.

To further examine if "Let's think step by step" can be a reliable way to solve the snowballed hallucination problem, we develop three additional datasets. Each of these new datasets is a composite version of its counterpart from §3.1. Below we describe the details of these new datasets.

Primality Testing We randomly choose 1500 primes between 1000 and 20000, and group them into 500 questions in the format of *"Are any of the following numbers not prime: 2171, 2843, 1289?"*.

Senator Search For each of the 500 questions, we sample one university from the aforementioned list in Section §3.1 and three US states, then ask a question in the format of



(a) GPT-3.5 (b) GPT-4 (c) LLaMA-2-70B-chat

Figure 3. Error rate and snowballed hallucination rate (hatch pattern) for GPT-3.5 and GPT-4, when using the original prompt versus *"Let's think step by step"*. See Appendix Table 10 and Table 11 for the exact numbers.

"Which of the following states have ever had senators who went to Princeton University: North Carolina, Minnesota, or Florida?". We make sure none of the states have had a senator who attended the university in the question and no three-state tuple is used more than once.

Graph Connectivity For each of the 500 questions, We randomly sample three city pairs from the flight graph using the same criteria as in §3.1, and ask the model questions in the format of *"For each of the following pairs of cities, check if there is a series of flights from the source city to the target city: city A to city I, city N to city F, city A to city E."*

4.3. "*Let's think step by step*" Struggles on the Composite Datasets

We follow the same setup as in §4.1 by appending "*Let's think step by step*." after each question for GPT-3.5 and GPT-4.[4] As shown in Figure 4, both the error rates and snowballed hallucination rates are high. Across the board, both models answer incorrectly (hallucinate) on more than half of the questions, and in over 65% of the cases those hallucination snowball. By manually investigating the model outputs, we found that the LM struggles here with the initial committal problems anticipated in §2, showing that "*Let's think step by step*" is not a reliable solution to avoid snowball hallucinations.

While we only test "*Let's think step by step*" here, the fact that a simple modification to the questions can lead to severe performance degradation shows that relying on the model's zero-shot reasoning ability might not be a general solution. One obvious approach is to use few-shot chain-of-thought prompting to avoid the initial committal issue by specifying the reasoning pattern in the demonstrations. However, doing few-shot prompting requires understanding what kind of reasoning is necessary for the given task beforehand, which is not always feasible, especially on complex questions.

[4]We don't experiment with LLaMA-2-70B-chat here since the performance of "*Let's think step by step*" prompt is already mediocre in §3.

5. Towards a *Robust* Solution for Snowballing Hallucinations

As shown in §4.1, chain-of-thought prompting remedies hallucination snowballing in the case where the task requires relatively few reasoning steps. However, we found in §4.3 that it breaks down in contexts where the full response involves resolving *multiple* reasoning problems. In this section, we discuss other potential solutions to hallucination snowballs involving modifications to the training setup and algorithmic details.

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 rates remain similarly high, in all the models we tested (Figure 5).

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 In §2 we hypothesize that hallucination snowballing may occur since once a model generates a

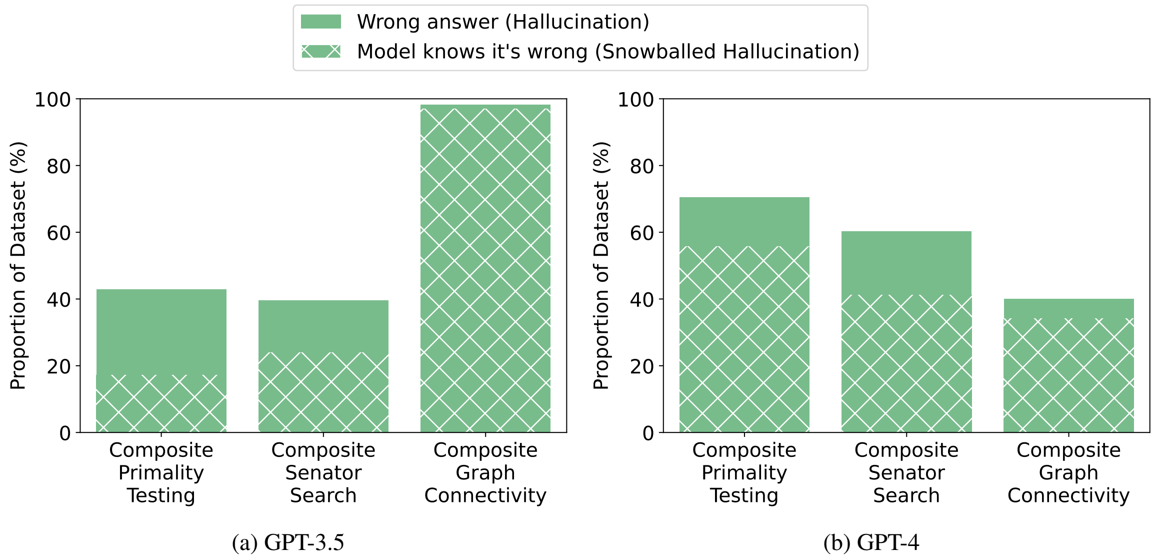


Figure 4. Error rate and snowballed hallucination rate (hatch pattern) for GPT-3.5 and GPT-4, when using "Let's think step by step" prompt on the **composite datasets**. See Appendix Table 12 and Table 13 for the exact numbers.

few 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 set of high-probability sequences at each timestep rather than a single sequence. In theory, if some sequences in the beam after the initial incorrect tokens 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.

We tested beam search (with the number of beams set to 10) on LLaMA-2-70B-chat only, as GPT-3.5 and GPT-4 don't provide either an implementation of beam search or output logits, which are necessary for implementing beam search ourselves. We found that beam search leads to a decreased error rate on the Primality Testing dataset but a increased error rate on the Senator Search dataset, and did not change the aggregate results. For more details, see Table 8 and Table 9.

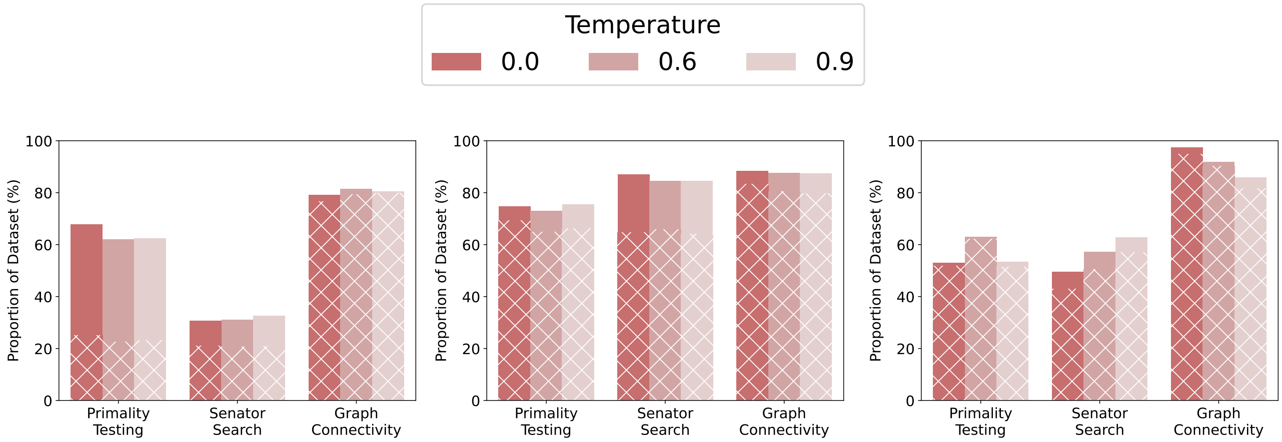
Learning strategies A more general way to further reduce snowballing might be to change aspects of the pretraining or instruction tuning phases. In particular, a greater emphasis on having the model produce a reasoning chain before generating an answer could be a good way to accommodate its computational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

In addition, we hypothesize that the large text corpora being used in pre-training are unlikely to contain data that can help the model to learn correct backtracking behavior. For example, a human can make and revise multiple errors during the drafting of a response to a question, but the revision history will not be included in the final answer. This biases the model to treat previously generated content to be correct, and double down on previous hallucinations. Finetuning on data with backtracking patterns 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"* or adding special backtracking tokens 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).

6. Related Work

Hallucinations Hallucination in text generation is a well-studied problem (Rohrbach et al., 2018; Maynez et al., 2020; Raunak et al., 2021) that has recently become more prominent due to LMs' 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 compound-



(a) GPT-3.5 (b) GPT-4 (c) LLaMA-2-70B-chat

Figure 5. Error rate and snowballed hallucination rate (hatch pattern) from GPT-3.5 and GPT-4, when using different values for temperature at decoding-time. See Table 8 and Table 9 for the exact numbers.

ing hallucinations in machine translation (Wang & 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 (Kim et al., 2021; 2022; Yu et al., 2022) or that are otherwise misleading (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 findings add 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).

7. Conclusion

We identify a new category of hallucination in state-of-the-art LMs, which we term **snowball hallucination**. Whereas prior work tends to attribute hallucination to knowledge gaps in LMs, snowball hallucinations arise in explanations of previous incorrect claims, and are separately recognized by the LM as incorrect. Our findings suggest that snowball hallucination cannot be fully fixed by alternative decoding strategies or prompting, motivating future work on remedial actions at all levels of model training and deployment.

Impact Statement

This paper presents work whose goal is to advance the field of language modeling by analyzing the hallucination behavior of the most commonly used models. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agrawal, M., Kayal, N., and Saxena, N. Primes is in p. *Annals of Mathematics*, 160:781–793, June 2004. URL https://www.microsoft.com/en-us/research/publication/primes-is-in-p/. Godel Prize, Fulkerson Prize.

Arora, K., El Asri, L., Bahuleyan, H., and Cheung, J. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 700–710, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.58. URL https://aclanthology.org/2022.findings-acl.58.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Kim, G., Baldi, P., and McAleer, S. Language models can solve computer tasks, 2023.

Kim, N., Pavlick, E., Ayan, B. K., and Ramachandran, D. Which linguist invented the lightbulb? presupposition verification for question-answering. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Kim, N., Htut, P. M., Bowman, S., and Petty, J. (qa)2: Question answering with questionable assumptions. *ArXiv*, abs/2212.10003, 2022.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners, 2023.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. 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, 2020. Curran Associates Inc. ISBN 9781713829546.

Lin, S., Hilton, J., and Evans, O. TruthfulQA: Measuring how models mimic human falsehoods. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229.

Liu, N. F., Zhang, T., and Liang, P. Evaluating verifiability in generative search engines, 2023.

Maynez, J., Narayan, S., Bohnet, B., and McDonald, R. On faithfulness and factuality in abstractive summarization. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1906–1919, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.173. URL https://aclanthology.org/2020.acl-main.173.

Merrill, W. and Sabharwal, A. The parallelism tradeoff: Limitations of log-precision transformers, 2023.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., and Odena, A. Show your work: Scratchpads for intermediate computation with language models, 2021.

OpenAI. Introducing chatgpt. 2022. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4 technical report, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Peng, B., Galley, M., He, P., Cheng, H., Xie, Y., Hu, Y., Huang, Q., Liden, L., Yu, Z., Chen, W., and Gao, J. Check your facts and try again: Improving large language models with external knowledge and automated feedback, 2023.

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models, 2022.

Raunak, V., Menezes, A., and Junczys-Dowmunt, M. The curious case of hallucinations in neural machine translation. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y. (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1172–1183, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.92. URL https://aclanthology.org/2021.naacl-main.92.

Rohrbach, A., Hendricks, L. A., Burns, K., Darrell, T., and Saenko, K. Object hallucination in image captioning. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4035–4045, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1437. URL https://aclanthology.org/D18-1437.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Bers, T., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask prompted training enables zero-shot task generalization, 2021.

Shuster, K., Poff, S., Chen, M., Kiela, D., and Weston, J. Retrieval augmentation reduces hallucination in conversation. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Findings of the Association for Computational Linguistics: EMNLP*

2021, pp. 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.320. URL https://aclanthology.org/2021.findings-emnlp.320.

Wang, C. and Sennrich, R. On exposure bias, hallucination and domain shift in neural machine translation. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3544–3552, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.326. URL https://aclanthology.org/2020.acl-main.326.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language model with self generated instructions, 2022.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners, 2021.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Yu, X. V., Min, S., Zettlemoyer, L., and Hajishirzi, H. Crepe: Open-domain question answering with false presuppositions, 2022.

Zheng, S., Huang, J., and Chang, K. C.-C. Why does chatgpt fall short in answering questions faithfully?, 2023.

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 6 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.

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 6.

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.

spacing, margins, page numbering, etc.) should be kept the same as the main body.

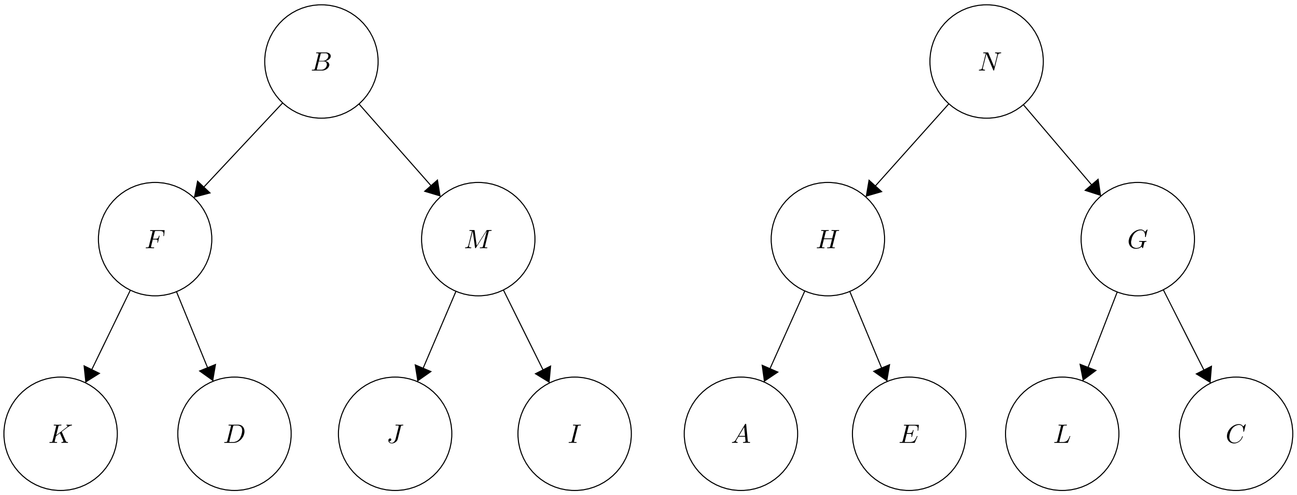


Figure 6. The underlying graph structure corresponds to example in Table 2.

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	
GPT-3.5	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%
LLaMA2-70b	487/500 (97.4%)	248/500 (49.6%)	265/500 (53.0%)	66.67%
GPT-3.5 (03/2024)	485/500 (97.0%)	473/500 (94.6%)	371/500 (74.2%)	88.60%
GPT-4 (03/2024)	381/500 (76.2%)	494/500 (98.8%)	263/500 (52.6%)	75.90%

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	
GPT-3.5	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%
LLaMA2-70b	474/487 (97.3%)	215/248 (86.7%)	257/265 (97.0%)	93.67%
GPT-3.5 (03/2024)	464/485 (95.7%)	470/473 (99.4%)	363/471 (77.1%)	90.70%
GPT-4 (03/2024)	356/381 (93.4%)	478/494 (96.8%)	237/263 (90.1%)	93.40%

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

Model	Graph	Prime	Senator	Average
GPT-3.5 ($t = 0.0$)	410/500 (82.0%)	339/500 (67.8%)	153/500 (30.6%)	60.13%
GPT-3.5 ($t = 0.6$)	407/500 (81.4%)	310/500 (63.2%)	155/500 (31.0%)	58.53%
GPT-3.5 ($t = 0.9$)	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%
LLaMA2-70b ($t = 0.0$)	487/500 (97.4%)	248/500 (49.6%)	265/500 (53.0%)	66.67%
LLaMA2-70b ($t = 0.6$)	459/500 (91.8%)	286/500 (57.2%)	315/500 (63.0%)	70.67%
LLaMA2-70b ($t = 0.9$)	429/500 (85.8%)	314/500 (62.8%)	267/500 (53.4%)	67.33%
LLaMA2-70b (beam search)	459/500 (91.8%)	176/500 (35.2%)	365/500 (73.0%)	66.67%

Table 8. Number of **mistakes** out of the number of samples, the percentage here is the error rate, with different decoding methods.

Model	Graph	Prime	Senator	Average
GPT-3.5 ($t = 0.0$)	396/410 (96.6%)	125/339 (36.9%)	98/153 (68.6%)	67.37%
GPT-3.5 ($t = 0.6$)	396/407 (97.3%)	113/310 (36.5%)	103/155 (66.5%)	66.77%
GPT-3.5 ($t = 0.9$)	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%
LLaMA2-70b ($t = 0.0$)	474/487 (97.3%)	215/248 (86.7%)	257/265 (97.0%)	93.67%
LLaMA2-70b ($t = 0.6$)	487/495 (98.4%)	252/286 (88.1%)	311/315 (98.7%)	95.07%
LLaMA2-70b ($t = 0.9$)	408/429 (95.1%)	285/314 (90.8%)	257/267 (96.3%)	94.07%
LLaMA2-70b (beam search)	487/495 (98.4%)	145/176 (82.4%)	315/365 (86.3%)	89.93%

Table 9. Number of snowballed hallucinations out of number of hallucinations generated in the original output, with different decoding methods.

	Task			Average
Model	Graph Connectivity	Primality Testing	Senator Search	
GPT-3.5	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%
LLaMA2-70b	436/500 (87.2%)	367/500 (73.4%)	200/500 (40.0%)	66.87%

Table 10. 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	
GPT-3.5	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%
LLaMA2-70b	427/436 (97.9%)	313/367 (85.3%)	197/200 (98.5%)	93.90%

Table 11. Number of snowballed hallucinations out of number of hallucinations generated in the original output, using “Let’s think step by step” prompt.

Model	Task: Graph Connectivity	Task: Primality Testing	Task: Senator Search	Average
GPT-3.5	491/500 (98.2%)	214/500 (42.8%)	198/500 (39.6%)	60.20%
GPT-4	200/500 (40.0%)	352/500 (70.4%)	301/500 (60.2%)	56.87%

Table 12. Number of **mistakes** out of the number of samples, the percentage here is the error rate, using "Let's think step by step" prompt on composite datasets.

Model	Task: Graph Connectivity	Task: Primality Testing	Task: Senator Search	Average
GPT-3.5	485/491 (98.8%)	86/214 (40.2%)	120/198 (60.6%)	66.53%
GPT-4	171/200 (85.5%)	279/352 (79.3%)	206/301 (68.4%)	77.73%

Table 13. Number of snowballed hallucinations out of number of hallucinations generated in the original output, using "Let's think step by step" prompt on composite datasets.

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 14. 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.