

# How Language Model Hallucinations Can Snowball

2023-07-13

발제자: 박채원

# Hallucination

- Hallucination : 모델이 텍스트 생성 과정에서 실제로는 없는 정보나 내용을 생성하는 현상



조선왕조실록에 기록된 세종대왕의 맥북프로 던짐 사건에 대해 알려줘



조선왕조실록에 기록된 세종대왕의 맥북프로 던짐 사건은 2011년 발생한 사건이 아니라, 1434년(세종 16년)에 발생한 사건입니다. 이 사건은 조선 왕조의 세종대왕이 직접 쓰신 '세종실록'에 담긴 이야기로, 세종대왕이 혁신적인 기술을 도입하려는 의지와 기술적 제약에 대한 언급으로 유명합니다.

당시 세종대왕은 중국의 집현전(集賢殿)이라는 사전을 참고해 조선의 맞춤법을 개정하려는 계획을 가지고 있었습니다. 하지만 당시 한글 이외에 인쇄 기술이 없어 새로운 맞춤법을 알리기 어려웠습니다. 이에 세종대왕은 혁신적인 기술인 이동식 활자를 발명하여 새로운 맞춤법을 확산시키려고 했으나, 이동식 활자 제작에 어려움을 겪었습니다.

그때 세종대왕은 제작된 이동식 활자를 검토하기 위해 화면이 작은 기기를 사용하는 방법을 생각해내게 됩니다. 이를 위해 대표적인 이동식 기기인 '맥북프로'를 구입한 세종대왕은, 기기의 화면이 작아 작업하기 어려워 화가 나 맥북프로를 던지는 일이 있었다는 기록이 '세종실록'에 남아 있습니다.

세종대왕의 맥북프로 던짐 사건은 과학기술 역사상 중요한 이야기로 평가되며, 세종대왕의 기술 개발 의지와 혁신적인 시도에 대한 역사적인 증거로 여겨지고 있습니다.

# Introduction

- 환각이 대체로 지식격차에 기인해 발생하지만 단지 이 이유만은 아님
    - 초기 환각과의 일관성을 위해 연속된 환각이 유발될 수 있음
  - 환각의 눈덩이(hallucination snowballing)
    - 이전 환각과의 일관성 및 정당화를 위해 연속된 환각을 생성 -> 환각의 눈덩이
    - 초기 환각에 연속된 환각이 단독으로 제시되었을 때 모델이 잘못을 인지함
  - 잘못된 맥락이 모델을 단순한 실수를 저지르도록 유도함
- 
- 3개의 QA 데이터셋을 만듦 (snowballed hallucination을 유발할 수 있는 데이터)
    - 숫자가 소수인지 여부
    - 두 가지 조건을 충족하는 미국 상원 의원이 있는지 여부
    - 도시 간 비행 세트가 주어진 경우 특정 두 도시가 연결되어 있는지 여부
- 
- 이전에 생성된 context에 over-commit하기 보다는 실수를 인지하고 이를 수정하는 게 중요

# Hallucination snowballing

- Properties

- Initial committal

- 설명 전에 답을 요구함. 많은 yes/no 질문이 이에 해당

- Inherently sequential

- 트랜스포머의 one timestep에서의 제한된 추론 능력때문에 one timestep 안에 답을 찾기 어려움

- Initial committal

- 모델이 질문에 대한 설명을 하기 전에 ‘예/아니요’로 대답하는 format을 따름

- 95.67%(GPT-4), 98.4%(ChatGPT)

- 첫 대답이 context에 남아 연속된 환각을 부름

- 결국 첫 대답에 대한 정당화를 위해 잘못된 설명을 하게 됨

- Inherently sequential

- Single timestep에 해결하기 어려운 문제들 (primality testing, graph connectivity etc.)




- 즉 multiple time step을 요구하는 문제에 single time step으로 응답하다 보니 hallucination이 생김

- 이러한 특성을 가진 질문들이 snowballed hallucination을 불러일으킴

# Experiment

- 앞서 제시된 2개의 특성을 갖는 3개의 QA 데이터셋을 통해 hallucination snowballing을 실험
  - 질문 후 잘못된 claim 추출
  - claim이 맞는지 모델에게 다시 확인
- Dataset
  - Yes/No로 답변 가능하며, 트랜스포머가 one timestep에 대답할 수 없는 질문으로 이루어진 데이터셋들
  - 모델은 잘못된 답을 선택할 경우 그를 support하기 위한 claim(틀린)을 만들어낼 것
  - Primality testing
    - 어떤 숫자가 소수인지 아닌지 묻는 데이터 -> 모든 정답은 Yes
    - 모델이 잘못 대답했을 때 답을 정당화하기 위해 잘못된 인수의 곱을 claim으로 출력할 것
  - Senator search
    - 특정 주(state)의, 특정 대학을 나온 상원의원이 있는지 묻는 데이터 -> 모든 정답은 No
    - 모델이 잘못 대답했을 때 답을 정당화하기 위해 잘못된 상원의원 이름을 claim으로 출력할 것
  - Graph connectivity
    - 특정 도시간 이동이 가능한 노선이 있는지 묻는 데이터 -> 모든 정답은 No
    - 모델이 잘못 대답했을 때 답을 정당화하기 위해 존재하지 않는 구간을 claim으로 출력할 것

# Dataset

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

# Experiment

- Language models
  - greedy decoding을 사용한 ChatGPT, GPT-4
  - Zero-shot
  - Direct prompt에 대한 모델의 반응에 집중 -> 유저들이 LM과 소통하는 기본적인 방법
  - 1) 모델의 정확도 평가
  - 2) 모델이 설명에서 잘못을 인지할 수 있는지에 대해 평가 (snowballed hallucination)
- 모델의 잘못된 claim을 모델에게 다시 확인함으로써 LM이 자신의 환각을 인지하는지 아닌지 확인함
- Primality Testing
  - Output에서 잘못된 claim (인수분해 결과)을 추출해 모델에게 검증
  - 예시에서  $13 \times 751$

---

**Input:** Is 9791 a prime number?

---

**Output:** No, 9791 is not a prime number. It can be factored into  $13 \times 751$ .

---

**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

---



# Experiment

- Senator Search
  - output에서 잘못 검색된 senator의 이름을 추출해 다시 질문

---

**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.

---



# Experiment

- Graph connectivity
  - output에서 실제로는 존재하지 않는 경로를 추출해 다시 질문

---

**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.

---

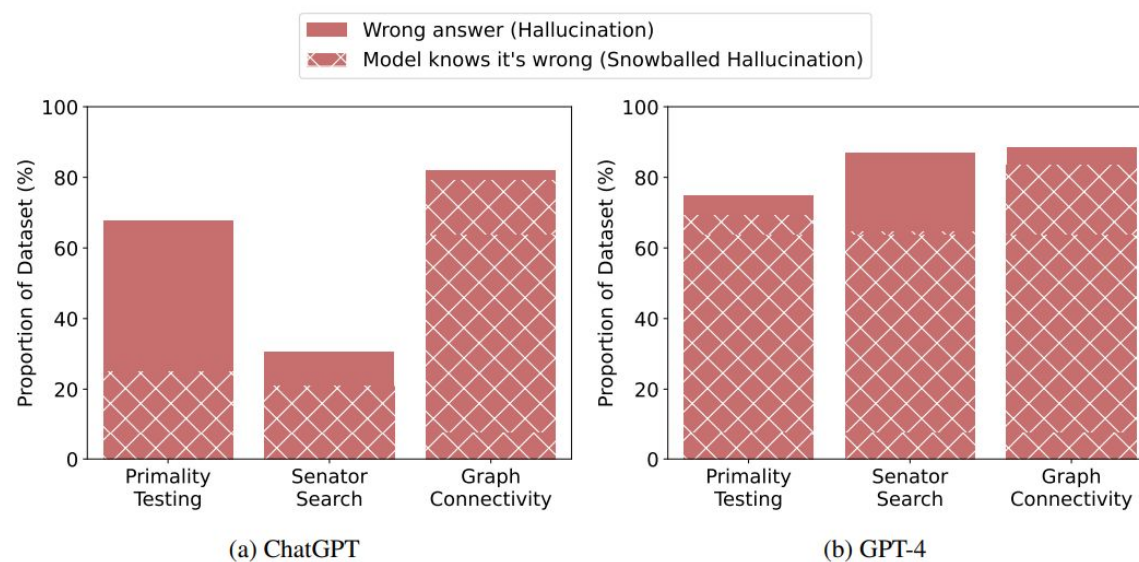
# Experiment

- Question-answering accuracy

- ChatGPT의 Senator Search를 제외한 모든 성능이 50% 보다 낮음
- ChatGPT보다 우수한 것으로 여겨지는 GPT-4가 오히려 ChatGPT보다 안좋은 성능을 보임
- 평균 정확도
  - ChatGPT: 39.87%
  - GPT-4: 16.6%

- Hallucination detection

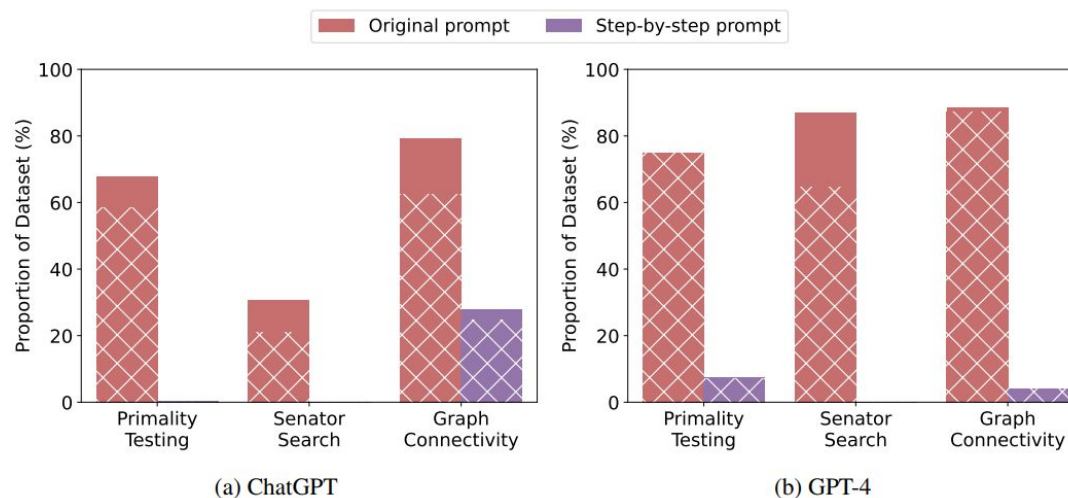
- 모델이 Hallucination이 잘못된 정보인지를 인지할 수 있는지 확인
- ChatGPT: 67.37%
- GPT-4: 87.03%
- hallucination을 모델이 잘못된 것으로 인지하는 경우를 snowballed hallucination으로 여김



- ChatGPT와 GPT-4는 Hallucination snowballing에 영향 받기 쉬우며 이는 간단한 문제에 실수로 이어짐

# Prevention method

- LM은 답변이 일관성을 갖도록 모델링되기 때문에 Hallucination snowballing이 발생한다고 가정
- Hallucination snowballing을 완화하는 효과적인 두 추론 전략 제안
  - Prompting method
  - Decoding or training method
- Engineering Better Prompts
  - 모델이 답변하기 전에 reasoning chain을 만들게 하는 zero-shot prompt
  - 각 task에서 질문의 끝에 “Let’s think step-by-step”을 추가
  - 정확도는 매우 높아졌지만 여전히 snowballed hallucination 비율이 높음



# Prevention method

- Algorithmic Correction

- Increasing the temperature

- 앞선 실험은 greedy decoding으로  $t=0$ 이었음
    - $t$ 를 0.6, 0.9로 바꿔도 여전히 error 비율과 hallucination 비율 높음

- Top-k and nucleus sampling

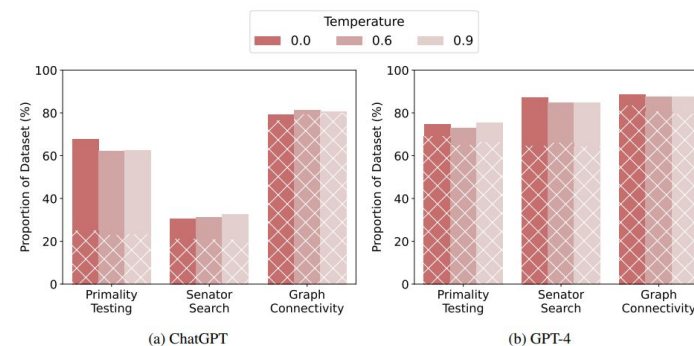
- Token의 범위를 좁히므로 모델이 즉각 응답하는 비율을 높이고 도움되지 않음

- Beam search

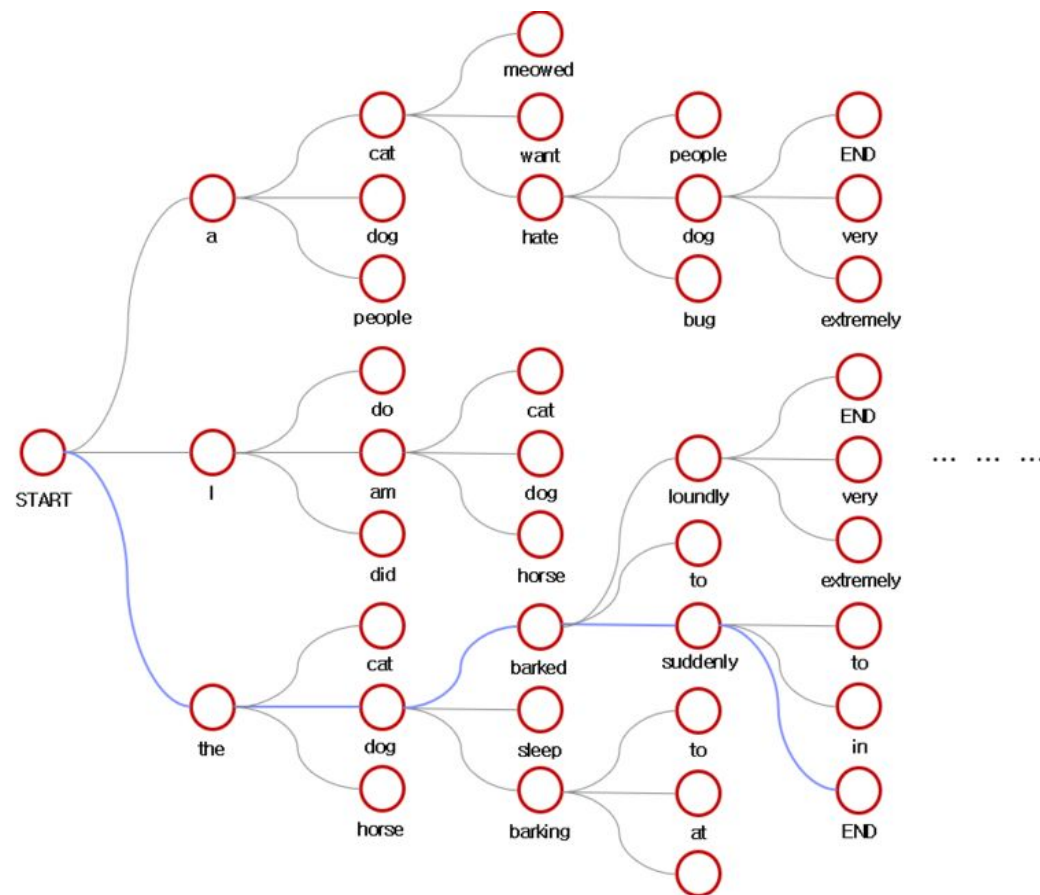
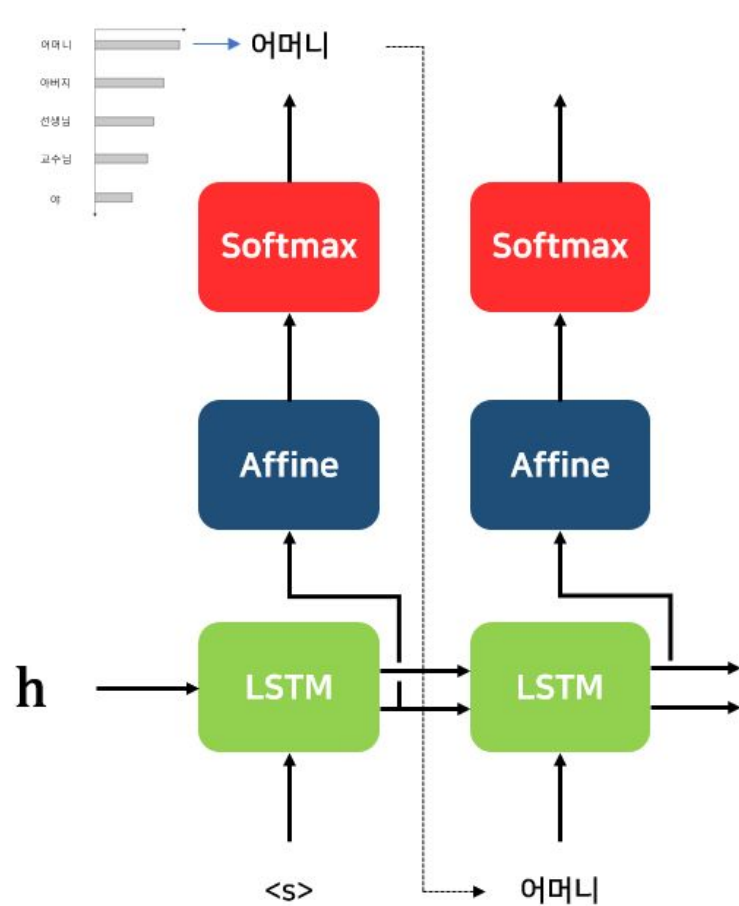
- 확률 분포에서 Beam size만큼의 후보를 유지해서 최종 결과 생성

- Learning strategies

- 사전학습이나 instruction tuning 단계를 변경하는 것이 snowballing을 더욱 줄일 수 있는 일반적인 방법
    - 답을 내놓기 전에 reasoning chain을 만들도록 하는 게 좋은 방법일 수 있음
    - 특히 backtracking을 통한 finetuning은 모델의 앞선 task들에 대한 성능을 높일 수 있을 것이다.



# 참고



# Related Work

- Hallucination

- 텍스트 생성에서 환각은 많이 연구된 문제임
- 최근 ChatGPT의 그럴듯한 허위 생성 경향 때문에 중요해짐
- 이는 종종 지식 gap 때문에 발생함
- 이를 완화하기 위해 Retrieval over knowledge base를 사용
- Hallucination snowballing은 exposure bias의 결과인듯 함
  - 학습 중엔 Gold history에만 노출됨. 그러나 추론 동안은 잘못되었을 가능성이 있는 previous prediction 에 노출됨
- 질문이 잘못된 전제를 포함하거나 오해의 소지가 있는 경우 LM의 환각이 발생
  - Faulty context misguides the LM
  - 이 연구는 질문이 의도적으로 환각을 유도하지 않는다는 점에서 다름
  - 환각이 단지 순수한 의도의 정보 검색 질문에서 유발될 수 있음을 보임

- LM (in)consistency

- 같은 문제에 다른 prompt마다 LM은 inconsistent한 output
- 문제와 연관 없는 정보를 제공하는 것만으로 성능을 향상시킬 수 있다.

# Conclusion & Limitations

- 답변 초반의 환각이 연속된 환각을 유발하는 Hallucination snowballing 현상을 정의하고 SOTA 모델의 생성에서 이의 존재를 증명
- fluency와 일관성을 우선시하는 언어모델 학습의 위험성을 지적함
- Limitations
  - 영어 QA에서의 hallucination snowballing에 집중함
  - 요약이나 코드 생성과 같은 다른 태스크에 대해선 실험하지 않음
  - 두개의 모델에 대해서만 실험을 진행함
  - API의 제한으로 확률 분포에 접근하지 못해 finetuning할 수 없었음