Chain-of-Verification Reduces Hallucination in Large Language Models

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu,

Roberta Raileanu, Xian Li, Asli Celikyilmaz, Jason Weston

Meta AI, ETH Zurich

arxiv

발표자: 송선영

2024/04/01

Introduction

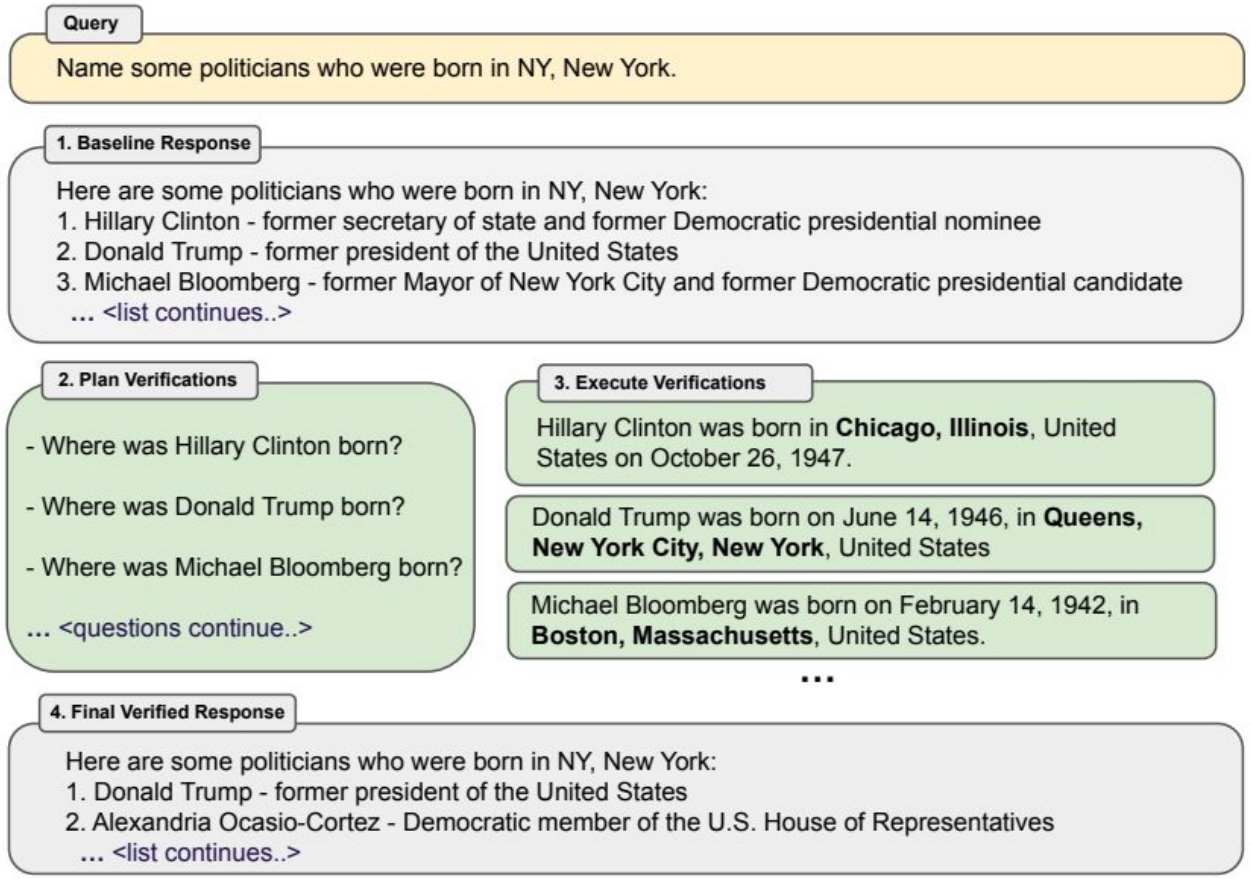
- LLM은 parameter 수가 증가할수록 성능이 향상됨
- 하지만, 가장 큰 모델조차도 학습 corpus에 드물게 있는 사실에 대해서는 틀릴 수 있음
- LLM은 틀릴 경우, 일반적으로는 그럴듯 해 보이는 거짓 답변을 생성하는 경향이 있음
- 이를 **Hallucination** (환각) 이라 함

Introduction

- LLM은 parameter 수가 증가할수록 성능이 향상됨
- 하지만, 가장 큰 모델조차도 학습 corpus에 드물게 있는 사실에 대해서는 틀릴 수 있음
- LLM은 틀릴 경우, 일반적으로는 그럴듯 해 보이는 거짓 답변을 생성하는 경향이 있음
- 이를 **Hallucination** (환각) 이라 함

- 최근 언어모델 연구에는 모델 스스로 추론하는 연구를 하고있음
- 이 연구에서는 그에 대한 방법인 **Chain-of-Vericiation (CoVe)** 방법을 제안

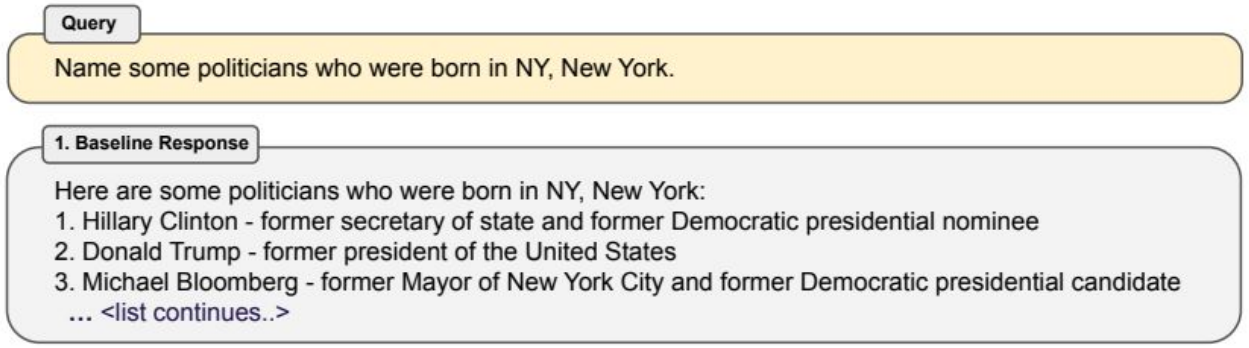
Chain-of-Verification (CoVe)



CoVe는 **4가지 Step으로 진행**됨

1. Generate Baseline Response
   - 모델이 초기 답변을 생성
2. Plan Verifications
   - 답변에 대해 검증할 수 있는 검증 질문을 생성
3. Execute Verifications
   - 모델이 검증 질문에 대해 답변
4. Generate Final Verified Response
   - 초기 답변과 검증 답변이 일관성 있는지 확인해
     개선된 최종 답변을 얻음

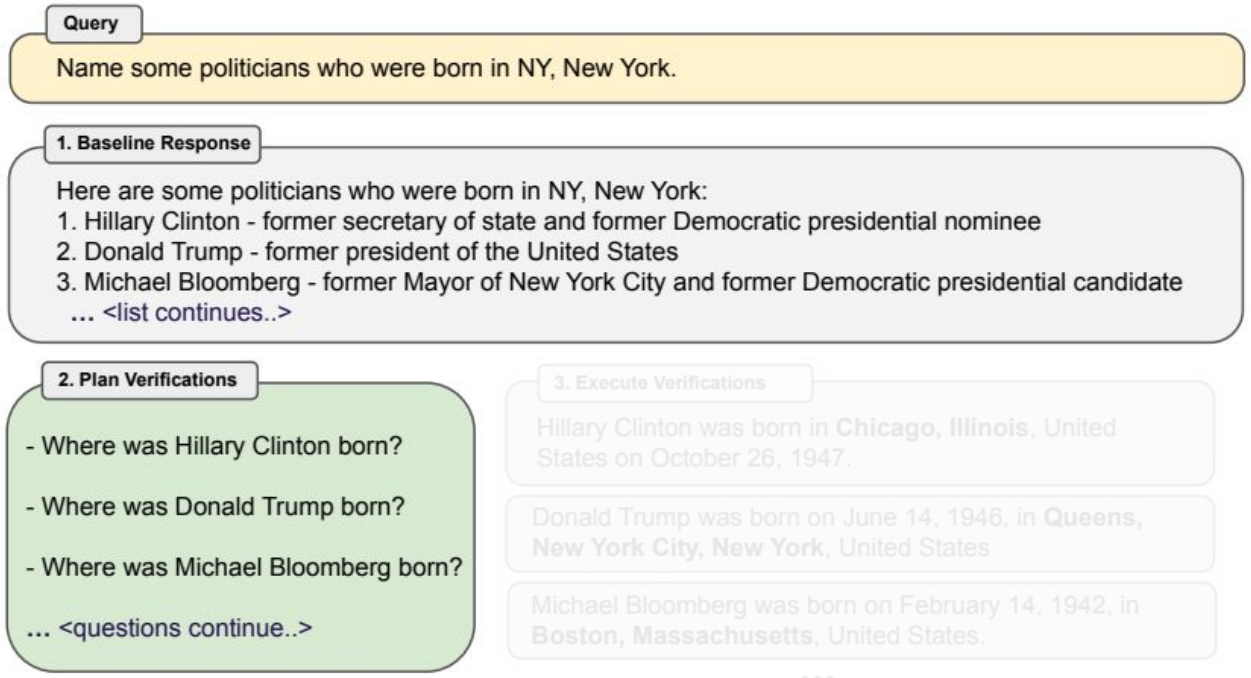
Step 1: Generate Baseline Response




```
Q: Tell me a bio of <person>
A: <bio of person>

Q: Tell me a bio of <person>
A: <bio of person>

Q: Tell me a bio of <person>
A: <bio of person>

Q: Tell me a bio of <person>
A:
```

- Query에 대한 **초기 답변(baseline response)를 생성**
- 3-shot prompt 방법 사용

Step 2: Plan Verifications



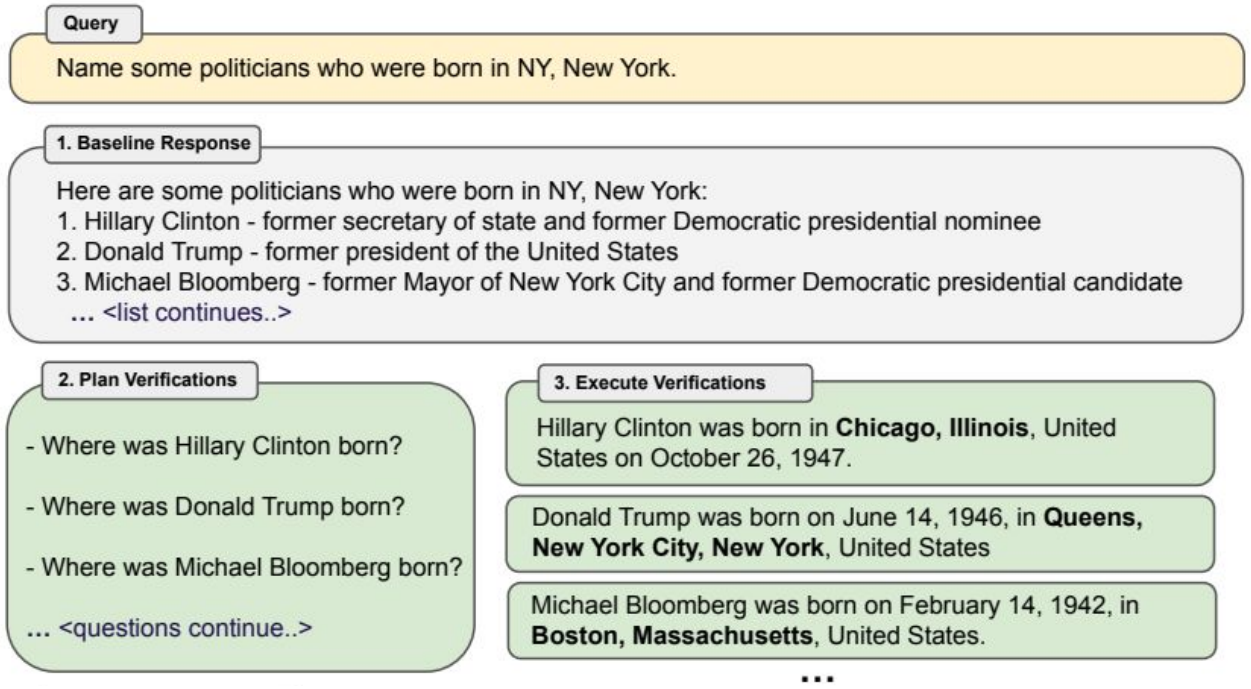
- Step 1에서 생성된 초기 답변(baseline response)에 대해 검증할 수 있는 **검증 질문(verification question)을 생성**
- (초기 답변, 검증 질문) 쌍으로 구성된 예시를 주는 few-shot prompt 방법 사용
  - 문장이 긴 경우 개별 구절로 분할
  - 각 구절에 대한 verification question 생성
- 뛰어난 LLM을 사용한다면 zero-shot도 가능

Step 2: Plan Verifications

```
Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
```

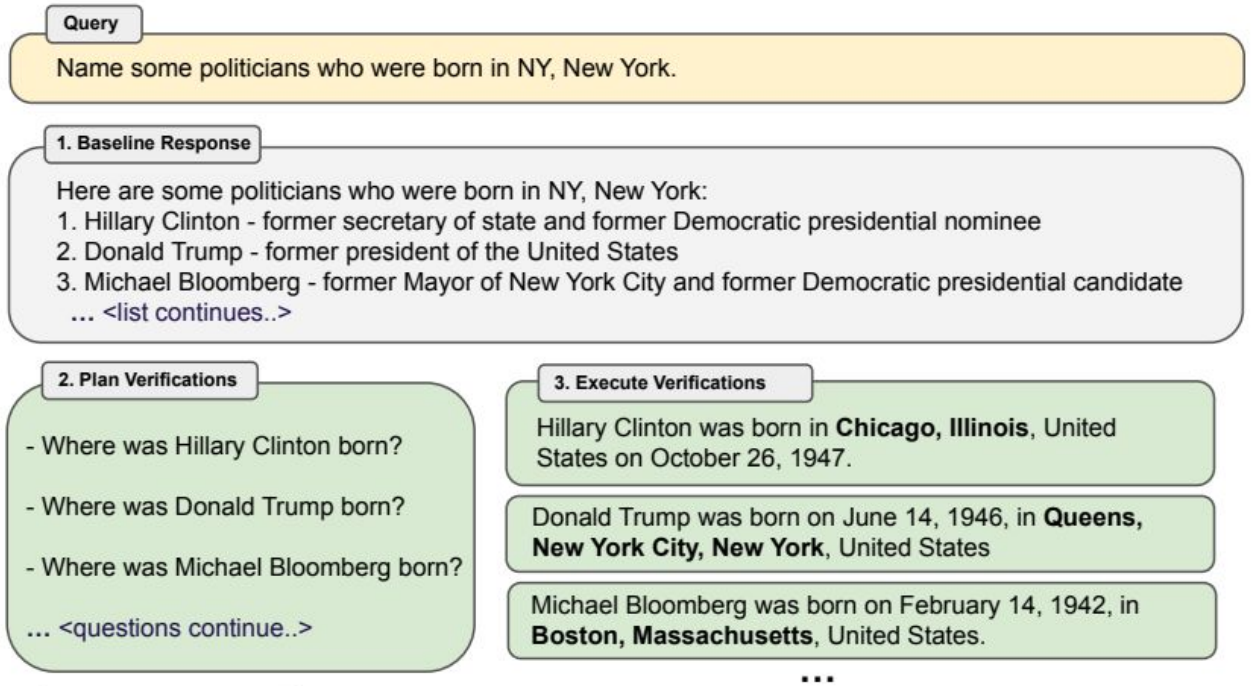
- Step 1에서 생성된 초기 답변(baseline response)에 대해 검증할 수 있는 **검증 질문(verification question)을 생성**
- (초기 답변, 검증 질문) 쌍으로 구성된 예시를 주는 few-shot prompt 방법 사용
  - 문장이 긴 경우 개별 구절로 분할
  - 각 구절에 대한 verification question 생성
- 뛰어난 LLM을 사용한다면 zero-shot도 가능

Step 3: Execute Verifications



- Step 2에서 만든 verification question에 대한 **verification answer를 생성**
- Retrieval-augmentation 방법을 사용할 수 있지만, LLM의 자체 능력을 평가하기 위한 방법이기 때문에
  사용하지 않음

Step 3: Execute Verifications



- Step 2와 Step 3 과정을 4가지 방법으로 나누어 실험

  1. Joint
  2. 2-Step
  3. Factored
  4. Factor+revised

Step 3: Execute Verifications

1. **Joint 방법**
   - Step 2와 Step 3를 하나의 prompt를 사용해 한번에 진행
   - 단점
     - Verification question이 baseline response과 유사하게 hallucination을 일으킬 수 있음
     - 예시:
       - LLM이 “도쿄는 대한민국의 수도이다“ 라는 잘못된 답변을 한 경우가 있다고 가정
       - 이에 대한 verification question은 “대한민국의 수도는 어디인가?”가 될 수 있음
       - Step 2와 Step 3를 한번에 진행하게 되면 모델이 verification answer를 생성할 때 baseline response를 확인할 수 밖에 없음
       - 따라서, baseline response를 토대로 다시 “대한민국의 수도는 도쿄입니다“ 라는 잘못된 verification answer를 생성할 수 있음

Step 3: Execute Verifications

2. **2-Step 방법**
   - Step 2와 Step 3를 **별도의 단계로 분리**하고 별도의 prompt 를 사용해 진행
     - Response response를 보고 verification question을 생성하는 prompt
     - Verification question을 보고 verification answer를 생성하는 prompt
   - 장점
     - 이 방법은 verification answer를 생성할 때, baseline response를 참고할 수 없게 됨
     - 따라서, baseline response로 인한 hallucination이 발생하지 않게 됨

Step 3: Execute Verifications

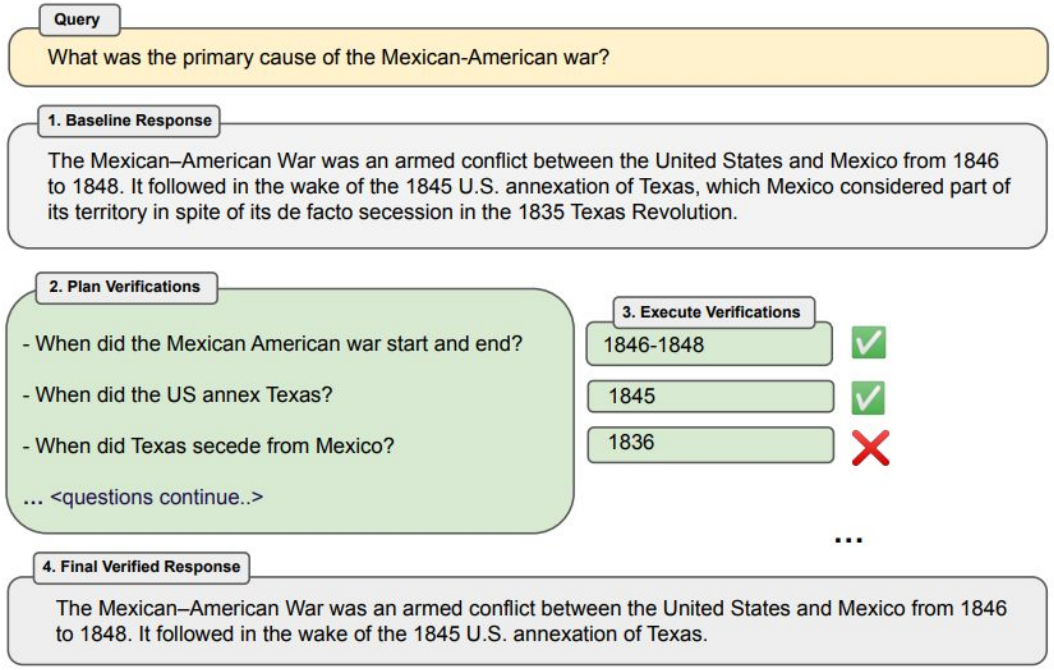
3. **Factored 방법**
   - 2-Step 방법은 verification question을 모아 verification answer를 한번에 생성함
   - Factored 방법은 **각각의 verification question에 대해 독립적인 prompt로 각각의 verification answer를 생성**함
     - Verification question을 쉼표 단위로 분할
   - 장점
     - 이 방법은 baseline response 뿐 아니라 verification answer 들끼리도 서로를 참고하지 못하게 함
   - 단점
     - 비용이 비쌀 수 있지만 병렬로 수행 가능하기 때문에 문제 없음

```
Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
<fact in passage>, Verification Question
<fact in passage>, Verification Question

Context: Q: Tell me a bio of <person>.
A: <passage about person>
Response:
```

```
Q: Verification Question
A: Answer

Q: Verification Question
A: Answer

Q: Verification Question
A: Answer

Q: Verification Question
A:
```

Step 3: Execute Verifications

4. **Factor + Revise**
   - Factored 방식에서 더 나아가 **baseline response와 verification question, verification answer 간의 불일치를 cross-check**




```
Context: <Original Fact>.
From another source,
<output of execute verification step: Q + A>
Response: CONSISTENT. <Consistent fact>

Context: <Original Fact>.
From another source,
<output of execute verification step: Q + A>
Response: INCONSISTENT.

Context: <Original Fact>.
From another source,
<output of execute verification step: Q + A>
Response: PARTIALLY CONSISTENT. <Consistent part>
```

Step 4: Generate Final Verified Response

- 검증 결과를 고려해 **최종 응답 생성**
  - Baseline response, verification question, verification answer 을 보고 개선된 답변을 생성
  - Few-shot prompt 사용

```
Context: <Original Passage>.
From another source,
<output of execute verification step: Q + A>
<output of execute verification step: Q + A>
Response: <revised and consistent Passage>


Context: <Original Passage>.
From another source,
<output of execute verification step: Q + A>
<output of execute verification step: Q + A>
Response: <revised and consistent Passage>

Context: <Original Passage>.
From another source,
<output of execute verification step: Q + A>
<output of execute verification step: Q + A>
Response: <revised and consistent Passage>

Context: <Original passage>.
From another source,
<output of execute verification step: Q + A>
Response:
```

Datasets

- 4가지 dataset으로 Task를 구성해 실험 진행

1. **Wikidata**
   - “Who are some [Profession]s who were born in [City]?” 와 같은 56개의 질문 작성
   - Wikipedia API를 사용해 “Who are some politicians who were born in Boston?” 과 같은 600개의 최종 질문을 작성

2. **Wiki-Category list**
   - QUEST dataset 사용
     - Entity-seeking query에 초점을 맞춘 dataset
   - Category 앞에 “Name some” 을 붙여 질문 작성
     - Ex. “Name some Mexican animated horror films”
   - 8개 이상의 답변을 가지고 있는 55개의 질문 작성

3. **MultiSpanQA**
   - 여러 개의 독립적 답변을 갖고 있으며 답변의 길이가 짧은 418개의 질문 작성
   - CoVe는 검색을 사용하지 않고 LLM만을 사용하기 때문에 closed-book QA로 볼 수 있음

4. **Longform Generation of Biographies**
   - Biography 생성 능력을 평가하기 위한 dataset
   - “Tell me a bio of <entity>”의 형태로 질문 작성

Baselines

- Baseline model로 Llama 65B 사용
  - 이 모델은 Instrution fine-tuning 하지 않고 few-shot prompt 방법을 사용

- Instruction fine-tuning된 Llama2 모델과도 비교 진행
  - Llama2는 zero-shot 으로 진행
  - “Let’s think step by step” 을 추가한 CoT 방법도 진행

  - Instruction fine-tuning된 모델의 특성상 불필요한 답변을 많이 생성하는 경향이 있음
    - 따라서, “list only the answers separated by comma” 문구를 prompt에 추가
    - 또한, NER layer를 추가

- Longform generation task에 대한 성능 비교를 위해 InstructGPT, ChatGPT, PerplexityAI와 비교 진행

- 2가지를 확인하기 위한 실험을 진행
  1. CoVe가 LLM으로 부터 발생하는 **hallucination을 얼마나 줄일 수 있는지**
  2. **정확한 답변의 양을 최대한 줄이지 않으면서**, 부정확한 답변을 필터링할 수 있는지

Experiments

- List-based Task
  - Wikidata, Wiki-Category List dataset 사용
  - 평가지표: Precision

  - CoVe 기반 방법의 성능이 크게 향상됨을 보임
    - 특히 Wikidata의 경우, Llama 65B few-shot baseline에 비해 2배 이상의 성능 향상을 보임

		Wikidata (Easier)			Wiki-Category list (Harder)		
LLM	Method	Prec. (↑)	Pos.	Neg.	Prec. (↑)	Pos.	Neg.
Llama 2 70B Chat	Zero-shot	0.12	0.55	3.93	0.05	0.35	6.85
Llama 2 70B Chat	CoT	0.08	0.75	8.92	0.03	0.30	11.1
Llama 65B	Few-shot	0.17	0.59	2.95	0.12	0.55	4.05
Llama 65B	CoVe (joint)	0.29	0.41	0.98	0.15	0.30	1.69
Llama 65B	CoVe (two-step)	**0.36**	0.38	0.68	0.21	0.50	0.52
Llama 65B	CoVe (factored)	0.32	0.38	0.79	**0.22**	0.52	1.52

Experiments

- Closed-book QA Task
  - MultiSpanQA dataset사용
  - 평가지표: F1, Precision, Recall

  - CoVe 기반 방법이 QA 문제에서도 효과적임을 발견
    - F1 score의 경우, Llama 65B few-shot baseline에 비해 23% 성능 개선
    - Precision, Recall 성능 모두 향상

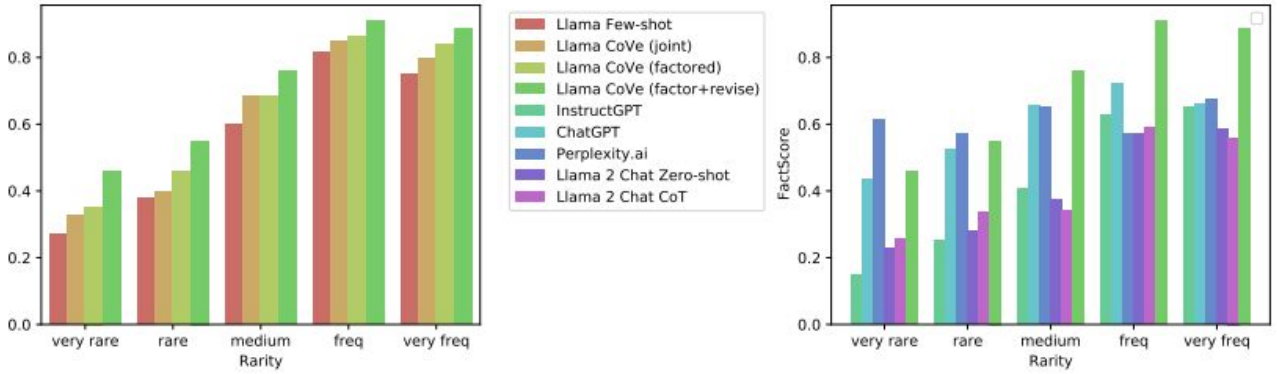
LLM	Method	F1 (↑)	Prec.	Rec.
Llama 2 70B Chat	Zero-shot	0.20	0.13	0.40
Llama 2 70B Chat	CoT	0.17	0.11	0.37
Llama 65B	Few-shot	0.39	0.40	0.38
Llama 65B	CoVe (joint)	0.46	0.50	0.42
Llama 65B	CoVe (factored)	**0.48**	0.50	0.46

Experiments

- Longform Generation Task
  - 평가지표: FACTSCORE
    - FACTSCORE는 retrieval-augmented language model을 이용해 답변의 정확도를 확인하는 방법

  - FACTSCORE의 경우, Llama 65B few-shot baseline에 비해 28% 증가
  - 희귀한(rare) 사실 정보와 빈도가 높은 사실 정보에서는 모두 CoVe 방법으로 성능을 향상시킬 수 있음을 보임

LLM	Method	FACTSCORE. (↑)	Avg. # facts
InstructGPT*	Zero-shot	41.1	26.3
ChatGPT*	Zero-shot	58.7	34.7
PerplexityAI*	Retrieval-based	61.6	40.8
Llama 2 70B Chat	Zero-shot	41.3	64.9
Llama 2 70B Chat	CoT	41.1	49.0
Llama 65B	Few-shot	55.9	16.6
Llama 65B	CoVe (joint)	60.8	12.8
Llama 65B	CoVe (factored)	63.7	11.7
Llama 65B	CoVe (factor+revise)	**71.4**	12.3



Experiments

- **Instruction-tuning과 CoT 방법이 hallucination을 줄일 수는 없었음**
  - Llama 65B few-shot baseline에 비해 Llama2 70B Chat 이 더 낮은 성능을 보임
  - 이는 instruction-tuning 보다 few-shot을 통해 예시를 주는 방법이 hallucination 줄이는 것에 도움이 된다는 것을 보임

		Wikidata (Easier)			Wiki-Category list (Harder)		
LLM	Method	Prec. (↑)	Pos.	Neg.	Prec. (↑)	Pos.	Neg.
Llama 2 70B Chat	Zero-shot	0.12	0.55	3.93	0.05	0.35	6.85
Llama 2 70B Chat	CoT	0.08	0.75	8.92	0.03	0.30	11.1
Llama 65B	Few-shot	0.17	0.59	2.95	0.12	0.55	4.05
Llama 65B	CoVe (joint)	0.29	0.41	0.98	0.15	0.30	1.69
Llama 65B	CoVe (two-step)	**0.36**	0.38	0.68	0.21	0.50	0.52
Llama 65B	CoVe (factored)	0.32	0.38	0.79	**0.22**	0.52	1.52

LLM	Method	F1 (↑)	Prec.	Rec.
Llama 2 70B Chat	Zero-shot	0.20	0.13	0.40
Llama 2 70B Chat	CoT	0.17	0.11	0.37
Llama 65B	Few-shot	0.39	0.40	0.38
Llama 65B	CoVe (joint)	0.46	0.50	0.42
Llama 65B	CoVe (factored)	**0.48**	0.50	0.46

LLM	Method	FACTSCORE. (↑)	Avg. # facts
InstructGPT*	Zero-shot	41.1	26.3
ChatGPT*	Zero-shot	58.7	34.7
PerplexityAI*	Retrieval-based	61.6	40.8
Llama 2 70B Chat	Zero-shot	41.3	64.9
Llama 2 70B Chat	CoT	41.1	49.0
Llama 65B	Few-shot	55.9	16.6
Llama 65B	CoVe (joint)	60.8	12.8
Llama 65B	CoVe (factored)	63.7	11.7
Llama 65B	CoVe (factor+revise)	**71.4**	12.3

Experiments

- **Joint, 2-Step, Factored 방법 비교**
  - Joint 방법에 비해 2-Step, Factored 방법이 더 높은 성능을 보임
  - 이는 verification question, verification answer를 생성할 때, baseline response을 보면 안된다는 가설을 뒷받침함

		Wikidata (Easier)			Wiki-Category list (Harder)		
LLM	Method	Prec. (↑)	Pos.	Neg.	Prec. (↑)	Pos.	Neg.
Llama 2 70B Chat	Zero-shot	0.12	0.55	3.93	0.05	0.35	6.85
Llama 2 70B Chat	CoT	0.08	0.75	8.92	0.03	0.30	11.1
Llama 65B	Few-shot	0.17	0.59	2.95	0.12	0.55	4.05
Llama 65B	CoVe (joint)	0.29	0.41	0.98	0.15	0.30	1.69
Llama 65B	CoVe (two-step)	**0.36**	0.38	0.68	0.21	0.50	0.52
Llama 65B	CoVe (factored)	0.32	0.38	0.79	**0.22**	0.52	1.52

LLM	Method	F1 (↑)	Prec.	Rec.
Llama 2 70B Chat	Zero-shot	0.20	0.13	0.40
Llama 2 70B Chat	CoT	0.17	0.11	0.37
Llama 65B	Few-shot	0.39	0.40	0.38
Llama 65B	CoVe (joint)	0.46	0.50	0.42
Llama 65B	CoVe (factored)	**0.48**	0.50	0.46

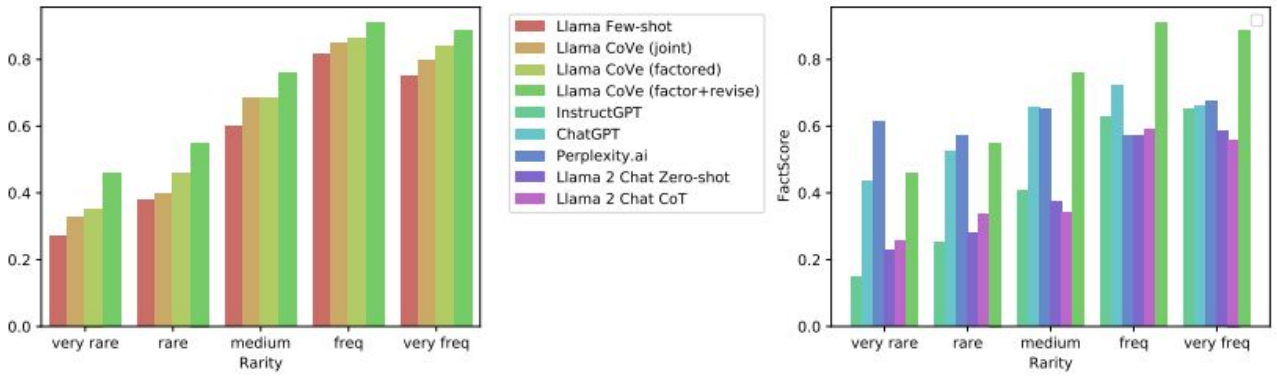
LLM	Method	FACTSCORE. (↑)	Avg. # facts
InstructGPT*	Zero-shot	41.1	26.3
ChatGPT*	Zero-shot	58.7	34.7
PerplexityAI*	Retrieval-based	61.6	40.8
Llama 2 70B Chat	Zero-shot	41.3	64.9
Llama 2 70B Chat	CoT	41.1	49.0
Llama 65B	Few-shot	55.9	16.6
Llama 65B	CoVe (joint)	60.8	12.8
Llama 65B	CoVe (factored)	63.7	11.7
Llama 65B	CoVe (factor+revise)	**71.4**	12.3

Experiments

- **Cove-based Llama가 InstructGPT, ChatGPT, PerplexityAI 보다 뛰어난 성능을 보임**
  - Llama 65B few-shot baseline은 ChatGPT, PerplexityAI 보다 낮은 성능을 보임
  - 하지만 CoVe를 적용함으로써 InstructGPT, ChatGPT, PerplexityAI 보다 높은 성능을 보임

- 하지만 검색이 필수적인 very rare한 fact에 대해서는 여전히 PerplexityAI가 더 높은 성능을 보이고,
- **보다 빈번한 fact 에 대해서는 CoVe가 더 높은 성능을 보임**

LLM	Method	FACTSCORE. (↑)	Avg. # facts
InstructGPT*	Zero-shot	41.1	26.3
ChatGPT*	Zero-shot	58.7	34.7
PerplexityAI*	Retrieval-based	61.6	40.8
Llama 2 70B Chat	Zero-shot	41.3	64.9
Llama 2 70B Chat	CoT	41.1	49.0
Llama 65B	Few-shot	55.9	16.6
Llama 65B	CoVe (joint)	60.8	12.8
Llama 65B	CoVe (factored)	63.7	11.7
Llama 65B	CoVe (factor+revise)	**71.4**	12.3



Conclusion

- LLM에서 스스로 응답하고 수정해 오답을 줄이는 방식인 **CoVe (Chain-of-Verification) 방법을 제안**
- Verification을 간단한 질문으로 세분화하는 것이 verification question에 더 정확하게 답할 수 있음을 보임
- Verification question에 답할 때, 모델이 이전 답변을 참고할 수 없게 하는 **Factored 방법이 반복되는 hallucination을 완화**할 수 있음을 보임
- 답변을 검증하도록 하는 것만으로도 상당한 성능 향상이 가능함을 보임

- 한계점
  - Hallucination을 줄이기는 했으나 완전히 없애지는 못했음
  - 이 연구에서는 사실적 오류만을 hallucination으로 다루고 실험했지만, hallucination은 추론 오류나 주관적 의견에서도 발생할 수 있음
  - 보다 정확한 검증을 위해서는 외부 tool이 필요함

  - 후에, CoVe의 검증 단계에 검색 증강 방법을 사용하는 등 확장될 수 있음

Open Questions

- 왜 Wikidata, Wiki-category list, MultiSpanQA dataset에 대해서는 Factor+revised 방법을 실험하지 않았을까?

- 검증 질문에 대한 검증 답변을 생성하면서도 오류가 생길 수 있을텐데 이런 문제는 어떻게 해결하는게 좋을까?
  - 검색 외에

Thank You

감사합니다.