

ReAct: Synergizing Reasoning and Acting in Language Models

REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS

Shunyu Yao^{*1}, Jeffrey Zhao², Dian Yu², Nan Du², Izhak Shafran², Karthik Narasimhan¹, Yuan Cao²

¹Department of Computer Science, Princeton University

²Google Research, Brain team

¹{shunyuy, karthikn}@princeton.edu

²{jeffreyzhao, dianyu, dunan, izhak, yuancao}@google.com

ABSTRACT

While large language models (LLMs) have demonstrated impressive performance across tasks in language understanding and interactive decision making, their abilities for reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the use of LLMs to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state-of-the-art baselines in addition to improved human interpretability and trustworthiness. Concretely, on question answering (HotpotQA) and fact verification (Fever), ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. Furthermore, on two interactive decision making benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, while being prompted with only one or two in-context examples.

대규모 언어 모델(LLM)들이 언어 이해와 상호작용적 의사결정 분야에서 인상적인 성능을 보여왔지만, 그들의 추론 능력(예: 사고의 연쇄 프롬프팅)과 행동 능력(예: 행동 계획 생성)은 주로 별개의 주제로 연구되어 왔다. 본 논문에서는 LLM을 사용하여 풍부한 추론 과정과 작업 특화된 행동을 통합적으로 생성하는 방법을 탐구한다. 이는 두 요소 사이의 시너지를 높이고, 추론 과정이 모델의 추적, 계획 갱신, 예외 처리를 돕게 한다. 동시에 행동들은 외부 소스(지식베이스나 환경)와의 상호작용을 통해 추가 정보를 얻을 수 있게 한다. 우리는 이 접근

법인 ReAct를 다양한 언어 및 의사결정 작업에 적용하고, 최신 기법들에 비해 향상된 성능을 보여준다. 구체적으로, 질문 답변(HotpotQA)과 사실 검증(Fever) 작업에서 ReAct는 단순한 위키피디아 API와의 상호작용과 인간다운 작업 해결 과정 생성을 통해 사고의 연쇄 추론에서 나타나는 환각과 오류 전파 문제를 극복한다. 또한 두 가지 상호작용적 의사결정 벤치마크(ALFWorld와 WebShop)에서 ReAct는 모방과 강화학습 방법들을 능가하며, 단 한 두 개의 맥락 내 예시만으로 34% 이상의 성능 향상을 달성한다.

ReAct:

1. 정의:

- ReAct는 모델이 아니라 학습 기법 또는 프레임워크이다.
- 이는 대규모 언어 모델(예: GPT)을 사용하여 추론과 행동을 결합하는 방법론이다.

2. GPT/트랜스포머 모델과의 차이:

- GPT나 트랜스포머 모델은 기본적인 언어 모델 구조이다.
- **ReAct는 이러한 모델들을 활용하여 특정한 방식으로 문제를 해결하는 기법**이다.

3. 주요 특징:

- ReAct는 추론(Reasoning)과 행동(Acting)을 번갈아가며 수행하는 구조를 가진다.
- 이는 모델이 생각하고, 행동하고, 관찰하는 과정을 반복하게 한다.

4. 목적:

- ReAct의 목적은 언어 모델의 능력을 확장하여 복잡한 다단계 작업을 수행할 수 있게 하는 것이다.

5. 구현 방식:

- ReAct는 특정한 프롬프트 구조와 fine-tuning을 통해 기존 언어 모델에 구현된다.

6. 활용:

- 질문 답변, 의사결정, 작업 계획 등 다양한 분야에서 활용될 수 있다.

따라서 ReAct는 GPT나 트랜스포머 같은 언어 모델을 더 효과적으로 활용하기 위한 방법론이며, 그 자체로는 독립적인 모델이 아니다. 이는 기존 모델의 능력을 체계적으로 활용하고 확장하는 프레임워크라고 볼 수 있다.

ReAct의 Reasoning과 Acting:

1. Reasoning (추론):

- 이는 모델이 현재 상황을 분석하고 다음 단계를 계획하는 과정이다.
- 주로 'Thought' 단계로 표현되며, 모델이 문제 해결을 위한 생각을 텍스트로 생성한다.
- 예: "이 문제를 해결하려면 먼저 X에 대한 정보가 필요하다."

2. Acting (행동):

- 이는 추론을 바탕으로 실제 행동을 수행하는 과정이다.
- 'Action' 단계로 표현되며, 모델이 외부 환경과 상호작용하기 위한 명령을 생성한다.
- 예: "Search[X]" 또는 "Calculate[Y+Z]"

3. 과정:

- Reasoning과 Acting은 번갈아가며 반복적으로 수행된다.
- 각 Action 후에는 'Observation' 단계가 있어, 행동의 결과를 관찰한다.
- 이 관찰 결과를 바탕으로 다시 Reasoning을 수행하는 식으로 진행된다.

4. 구현:

- 이 과정은 언어 모델에 특별히 설계된 프롬프트를 통해 구현된다.
- 모델은 Thought, Action, Observation의 패턴을 학습하고 이에 따라 출력을 생성한다.

5. 목적:

- 이러한 구조는 모델이 복잡한 문제를 단계적으로 해결하고, 필요한 정보를 동적으로 수집할 수 있게 한다.

이러한 Reasoning과 Acting의 반복은 ReAct가 단순한 텍스트 생성을 넘어 복잡한 문제 해결과 의사결정을 수행할 수 있게 하는 핵심 메커니즘이다.

ReAct = Reason + Act

= 추론 + 실행

즉 ReAct란, 추론의 'CoT'와 실행의 'Action(검색 등)'을 조합하여

답변의 신뢰성과 추론의 능력을 키우고자 의도한 기법이다.

Prompting Method를 아래와 같이 4가지 방법으로 구분할 수 있다.

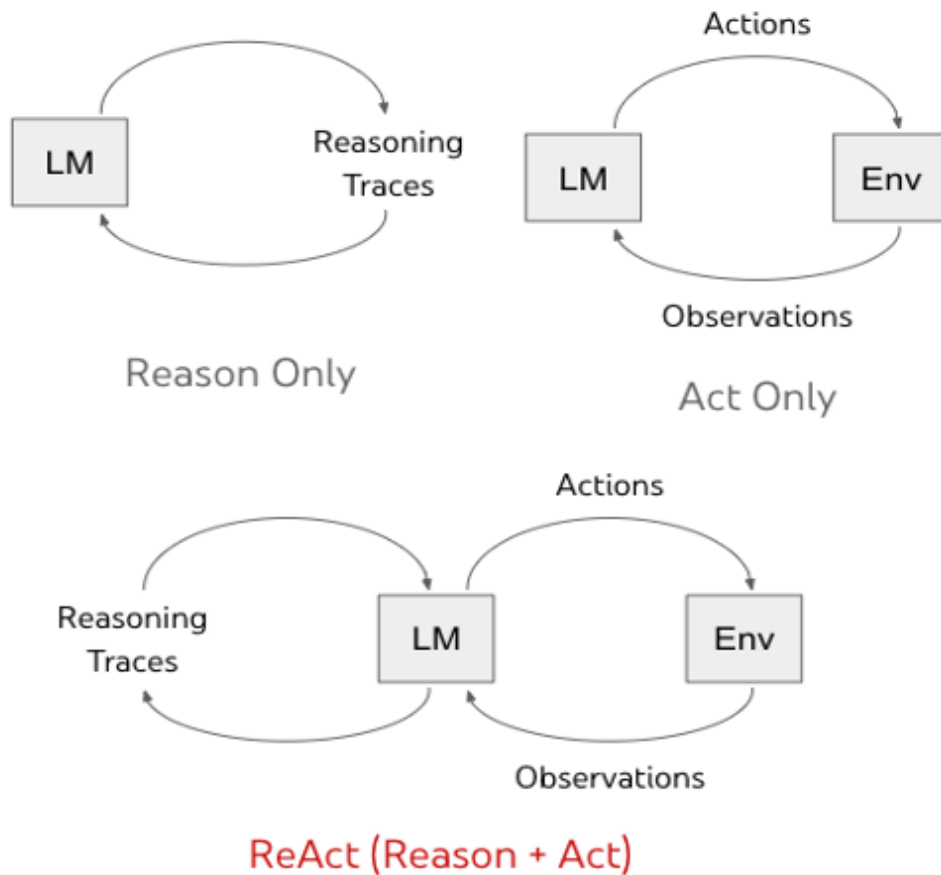
- Standard Prompt

- Reason-only(CoT, Chain of Thought) : 즉, 추론 only
- Act-only : Action과 Observation(관찰) only

(action은 인터넷이나 문서같은 외부 요소에 접촉하여 정보를 수집)

- ReAct : 모두 적용

추론과 Action/observation



Reason Only는 내부의 지식을 활용하고,

Act Only는 외부의 지식(ex. WebSearch)을 활용하는 것이다.

Reason-only 모델은 자체 정보가 부족할 경우, Hallucination에 의한 부정확한 정보를 출력할 수 있고,

Act-only 모델은 추론 능력 부족으로 외부 정보를 기반으로도 최종 답에 이르지 못하고 엉뚱한 대답을 할 수 있다.

문제: "John은 사과 5개를 가지고 있었습니다. 그는 Mary에게 2개를 주었고, 상점에서 3개를 더 샀습니다. John은 지금 몇 개의 사과를 가지고 있나요?"

Act only:

Action: John의 초기 사과 개수는 5개이다.

Action: Mary에게 2개를 주어 3개가 남는다.

Action: 상점에서 3개를 더 사서 총 6개가 된다.

Action: 최종적으로 John은 6개의 사과를 가지고 있다.

Reason only:

Thought: 문제를 해결하기 위해 초기 상태, 변화, 최종 상태를 고려해야 하는 것이다.

Thought: John의 초기 사과 개수에서 Mary에게 준 개수를 빼야 하는 것이다.

Thought: 그 후 상점에서 산 사과 개수를 더해야 하는 것이다.

Thought: 이 과정을 통해 John의 최종 사과 개수를 구할 수 있는 것이다.

ReAct:

Thought: 이 문제를 해결하기 위해 단계별로 접근해야 하는 것이다. 먼저 초기 상태를 파악한 후, 변화를 계산하고 최종 결과를 도출해야 하는 것이다.

Action: 초기 상태 파악은 John의 처음 사과 개수가 5개라는 것이다.

Thought: 이제 John이 Mary에게 준 사과의 수를 빼야 하는 것이다.

Action: 계산 수행은 $5 - 2 = 3$ 이다. John은 Mary에게 사과를 준 후 3개의 사과를 가지고 있는 것이다.

Thought: 다음으로 John이 상점에서 산 사과의 수를 더해야 하는 것이다.

Action: 계산 수행은 $3 + 3 = 6$ 이다. John은 상점에서 사과를 산 후 총 6개의 사과를 가지게 되는 것이다.

Thought: 모든 변화를 고려했으므로 최종 결과를 도출할 수 있는 것이다.

Action: 최종 답변 생성은 John이 현재 6개의 사과를 가지고 있다는 것이다.

이 예시에서 볼 수 있듯이, ReAct 접근 방식은:

1. 각 단계에서 명시적으로 '생각'(Thought)을 표현하는 것이다.
2. 'Action' 단계에서 실제 계산이나 정보 처리를 수행하는 것이다.
3. 문제를 작은 단계로 나누어 해결하는 것이다.
4. 각 단계의 결과를 바탕으로 다음 단계를 계획하는 것이다.

이러한 방식은 모델의 추론 과정을 더 명확하게 보여주는 것이다. **복잡한 문제를 체계적으로 해결**할 수 있게 해주는 것이다. 기존의 모델들이 이러한 과정을 내부적으로 수행하는 반면,

ReAct는 이를 명시적으로 보여줌으로써 모델의 '사고' 과정을 관찰하고 평가할 수 있게 하는 것이다.

Combining Internal and External Knowledge As will be detail in Section 3.3, we observe that the problem solving process demonstrated by ReAct is more factual and grounded, whereas CoT is more accurate in formulating reasoning structure but can easily suffer from hallucinated facts or thoughts. We therefore propose to incorporate ReAct and CoT-SC, and let the model decide when to switch to the other method based on the following heuristics: A) **ReAct → CoT-SC**: when ReAct fails to return an answer within given steps, back off to CoT-SC. We set 7 and 5 steps for HotpotQA and FEVER respectively as we find more steps will not improve ReAct performance³. B) **CoT-SC → ReAct**: when the majority answer among n CoT-SC samples occurs less than $n/2$ times (i.e. internal knowledge might not support the task confidently), back off to ReAct.

ReAct는 내부의 지식과 외부의 자원을 모두 활용하는 방법

추론(CoT) 방식과 Action/Observation들을 기반으로 해석 가능하고 사실에 기반한 Trajectory로 답변을 찾아간다.(거시적인 관점에서 볼 때, ReAct는 내부의 지식과 외부의 자원을 활용하는 RAG의 상위 개념으로도 볼 수 있을 듯하다)

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control ...

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ...

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software ...

Act 4: Finish[yes] ✗

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

Thought 3: Front Row is not found. I need to search Front Row (software) ...
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software ...

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓

(2) AlfWorld

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1. Your task is to: Put some pepper shaker on a drawer.

(2a) Act-Only

Act 1: Go to drawer 1
Obs 1: The drawer 1 is closed.

Act 2: Open drawer 1
Obs 2: You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.

Act 4: Take peppershaker 1 from sinkbasin 1
Obs 4: Nothing happens.

Act 5: Take peppershaker 1 from sinkbasin 1
Obs 5: Nothing happens. ✗

(2b) ReAct (Reason + Act)

Act 1: Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]
Act 2: Go to cabinet 1
Obs 2: On the cabinet 1, you see a vase 2.
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

Act 6: Go to countertop 3
Obs 6: On the countertop 3, you see a apple 1, a bread 1, a pepper shaker 1, and a vase 3.

Act 7: Take pepper shaker 1 from countertop 3
Obs 7: You pick up the pepper shaker 1 from the countertop 3.

Act 8: Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1.]
Act 9: Go to drawer 1
Obs 9: Drawer 1 is closed.

Act 10: Open drawer 1
Obs 10: You open Drawer 1 ...

Act 11: Put pepper shaker 1 in/on drawer 1
Obs 11: You put pepper shaker 1 in/on the drawer 1. ✓

CoT-SC:

1. CoT는 "Chain of Thought"의 약자이다. 이는 대규모 언어 모델이 복잡한 문제를 해결할 때 단계적인 추론 과정을 거치는 방법이다.
2. SC는 "Self-Consistency"의 약자.
3. CoT-SC는 Chain of Thought와 Self-Consistency를 결합한 방법론이다. 이 방법은 여러 개의 추론 경로를 생성하고, 그 중에서 가장 일관성 있는 답변을 선택하는 접근 방식이다.
4. 이 방법은 단일 추론 경로에 의존하는 것보다 더 신뢰할 수 있는 결과를 얻을 수 있다는 장점이 있다.
5. CoT-SC는 모델이 스스로 여러 가지 가능한 추론 과정을 탐색하고, 그 중에서 가장 타당한 결론을 도출하는 방식으로 작동한다.

따라서 CoT-SC는 복잡한 추론 과정을 여러 번 수행하고 그 결과들 사이의 일관성을 확인함으로써 더 정확한 답변을 찾아내는 방법이다.

ReAct 진행 과정은 1) ReAct \Rightarrow CoT-SC과 2) CoT-SC \Rightarrow ReAct로 나눌 수 있음.

CoT-SC \Rightarrow ReAct 방식의 경우

1. 답변을 위한 추론(Thought)의 과정을 거치고,
2. 외부 지식에 기반한 Action을 실행하고,
3. 외부 지식에 대한 Observation(관찰)을 진행한다.
4. 그리고 다시 위의 1, 2, 3 과정을 반복해가며 답변에 도달한다.

ReAct \Rightarrow CoT-SC 방식은 순서가 바뀌어 Action-Observation-CoT 순서이다.

(CoT-SC는 CoT Self-Consistency를 말한다.

Self-Consistency prompt는 CoT(Chain of Thought)의 연장선으로,

하나의 답변이 아니라 여러 개의 사고 과정을 생성하고 그 중 다수결로 최종 답변을 결정하는 방식이다.

즉, *퓨샷 생각의 사슬(few-shot CoT)*을 통해 여러 번의 다양한 추론 경로를 생성하고, 가장 일관된 답을 선택하는 것.

이때 일관된 답이라는 것은 가장 많이 나온 답을 택하거나, 가장 자주 나온 추론 경로를 택하는 것을 말한다.)

이와 같이 ReAct 프롬프트는 문제 해결을 위한 일련의 'trajectory'들로 구성된다고 볼 수 있다.

추론과 외부 지식 기반의 조합인 ReAct 방식은 높은 성능을 보였고, 논문 발표 당시 몇몇 Task에서는 SOTA를 달성했었다.

※ *Trajectory : Reasoning/Thoughts, Actions, Observation* 등 기본 프롬프트 질문에 이어지는 과정을 나열한 것으로, ReAct Prompting에서 few shot example로 제공하거나,

fine-tuning 진행 후 최종 질문을 하였을 때, 학습했던 Trajectory에 따라 추론-act-obs를 실행하도록 지시한다.

1. ReAct \Rightarrow CoT-SC:

- 이 방식은 먼저 ReAct의 Action-Observation 단계를 수행한 후, CoT-SC의 추론 과정을 진행하는 것이다.
- 즉, 외부 정보를 먼저 수집하고 관찰한 다음, 그 정보를 바탕으로 여러 추론 경로를 생성하고 가장 일관된 결론을 도출하는 것이다.

2. CoT-SC \Rightarrow ReAct:

- 이 방식은 먼저 CoT-SC의 추론 과정을 수행한 후, ReAct의 Action-Observation 단계를 진행하는 것이다.
- 즉, 여러 추론 경로를 먼저 생성하고 가장 유력한 추론을 선택한 다음, 그에 따라 외부 환경과 상호작용하며 정보를 수집하고 관찰하는 것이다.

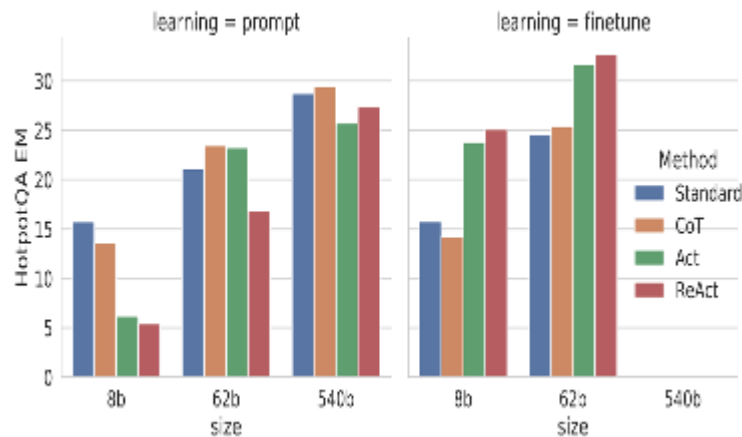
이 두 방식의 차이점은 추론과 행동의 순서이다. 어떤 문제에는 정보를 먼저 수집하는 것이 유리할 수 있고, 다른 문제에는 추론을 먼저 하는 것이 효과적일 수 있다. 따라서 문제의 특성에 따라 적절한 접근 방식을 선택하는 것이 ReAct 프레임워크의 유연성이다.

Prompt Method ^a	HotpotQA (EM)	Fever (Acc)
Standard	28.7	57.1
CoT (Wei et al., 2022)	29.4	56.3
CoT-SC (Wang et al., 2022a)	33.4	60.4
Act	25.7	58.9
ReAct	27.4	60.9
CoT-SC \rightarrow ReAct	34.2	64.6
ReAct \rightarrow CoT-SC	35.1	62.0
Supervised SoTA^b	67.5	89.5

Table 1: PaLM-540B prompting results on HotpotQA and Fever.

눈에 띄는 점으로는 ReAct와 Act 방법은 더 많은 훈련 단계(혹은 더 많은 훈련 데이터)에서 성능 향상이 확인되지만, Standard Prompt와 CoT 방법은 Fine-tuning 이후에 빠르게 저하된다는 것이다.

외부 지식 기반(RAG)의 강인성이 여기서도 확인되는 것 같다.



For all finetuning we use a batch size of 64. On PaLM-8B, we finetune ReAct and Act methods for 4,000 steps and Standard and CoT methods for 2,000 steps. On PaLM-62B, we finetune ReAct and Act methods for 4,000 steps and Standard and CoT methods for 1,000 steps. We find ReAct and Act methods generally benefit from more training steps (and more training data), while Standard and CoT methods degrade soon after finetuning.

fine-tuning은 3,000개의 Trajectory를 사용하여 작은 언어 모델(PaLM-8/62B)에 학습을 실시하였는데,

Input용 질문/주장에 조건을 걸어, Trajectory(모든 생각, 행동, 관측)를 단계적으로 분리 실행할 수 있도록 했다.

출처

- [1] Yao, Shunyu, et al. "REACT: Synergizing Reasoning and Acting in Language Models." *arXiv preprint arXiv:2210.03629* (2022).
- [2] Bbijun. "논문 정리: REACT - Synergizing Reasoning and Acting in Language Models." Velog, <https://velog.io/@bbijun/논문-정리-REACT-SYNERGIZING-REASONING-AND-ACTING-IN-LANGUAGE-MODELS>
- [3] Claude. (2024). Claude (Oct 15 Claude 3.5 sonnet version) [Large language model]. <https://claude.ai/>.