GPT 모델과 Chain-of-thought 프롬프팅을 활용한 SayCan tabletop 환경에서 로봇작업 계획수립 성능 분석

옹효빈[1], *장민수[2]
[1]과학기술연합대학원대학교 한국전자통신연구원스쿨
[2]과학기술연합대학원대학교 한국전자통신연구원
e-mail : *ohnghb@etri.re.kr*, *minsu@etri.re.kr*

Performance analysis of robot task planning based on GPT model and Chain-of-thought prompting in the SayCan tabletop environment

Hyo-Bin Ong, *Min-Su Jang
Electronics and Telecommunications Research Institute School
University of Science and Technology, Korea

Abstract

In this paper, we investigate the impact of Chain-of-thought prompting on the SayCan tabletop manipulation task using a large language model. We show that chain-of-thought prompting leads to a 6.25% performance improvement over standard prompting using GPT-3 text-ada-001 model in the SayCan tabletop environment. These results demonstrate the potential of prompting strategies to optimize the performance of robotic task planning in a variety of scenarios in the SayCan tabletop environment.

I. 서론

최근 몇 년 동안 사전 학습된 대규모 언어 모델(LLM)이 개발되면서 자연어 처리 분야에 혁신을 일으켰다. LLM이란 대규모 데이터셋에서 얻은 지식을 기반으로 텍스트와 다양한 콘텐츠를 인식하고 요약, 번역, 예측, 생성할 수 있는 딥러닝 알고리즘이다.

OpenAI는 1,750억개의 매개변수로 구동되는 GPT-3[1]를 소개했고, Google은 1,370억개의 매개변수로 구동되는 LaMDA[2], 5,400억개의 매개변수로 구동되는 PaLM[3]을 소개했다.

LLM은 NLP(Natural Language Processing)분야에 한정되지 않고, 다양한 분야에서 새로운 가능성을 열고 있다. 그 중, 로보틱스 분야에 적용되어 인간과 로봇이 상호작용하는데 크게 이바지하고있다. Robotics at Google과 Everyday Robots가 공동 개발한 로보틱스 알고리즘인 PaLM-SayCan이 소개되었는데, 이는 실제 로봇에 LLM을 연결하여 추상적인 자연어 명령을 수행한다[4]. Microsoft는 자연어 명령을 로봇이 실행할 수 있는 작업으로 변환하기 위해 few-shot으로 ChatGPT를 사용하는 구체적인 예시를 소개하는 논문을 발표했다[5].

다양한 분야에 적용 할 수 있는 LLM은 프롬프트 인터페이스를 통해 사용 가능하며, 프롬프트 설계에 따라 LLM의 성능은 크게 영향을 받는다. 따라서 프롬프트는 LLM에서 중요한 요소 중 하나로 간주된다.

대표적인 프롬프팅 기법으로는 Chain-of-thought 프롬프팅[6], Least-to-most 프롬프팅[7] 등이 있다. Chain-of-thought 프롬프팅은 프롬프트 중간에

추론할 수 있는 문장을 추가하고, Least-to-Most 프롬프팅은 하위문제로 나누어 순차적으로 복잡한 문제를 해결할 수 있도록 프롬프팅한다.

Chain-of-thought 프롬프팅은 충분한 규모의 LLM 모델과 사용하면 성능에 긍정적인 영향을 미친다. 비교적 규모가 작은 LLM 모델에 사용해도 성능 향상이 있으나, 규모가 충분한 모델에 비하면 그 차이는 미미하다.

Least-to-most 프롬프팅은 GSM8K, DROP, SCAN 같은 특정 벤치마크에서 Chain-of-thought 프롬프팅에 비해 높은 성능을 나타낸다.

본 논문에서는 제한된 환경에서 사용할 수 있는 작은 규모의 LLM에 Chain-of-thought 프롬프팅 기법을 적용함으로써 규모가 큰 LLM이 Standard 프롬프트를 사용하는 것만큼의 성능에 도달할 수 있을지 실험한다. 또한, 프롬프트 내의 문장 수를 기존보다 줄이며 LLM이 생성한 계획수립의 성능에 어느정도 영향을 미치는지 분석하고 결과를 제시한다.

II. 본론

2.1 SayCan

"로봇에게 "음료를 쏟았는데 도와줄래?" 와 같은 복잡한 작업을 수행하도록 하려면 기존엔 사용자가 명령을 단계별로 나누어 수행했다. 언어모델을 사용하여 명령을 하위 단계로 나눌 수 있지만, 로봇에게 주어진 능력, 로봇의 현재 상태, 로봇이 처한 환경에 대한 지식이 없다면 효과적으로 수행할 수 없다. 앞선 질문에 대해 언어모델이 "진공 청소기를 사용해 보세요." 라고 응답할 수는 있겠지만, 이런 응답은 합리적일 수는 있으나 현재 환경에서 로봇의 능력으로는 실행할 수 없다."[4]

이러한 한계점을 극복하기 위해 Robotics at Google, Everyday Robots은 SayCan을 발표했다. SayCan은 실제 물리적인 환경 기반의 작업에서 LLM 내의 지식을 추출하여 활용한다. SayCan에서의 Say는 LLM의 높은 수준의 목표를 위해 도움이 되는 동작(action)을 결정하기 위한 task grounding을 제공한다. Can은 학습된 어포던스 함수(Affordance function)를 말하는데, 이 함수는 작업을 수행 할 계획에 따라 실행이 가능한 것을 결정하기 위한 world ground를 제공한다.

즉, 사용자의 명령에 대해 LLM이 판단한 주어진 명령에 대해 수행할 작업의 가능성을 점수화한 것, 주어진 환경에서 로봇이 작업 수행을 성공할 가능성을 정량화 하는 함수인 어포던스 함수에서 출력된 점수를 수행할 작업의 가능성에 가중치를 부여하여 명령을 수행한다.

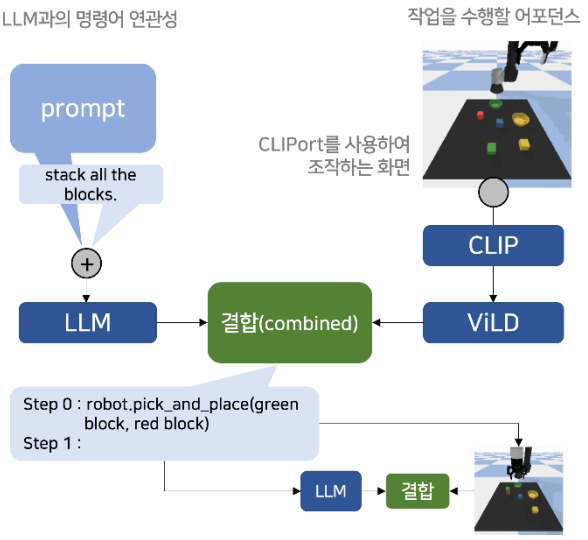

그림 1. SayCan tabletop 동작 구조도

SayCan Tabletop 환경은 Google에서 LLM 기반 로봇 작업 계획 기능을 시험 평가하기 위해 개발한 시뮬레이션 환경으로 테이블 위에 블럭과 그릇을 임의로 배치하고 로봇팔을 조작하여 블럭을 들어 옮기는 조작 작업을 수행할 수 있다. 이 환경의 작동 구조는 그림 1과 같다.

환경 안에는 테이블 위에 무작위로 생성한 색상 블록과 그릇들이 배치되고 UR5를 모사한 로봇팔을 제어하여 블럭을 조작할 수 있다. 블럭을 들어 옮겨 놓는 Pick and Place 기능은 CLIPort[8]를 이용하여 수행하도록 구현되어 있다. SayCan은 어포던스를 계산하는 가치 함수(Value Function)를 통해 LLM이 생성한 여러 작업 행위 후보들 중 주어진 환경 상황에서 수행할 수 있는 행위를 선별함으로써 작업 계획 성공율을 높인다. SayCan Tabletop 환경에서는 ViLD[9] 객체 검출기를 이용하여 테이블 위에 조작 대상 사물이 존재하는지 여부를 판단하는 기능을 통해 가치 함수의 역할을 수행하도록 만들어져 있다.

언어모델은 GPT-3를 사용하고, 코드 구조를 출력하는 LLM의 기능을 활용하여 "pick_and_place(pick, place)"의 형식으로 step이 출력된다.

2.2 Chain-of-thought 프롬프팅

NLP(Natural Language Processing) 분야는 최근 LLM에 의해 혁신적으로 변화하고 있고, 언어모델의 규모를 증가시키면 성능을 향상시킬 수 있는 것

외에도 다양한 이점을 얻을 수 있다. 그러나 까다로운 작업인 산술 계산 (arithmetic), 상식 추론 (commonsense), 기호 추론 (symbolic reasoning)에서 높은 성능을 달성하기 위해서는 모델 크기를 증가시키는 것 만으로는 충분하지 않은 것으로 입증되었다. 그 한계점을 개선하기 위해 Google Research에서 제안한 프롬프팅 기법이 Chain-of-thought 프롬프팅이다.

가장 기본적인 프롬프트인 스탠다드 프롬프트는 모델이 해당 작업을 수행하는 데 필요한 최소한의 정보만을 입력으로부터 추출하고, 이를 기반으로 출력을 생성하는 것을 말한다. 스탠다드 프롬프트 말고도 다양한 프롬프팅 기법이 있다. 그 중 하나가 Chain-of-thought 프롬프팅이고, 이 기법은 중간에 추론 단계를 생성함으로써 복잡한 문제를 개별적으로 해결할 수 있도록 중간 단계로 분해하여 추론한다.

SayCan 논문에 따르면 스탠다드 프롬프트를 사용하면 부정(negation)을 포함하는 명령(e.g., 사과가 아닌 간식을 가져와줘)을 처리하지 못한다는 한계점이 있었지만, 이 프롬프팅 기법을 사용하여 한계점을 개선했다고 한다.

Ⅲ. 구현

명령과 지정할 물체 목록
put all the blocks in different corners. objects = [red block, yellow block, green block, blue block, red bowl]
move the block to the bowl. objects = [red block, green bowl]
put any blocks on their matched colored bowls objects = [yellow block, green block, blue block, yellow bowl, green bowl, blue bowl]
put all the blocks in the green bowl. objects = [yellow block, green block, blue block, yellow bowl, green bowl]
stack all the blocks. objects = [yellow block, blue block, red block, blue bowl, red bowl]
make the highest block stack. objects = [yellow block, blue block, red block, blue bowl, red bowl]
stack all the blocks. objects = [green block, blue block, red block, yellow bowl, green bowl]
put the block in all the corners. objects = [red block, blue bowl, green bowl]

표 1. 작업을 수행할 명령 및 지정할 물체 목록

SayCan tabletop 환경에서 명령을 내렸을 때, 작업을 수행할 명령과 물체는 표 1에 명시된 8가지이며 오픈소스에서 사용하던 명령을 그대로 사용했다.

실험에 사용한 LLM은 GPT-3의 text-ada-001 모델, GPT-3.5의 text-davinci-002 모델을 사용하였다. text-ada-001은 매우 간단한 작업을 수행할 수 있으며, 일반적으로 GPT-3 시리즈 중 가장 빠른 모델이며 비용이 가장 저렴한 모델이다. text-davinci-002 모델은 자연어 또는 코드를 이해하고 생성할 수 있는 GPT-3.5 모델 중 한 가지이다.

SayCan tabletop 환경에서 명령을 내렸을 때, 작업을 수행할 명령과 물체는 표 1에 명시된 8가지이며 오픈소스에서 사용하던 명령을 사용했다.
실험에 사용한 LLM은 GPT-3의 text-ada-001 모델, GPT-3.5의 text-davinci-002 모델을 사용했다.

text-ada-001 모델은 매우 간단한 작업을 수행할 수 있으며, 일반적으로 GPT-3 시리즈 중 가장 속도가 빠르고 비용이 가장 저렴한 모델이다. text-davinci-002 모델은 자연어 또는 코드를 이해하고 생성할 수 있는 GPT-3.5 모델 중 한 가지이다.

Standard prompt

objects = [red block, yellow block, blue block, green bowl]
# move all the blocks to the top left corner. 	
robot.pick_and_place(blue block, top left corner)
robot.pick_and_place(red block, top left corner)
robot.pick_and_place(yellow block, top left corner)
done()

표 2. SayCan tabletop에 사용된 스탠다드 프롬프팅 예시

Chain-of-thought prompting
objects = [red block, yellow block, blue block, green bowl] # move all the blocks to the top left corner. **explanation : move red block, yellow block, blue block to the top left corner.** robot.pick_and_place(blue block, top left corner) robot.pick_and_place(red block, top left corner) robot.pick_and_place(yellow block, top left corner) done()

표 3. SayCan tabletop에 사용된 Chain-of-thought

프롬프팅 예시[1]

표 2와 표 3을 보면 두 프롬프트 간에 explanation의 유무에 따라 차이가 있음을 알 수 있다. SayCan tabletop 프롬프트는 LLM이 코드를 출력할 수 있도록 설계되었는데, object는 색상 블록 및 그릇 세트를 지정해주는 변수이고, #으로 시작되는 문장은 수행해야 할 명령이다. pick_and_place 함수는 pick과 place 파라미터로 구성되어 있고, pick에 있는 변수를 place에 있는 변수에 가져다 놓는 작업을 수행한다.

스탠다드 프롬프트는 테이블 위에 놓인 사물 목록, 로봇 작업 명령, 정답 계획으로 이루어져 있다. 반면, Chain-of-Thought 프롬프팅은 작업 계획을 산출하게 된 논리를 기술한 explanation 항목을 추가로 포함한다.

SayCan 논문에서는 성능 평가 지표로써 계획 성공률, 실행 성공률 두가지를 설정하여 측정한다. 계획 성공률은 실제로 명령을 성공적으로 수행한 여부와 관계없이 LLM 모델이 선택한 작업이 적합한지를 측정한다. 실행 성공률은 전체 SayCan 시스템이 실제로 원하는 명령을 성공적으로 수행하는지 여부를 측정한다. SayCan은 실제 로봇으로 실험을 했는데, 로봇이 명령을 수행할 때 계획성공률은 높을지라도 실행성공률은 낮을 수 있다.

하지만, SayCan tabletop 환경은 이미 블록과 그릇 세트가 명령에 의해 구현되어 있기 때문에, LLM 모델이 선택한 작업을 그대로 실행한다. SayCan tabletop 환경에서는 계획성공률과 실행성공률을 통합하여 계획수립 정확도라고 성능지표를 설정한다.

정답 계획	objects = [green bowl, yellow bowl, green block, blue block, red block] # stack all the blocks. Step 0: robot.pick_and_place(green block, red block) Step 1: robot.pick_and_place(blue block, green block)
생성 계획	objects = [green bowl, yellow bowl, green block, blue block, red block] # stack all the blocks. Step 0: robot.pick_and_place(green block, red block)

표 4. 계획을 일부분만 수립한 경우

표 4는 LLM이 생성한 작업 계획의 일례이다. "Stack all the blocks" 명령에 대한 정답 계획과 비교해 보면 LLM 생성 계획은 정답 계획을 구성하는 2단계에 걸친 작업 중 1단계 작업만 정확하게 생성했음을 볼 수 있다. 이와 같이 LLM이 생성한 계획이 정답 계획과 일부분만 일치하는 경우 일치한 부분의 비율을 계산하여 계획수립 정확도를 계산한다. 표 4의 경우 계획수립 정확도는 50%이다. LLM이 정답 계획과 완전히 일치하는 계획을 생성한 경우 100%, 작업 계획을 전혀 생성하지 못한 경우 0%를 부여한다. 본 실험에서는 표 1에 제시한 8개 명령 각각에 대해 LLM이 생성한 작업 계획에 정확도를 계산하여 부여한 후 전체 평균을 계산하여 작업계획 정확도를 계산하였다.

	text-davinci-002		text-ada-001	
N-shot	4-shot	6-shot	4-shot	6-shot
Standard prompting	100	100	7.5	20
Chain-of-thought prompting	100	100	12.5	26.25

표 5. GPT 모델과 프롬프팅 기법에 대한 계획 수립 정확도(%)

N-shot은 프롬프트에 포함한 작업 수행 예제의 개수를 나타내며, 4-shot은 4개의 프롬프트, 6-shot은 기존의 6개의 프롬프트를 뜻한다. 표 5를 보면 text-ada-001 모델에 비해 text-davinci-002 모델의 성능이 월등히 높은 것을 알 수 있고, 본 실험에서 text-davinci-002 모델을 사용하니 모든 명령을 완료하는 결과를 보였다.

6-shot 기준으로 두 모델에 대하여 각각의 프롬프트의 계획 수립 정확도는 각 80%, 73.75%의 차이로 text-davinci-002 모델의 성능이 뛰어난 것을 볼 수 있다.

text-ada-001 모델을 사용하여 6-shot으로 진행한 결과는 스탠다드 프롬프트보다 Chain-of-thought 프롬프팅 기법이 6.25%의 성능 향상이 된 것을 볼 수 있다. 스탠다드 프롬프트를 사용한 경우 9개의 명령 중에서 1개를 완전히 수립했고, 2개는 부분적으로 수립하여 20%의 계획 수립 정확도를 보인다. Chain-of-thought 프롬프팅 기법을 사용한 경우 6개의 명령을 부분적으로 수립하여 26.25%의

[1] 전체 프롬프트는 https://github.com/ohnghb99/tabletop-SayCan에 있음

계획 수립 정확도를 보인다.

동일한 LLM 모델을 사용하여 4-shot으로 진행한 결과는 Chain-of-thought 프롬프팅 기법의 성능이 5%의 성능 향상이 있었다. 스탠다드 프롬프트를 사용한 경우 9개의 명령 중에서 2개만 부분적으로 수행하여 7.5%의 계획 수립 정확도를 보인다. Chain-of-thought 프롬프팅은 1개를 완전히 수립하여 12.5%의 성공률을 보인다.

Ⅳ. 결론 및 향후 연구 방향

본 논문에서는 Chain-of-thought 프롬프팅을 사용하여 SayCan tabletop 환경에서 기존의 프롬프트를 사용했을 때보다 모델의 성능을 향상시킬 수 있는지 비교하는 실험을 했다. 실험에 사용된 SayCan tabletop 환경의 기본 프롬프트를 보면 마치 코드를 설계 해 놓은 것과 parkhj1813같은데, 이런 프롬프트에서도 Chain-of-thought 프롬프팅 기법을 적용하여 text-ada-001 모델의 성능을 향상시킬 수 있는 결과를 보인다. 또한, 프롬프트 내의 문장 개수가 모델의 성능에 영향을 미치는 결과를 보인다.

결과적으로, 코드 설계를 출력으로 하는 작업에도 Chain-of-thought 프롬프팅 기법이 효과가 있다는 것을 알 수 있다. 추가적으로, 본래 모델의 성능보다 향상된 성능을 사용하기 위해서는 창의적인 프롬프팅 설계가 필요하며 프롬프트의 개수 또한 중요하다는 것을 알 수 있다.

향후 실제 로봇과 LLM을 결합하여 인간-로봇 상호작용을 통한 로봇 지능 수준 향상 연구를 진행할 때, SayCan tabletop 환경에서 사용했던 로봇의 어포던스를 활용할 수 있을 것으로 보인다. 또한, 본 실험에서 프롬프팅 기법과 상관없이 성능이 월등했던 text-davinci-002 모델은 다른 프롬프팅 기법을 사용하여 모델의 성능을 향상시킬 수 있을 것이라고 기대한다.

사사문

본 논문은 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2020-0-00951. 스스로 불확실성을 자각하며 질문하면서 성장하는 에이전트 기술 개발)

참고문헌

[1] T. Brown, et al., “Language models are few-shot learners”, arXiv:2005.14165, NeurIPS 2020.

[2] Thoppilan, R, et al., "Lamda: Language models for dialog applications." arXiv preprint arXiv:2201.08239, 2022.

[3] A. Chowdhery, et al., “Palm: Scaling language modeling with pathways”, arXiv preprint arXiv:2204.02311, 2022.

[4] M. Ahn, et al., “Do as I can, not as I say: Grounding language in robotic affordance”, arXiv preprint arXiv:2204.01691, 2022.

[5] Wake, N, et al., "ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application." arXiv preprint arXiv:2304.03893, 2023.

[6] J. Wei, et al., “Chain of thought prompting elicits reasoning in large language models”, arXiv preprint arXiv:2201.11903, 2022.

[7] Zhou, D, et al., "Least-to-most prompting enables complex reasoning in large language models." arXiv preprint arXiv:2205.10625, 2022.

[8] M. Shridhar, et al., “Mohit ShridharCliport: What and where pathways for robotic manipulation”, arXiv:2109.12098, CoRL 2021.

[9] X. Gu, et al., “Open-vocabulary object detection via vision and language knowledge distillation”, arXiv preprint arXiv:2104.13921, 2021.