ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 30, No. 2, pp. 63-69, 2024. 2
https://doi.org/10.5626/KTCP.2024.30.2.063

ARC 문제 해결을 위한 프롬프트 엔지니어링의 가능성

(The Possibility of Prompt Engineering for ARC Problem Solving)

심 우 창 [†] (Woochang Sim)　진 혜 빈 [††] (Hyebin Jin)　김 세 진 [†††] (Sejin Kim)　김 선 동 [††††] (Sundong Kim)

요 약 ARC(Abstraction and Reasoning Corpus)[1]는 범용인공지능을 평가하는 중요한 벤치마크 데이터셋이다. 하지만 ARC 벤치마크 데이터셋에서 좋은 성능을 보여주는 연구가 지금까지 나오지 못하고 있다. 이 와중에 대규모 언어 모델이 다양한 분야에서 좋은 성능을 보여주고 있고 이를 프롬프트 엔지니어링으로 활용하는 방식이 주목받고 있다. 그래서 본 연구에서는 ARC 문제 해결을 위한 프롬프트 엔지니어링의 가능성을 확인해보는 것이다. 이를 위해, 페르소나 프롬프트 및 생각의 사슬 등의 다양한 프롬프트를 활용하여 실험 및 심층 분석을 진행하였다. 실험 결과, 객체를 활용하는 문제에 대해서 어려움을 보이는 것을 확인할 수 있었다. 그러나 단순한 패턴 문제 및 객체를 찾는 문제에 대해서는 잘 푸는 것을 알 수 있었으며 적절한 프롬프트 엔지니어링 방법을 적용했을 때 정확도가 향상되었다. 이를 통해 프롬프트 엔지니어링의 가능성을 확인해 볼 수 있었다.

키워드: ARC, 범용인공지능, 대규모 언어 모델, 프롬프트 엔지니어링

Abstract The Abstraction and Reasoning Corpus (ARC) [1] is an important benchmark dataset for evaluating artificial general intelligence. However, there is a lack of research showing good performance on the ARC benchmark dataset. Large-scale language models are performing well in a variety of domains, and their use in prompt engineering is gaining traction. In this study, we investigated the possibility of prompt engineering for ARC problem solving. We conducted experiments using various prompts such as persona prompts and chain of thought prompts and in-depth analysis. In our experiments, we found that they had difficulty with problems involving objects. However, they were able to solve simple pattern problems and object finding problems well, and their accuracy improved when we applied appropriate prompt engineering methods. This shows the promise of prompt engineering.

Keywords: ARC, artificial general intelligence, large language model, prompt engineering

· 이 논문은 과학기술정보통신부와 광주과학기술원의 재원으로 한국연구재단과 정보통신기획평가원, 그리고 AI기반 융합인재 양성 지원사업의 지원을 받아 수행된 연구임(RS-2023-00240062, RS-2023-00216011, 2019-0-01842).
· 이 논문은 2023 한국컴퓨터종합학술대회에서 'ARC 문제 해결을 위한 프롬프트 엔지니어링의 가능성'의 제목으로 발표된 논문을 확장한 것임.

† 학생회원 : 광주과학기술원 AI 대학원 대학원생
woochang@gm.gist.ac.kr
†† 학생회원 : 광주과학기술원 전기전자컴퓨터 학생
hyebiness01@gm.gist.ac.kr
††† 비 회 원 : 광주과학기술원 AI 대학원 박사후 연구원
sejinkim@gist.ac.kr
†††† 비 회 원 : 광주과학기술원 AI 대학원 교수(GIST)
sundong@gist.ac.kr
(Corresponding author임)

논문접수 : 2023년 8월 21일
(Received 21 August 2023)
논문수정 : 2023년 11월 4일
(Revised 4 November 2023)
심사완료 : 2023년 11월 15일
(Accepted 15 November 2023)

정보과학회 컴퓨팅의 실제 논문지 제30권 제2호(2024. 2)

1. 서 론

인공지능 기술은 특정 도메인에 적합한 모델을 개발하는 방향으로 발전해 왔다. 이로 인해 한 번 만든 모델이 범용적으로 사용될 수 없다는 단점이 있었다. 만약 범용인공지능이 개발된다면 이러한 단점을 극복하여 비용측면에서 큰 효과를 불러올 수 있다. 더 나아가 추상화와 추론을 할 수 있는 범용인공지능은 인공지능 기술의 새로운 혁신을 불러올 것으로 보인다.

기존의 ARC 벤치마크를 활용한 범용인공지능 연구들은 도메인 특화 언어(Domain Specific Language, 이하 DSL)를 적용하는 연구[2] 위주로 진행되었다. 한편, 최근 방대한 양의 데이터를 이용해 다양한 정보를 미리 학습한 대규모 언어 모델(Large Language Model)이 많은 분야에서 활용되고 있다. 이러한 대규모 언어 모델을 활용하는 방식인 프롬프트 엔지니어링은 다양한 도메인에 적용[3] 및 활용되고 있다. 본 논문은 기존 연구들과 다르게 프롬프트 엔지니어링 방식을 ARC 벤치마크에 적용해보고 그 가능성을 확인하고자 한다.

1.1 ARC 벤치마크 데이터셋

ARC 벤치마크 데이터셋은 인공지능 모델의 추상적인 사고와 추론 사고를 평가하기 위해 만들어졌다[1]. ARC 벤치마크 데이터셋에는 그림 1과 같이 다양한 패턴의 여러 문제가 존재하며, 문제 당 2~5개 정도의 예제와 하나의 입력값이 제공된다. 각 예제는 2차원 배열 형태인 입력값과 출력값의 쌍으로 이뤄져 있다. 모델은 주어진 예제들을 통해 해당 문제에서의 패턴을 파악하고, 맞춰야 할 입력값에 대한 적절한 출력값을 추론해야 한다. 그림 1은 배열 형태인 입력값과 출력값을 시각화한 ARC 문제에 대한 예제들이다.

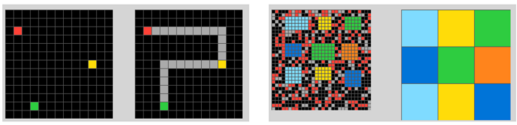

그림 1 ARC 문제 예제들의 예시
Fig. 1 Examples of ARC problem

1.2 프롬프트 엔지니어링

프롬프트 엔지니어링이란 주로 자연어 처리 분야에 해당하는 인공지능의 한 개념으로, 인공지능이 역량을 발휘할 수 있도록 적합한 지시어를 주는 역할을 한다.

자연어 처리 분야에서 GPT 계열의 모델은 문장 생성 부분에서 뛰어난 성능을 보였다. 그러나 해당 모델을 특정 분야의 데이터로 미세 조정하기 위해서는 많은 시간과 자원이 필요하다는 한계점이 있다. 이 때문에 대규모 언어 모델의 패러다임은 모델을 미세 조정하는 모델 엔지니어링 방식에서 모델에게 적합한 지시를 내리는 프롬프트 엔지니어링 방식으로 옮겨가고 있다.

대규모 언어 모델에서 프롬프트는 모델로부터 결과물을 생성하기 위한 입력을 의미한다[4]. 프롬프트를 구성하는 방식에 따라 결과물의 품질이 크게 달라질 수 있기에 프롬프트 엔지니어링의 중요성이 부각되고 있으며[5], 이를 활용한 기초 모델의 사용이 기대되고 있다.

2. ARC 문제에 프롬프트 엔지니어링 적용

2.1 대규모 언어 모델 선택 배경

공개된 대규모 언어 모델 중 ChatGPT의 GPT-4.0을 사용하였다. 이는 GPT-4.0이 다른 모델에 비해 사용하기 편하고 성능 또한 뛰어나기 때문이다. 예를 들면, ChatGPT는 웹서비스와 API 서비스를 제공하기 때문에 ARC 벤치마크 데이터셋에 쉽게 적용할 수 있다는 점이다. 또한 ARC 문제와 같은 퓨샷(few-shot) 상황일 때, GPT-4.0의 성능이 GPT-3.5를 포함한 다른 대규모 언어 모델들보다 대부분의 벤치마크에서 월등한 성능을 보여주었고, 심지어 일부 벤치마크에서는 미세 조정한 SOTA(State-of-the-Art) 모델보다 뛰어났다[6].

2.2 ARC 예제를 표현하는 다양한 방법

사전에 다양한 프롬프트를 사용해 ARC 문제 해결을 시도했을 때, ARC 예제의 그리드를 숫자로 표현할 경우 ChatGPT가 해당 숫자를 계산하는 경우가 있었고, 알파벳으로 표현할 경우 단어로 인식해 토큰화하는 문제가 발생했다. 이 문제를 해결하기 위해 각 그리드의 원소를 ‘,’와 ‘[’, ‘]’ 등을 이용해서 구분해주었다. 이와 같이 구분자를 사용해주면 그리드의 원소를 연속된 숫자나 단어로 토큰화하는 문제를 해결할 수 있었다. 또, 그리드의 원소를 다양한 방식으로 표기해보고 이에 따른 영향을 파악하고자 하였다. 그래서 숫자와 문자 이외에도 색상을 특수문자로 표현하는 방법을 고안했다. ARC 문제의 배열 원소로 알파벳 문자를 사용한 경우, 알파벳 소문자 ‘a’부터 ‘j’까지의 문자를 이용하였다. 이때 각 문자는 서로 다른 색상을 의미한다. 숫자를 사용한 경우, 배열 원소는 0~9 사이의 값을 가진다. 특수문자를 사용한 경우, 총 10가지의 특수문자(■□○ ◇¤♤♧△♡☆)로 색상을 표현했다.

2.3 페르소나 프롬프트 엔지니어링

ChatGPT를 보다 잘 활용하기 위해서 널리 알려진 프롬프트 엔지니어링 방법의 하나인 페르소나 프롬프트를 사용했다. 페르소나 프롬프트는 대규모 언어 모델이 특정 문제에 초점을 맞출 수 있도록 프롬프트를 작성하는 기법이다. 예를 들어, ChatGPT가 ARC 문제에 초점을 맞출 수 있도록 ARC 문제에 대한 정보들과 수행해야 하는 역할을 프롬프트에 명시하는 것이 이에 해당한

다. 그림 2의 왼쪽 프롬프트 내용이 이에 해당한다.

본 연구는 ARC 문제를 풀기 전에 페르소나 프롬프트를 우선 입력했다. 그 후, ARC 문제에 대한 예제들과 한 개의 입력값을 주었다. 이때 예제의 모든 입력값과 출력값은 2차원 배열 형태로 주었으며, ChatGPT가 주어진 정보로 적절한 출력값을 도출할 수 있도록 프롬프트를 구성했다. 이는 그림 2의 오른쪽 프롬프트와 같다.

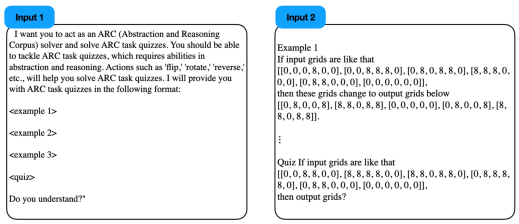

그림 2 실험에 사용된 프롬프트 (문제2)

Fig. 2 Prompts used in the experiment (Problem 2)

2.4 생각의 사슬

기존의 프롬프트들은 주어진 예시에 대한 답을 단답형으로 구성하여 주었고 이로 인해 대규모 언어 모델이 적절한 답을 도출하지 못한 경우가 많았다. 이를 해결하고자 등장한 프롬프트 엔지니어링 방법이 생각의 사슬(Chain-of-Thought)[7]이다. 생각의 사슬은 주어진 예시의 답을 단답형으로 작성하는 것이 아닌 여러 단계로 나눠 작성한다. 대규모 언어 모델은 생각의 사슬이 적용된 프롬프트를 활용하여 문제를 단계별로 풀어감으로써 인과관계를 잘 파악할 수 있다. 아래의 그림 3에서 왼쪽은 기존의 프롬프트이며 오른쪽은 생각의 사슬을 적용한 프롬프트이다.

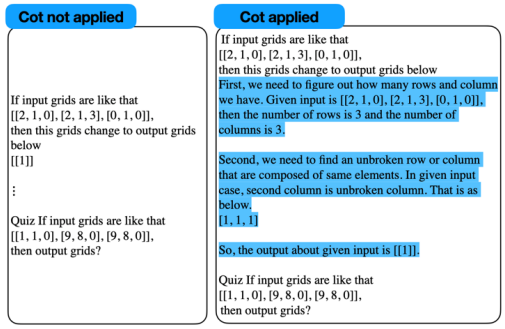

그림 3 생각의 사슬이 적용된 프롬프트 예시

Fig. 3 Example of a prompt with the chain of thought

3. 실 험

본 연구에서는 ARC 배열 구성 방식에 따른 실험 및 생각의 사슬을 적용한 실험 그리고 이를 확장한 실험 등을 진행하였다. 또한, ARC 문제 해결을 위한 프롬프트 엔지니어링의 가능성을 확인하기 위해 각 실험에 대한 심층 분석을 진행하였다.

3.1 ARC 예제 구성에 대한 비교 실험

ARC 학습 데이터셋에 대해서 OpenAI의 API로 davinci모델을 1에폭 미세 조정한 후, 해당 모델이 평가 데이터셋에서 맞춘 문제 4개를 실험에 사용했다. 해당 문제들을 ChatGPT가 어느 정도로 정확하게 풀 수 있는지, 그리고 배열 원소를 숫자, 문자, 특수문자로 주었을 때 어떤 영향을 미치는지 실험했다. 정확한 실험을 위해 문제별로 총 10회의 테스트를 반복했다. 실험에 사용한 4개 문제는 그림 4와 같다.

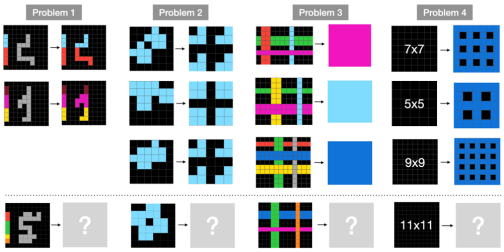

그림 4 실험에 사용한 4개 문제

Fig. 4 Four problems we used in our experiment

문제 1은 회색 부분의 값을 해당 행의 첫 번째 원소의 색상으로 바꿔주는 문제로, 행의 개념을 이해하고 있다면 풀 수 있다. 문제 2는 입력값에서 4개의 같은 모양으로 나눌 수 있는 도형을 찾은 후 해당 도형을 좌측 상단, 우측 상단, 좌측 하단, 우측 하단에 배치해야 하는 문제로, 객체라는 개념을 이해해야 풀 수 있다. 문제 3은 행 혹은 열 중에서 중간에 끊어지지 않고 끝까지 이어진 색상을 찾는 문제로, 행과 열의 개념을 이용해야 풀 수 있다. 문제 4는 배열의 행과 열이 줬을 때 홀수 행에는 파란색으로 다 채우고 짝수 행에는 파란색과 검은색을 번갈아 가며 채우는 문제로, 패턴을 그리드에 맞게 채우는 문제이다.

3.1.1 실험 결과

정리된 실험 결과 표 1를 통해 알 수 있는 유의미한 부분은 크게 2가지이다. ChatGPT가 풀기 어려워하는 유형의 문제를 파악했다는 점과 배열 원소 기호와 문제 유형에 따라 성능의 변화가 있다는 점이다.

예를 들어 문제 4는 배열 원소의 종류에 상관없이 100%의 정확도를 달성했지만, 문제 2는 전체적으로 낮은 정확도를 기록했다. 이를 통해 ChatGPT는 문제 4와 같이 반복되는 패턴을 파악하고 적용하는 문제는 쉽게

풀 수 있지만, 문제 2와 같이 객체를 찾고 활용하는 문제는 풀기 어렵다는 사실을 알 수 있다.

특히, 문제 2와 같이 객체를 찾고 이를 활용하는 문제에서는 배열 원소가 특수문자일 때의 성능이 더 좋았다. 반면 문제 1과 같이 각 행의 첫번째 원소를 특정 원소에 대입해서 풀어야 하는 문제는 문자와 숫자로 원소를 표기했을 때 성능이 더 좋았다. 또한 문제 3과 같이 행 혹은 열 중간에 원소가 바뀌지 않는 특정 원소를 찾는 문제의 경우, 문자보다 숫자와 특수문자로 원소를 표기했을 때 성능이 더 좋았다. 이를 통해 문제에 따라 적절한 배열 원소를 입력하는 것이 성능에 영향을 준다는 결론을 얻었다.

표 1 ARC 배열 구성 방식에 따른 실험 결과
Table 1 Experimental results based on how the ARC array is configured

		Number of correct answers	Number of incorrect answers	Accuracy
Problem 1	Characters	8	2	80%
	Number	8	2	80%
	Special characters	2	8	20%
Problem 2	Characters	0	10	0%
	Number	0	10	0%
	Special characters	4	6	40%
Problem 3	Characters	3	7	30%
	Number	8	2	80%
	Special characters	9	1	90%
Problem 4	Characters	10	0	100%
	Number	10	0	100%
	Special characters	10	0	100%

3.1.2 심층 분석

배열 원소의 종류에 따른 ChatGPT의 문제 접근법을 파악하고자, 표기 방식에 따라 정확도에 큰 변화가 있었던 문제 1, 문제 2, 문제 3의 경우를 추가로 분석해 보았다.

문제 1에서 ChatGPT의 접근법은 총 8가지가 있었다. (1) 회색(5, f, ♧)을 해당 행의 첫 번째 열의 색으로 바꾼다. (2) 회색(5, f, ♧)을 주변에 있는 가장 가까운 다른 색으로 바꾼다. 이때, 검정색(0, a, ■)인 경우는 제외한다. (3) 예제에서 관측된 패턴을 그대로 적용한다. 예를 들어 [2, 0, 5, 5]가 입력값이고 출력값이 [2, 0, 2, 2]일 때, 풀어야 할 문제의 첫 번째 열의 값이 2라면 [2, 0, 2, 2]를 넣어준다. (4) 입력값을 그대로 복제하여 출력값으로 사용한다. (5) 행마다 그 행에 존재하는 유일한 색으로 회색(5, f, ♧)을 대체한다. (6) 각 행에 존재하는 회색(5, f, ♧)을 바로 위 행의 유일한 색으로 대체한다. (7) 각 행의 첫번째 열이 검정색(0, a, ■)이 아닐 경우, 해당 검정색을 동일한 행의 두 번째 열의 색으로 바꿔준다. (8) 검정색(0, a, ■)을 제외한 모든 색을 회색(5, f, ♧)으로 바꿔준다.

배열 원소로 숫자를 사용했을 경우, 접근법 (1), (2), (3), (4)를 각각 6회, 2회, 1회, 1회 사용했다. 이 중에서 접근법 (1), (2)로 총 8번의 정답을 맞혔다. 문자로 표기했을 경우, 접근법 (1), (3), (7), (8)을 각각 7회, 1회, 1회, 1회 사용했고 이 중에서 (1), (3) 접근법으로 8번 맞혔다. 특수문자를 사용했을 경우, 접근법 (1), (2), (4), (5), (6)을 각각 2회, 1회, 1회, 3회, 3회 사용했으며, 이 중에서 접근법(5)으로 총 2회 맞혔다.

배열 원소를 숫자 혹은 문자로 사용했을 경우에는 주로 접근법(1)을 사용했지만, 특수문자를 사용했을 경우에는 주로 접근법 (5), (6)과 같이 각 행의 유일한 문자로 ♧을 대체하는 경우가 많았다. 그러나 접근법(6)과 같이 다른 행의 유일한 문자로 대체해서 잘못된 출력값을 도출하는 경우도 있었다. 그리고 올바른 접근법(5)을 선택했지만 이를 제대로 적용하지 못해서 부적절한 출력값을 도출하는 경우도 있었다.

문제 2에서 ChatGPT의 접근법은 배열 원소의 종류와 관계없이 세 가지로 나눌 수 있었다. 첫 번째 접근법은 각 행과 열을 제거, 교환, 변환, 삽입 등을 한 단계씩 수행하면서 배열를 변형시키는 방식이다. 해당 방식의 경우, 여러 단계를 거쳐야 하기 때문에 중간 결과가 잘못된 경우가 많았고 이로 인해 정답을 맞히는 비율이 낮았다. 두 번째 접근법은 각 행을 구성하는 원소의 조합에 따라서 제거, 변환 등의 과정을 수행하는 방식이다. 해당 문제는 행과 열을 통해서 객체를 판별하고 변환을 적용시켜야 하는 문제임으로 두 번째 접근법으로는 문제를 해결하기 어렵다. 세 번째 접근법은 예제의 출력을 그대로 출력하는 방식이다. 문제 2의 경우 예제에서 제시한 출력값이 2종류밖에 없었기에 이러한 접근법을 시도했던 것으로 예측되었다. 해당 방식의 경우, 단순히 예제의 출력값을 그대로 사용했고 2종류의 예제 중 정답이 있었기 때문에 상대적으로 높은 성능을 보였다.

다만 배열 원소로 숫자를 사용한 경우에는 첫 번째 접근법을, 문자를 사용한 경우에는 두 번째 접근법을, 특수 문자를 사용한 경우에는 세 번째 접근법을 채택한 비율이 높았기에 프롬프트 방식에 따른 성능 차이를 확인 할 수 있었다.

문제 3에서 ChatGPT의 접근법은 크게 3가지로 나눌 수 있다. 첫 번째 접근법은 동일한 원소로 행 혹은 열이

구성되어 있을 때 해당 원소를 찾는 방식이다. 두 번째 접근법은 특정 위치의 원소를 정답으로 예측하는 방식이다. 예를 들어, 현재 배열의 중앙에 있는 원소를 정답으로 예측하는 경우가 있다. 세 번째 접근법은 각 배열을 구성하는 원소별로 인접한 원소의 종류와 종류의 개수에 따라 답을 예측하는 방식이다.

배열 원소로 숫자와 특수문자를 사용했을 경우, 첫번째 접근법을 사용하는 경우가 많았다. 반면에, 문자의 경우, 두 번째와 세 번째 접근법으로 문제를 해결하고자 하는 경우가 많았다. 그러나 두 번째와 세 번째 접근법으로는 잘못된 결과를 예측하기 쉽기 때문에 문자로 배열을 표기했을 때, 성능이 상대적으로 낮았다.

3.2 생각의 사슬을 적용한 실험

3.1에서 사용했던 문제 4개에 대해서 생각의 사슬을 적용해 보았다. 아래의 내용은 각 문제에 적용한 생각의 사슬을 서술한 것이다.

문제 1의 경우, 행에 회색(5, f, ♧)이 있을 때 회색을 행의 첫 번째 원소로 치환하고 이에 대한 결과를 서술해 준다. 이와 같은 절차를 모든 행에 대해서 적용하여 최종 결과까지의 과정을 프롬프트로 작성해 주었다.

문제 2의 경우, 배열의 하늘색(8, i, ♡)을 통해 객체의 크기 예상하는 법, 각 객체를 찾는 법, 찾은 객체를 모서리에 배치하는 법, 마지막으로 3행 3열을 제거하여 배열의 크기를 맞춰주는 법을 순차적으로 설명한 프롬프트를 작성했다.

문제 3의 경우, 주어진 배열의 행과 열의 크기를 파악한 후 끊어지지 않고 단일 원소로 구성된 행 혹은 열을 찾고 단일 원소 출력하도록 프롬프트를 작성했다.

문제 4의 경우, 주어진 배열의 행과 열의 크기를 파악한 후 홀수 행에는 검은색(0, a, ■)으로 구성된 단일 패턴을 만들고 짝수 행에는 파란색(1, b, □)과 검은색(0, a, ■)이 반복되는 행을 구성하도록 프롬프트를 작성하였다.

3.2.1 실험 결과

생각의 사슬을 적용한 프롬프트를 4개의 문제에 대해서 문제별 10회씩 반복하였다.

정리된 실험 결과 표 2를 통해 알 수 있는 유의미한 부분은 크게 2가지이다. 배열의 원소를 숫자 혹은 문자로 표현했을 경우는 생각의 사슬을 적용한 결과가 적용하지 않았을 때보다 성능이 같거나 향상되었다. 특히, 문제 2의 경우, 생각의 사슬을 적용하기 전에는 한 문제도 맞추지 못했지만, 적용한 후에는 10번 중 9번을 맞추었다. 반면, 배열을 특수문자로 구성했을 경우에는 문제 유형에 따라 성능이 대폭 향상되거나 하락하였다. 문제 1에 경우에는 이전 실험 결과 표 1보다 8번 더 맞췄지만 문제 2와 문제 3번의 경우, 각각 3번, 5번 더 틀렸다.

표 2 생각의 사슬 적용 시 실험 결과

Table 2 Experimental results with the chain of thought

		Number of correct answers	Number of incorrect answers	Accuracy
Problem 1	Characters	10	0	100%
	Number	10	0	100%
	Special characters	10	0	100%
Problem 2	Characters	9	1	90%
	Number	9	1	90%
	Special characters	1	9	10%
Problem 3	Characters	4	6	40%
	Number	8	2	80%
	Special characters	4	6	40%
Problem 4	Characters	10	0	100%
	Number	10	0	100%
	Special characters	10	0	100%

3.2.2 심층 분석

생각의 사슬을 통해 성능이 큰 폭으로 향상된 문제 1-특수문자, 문제 2-숫자, 문자의 경우와 큰 폭으로 하락한 문제 2-특수문자, 문제 3-특수문자의 경우를 분석해 보았다.

성능이 큰 폭으로 향상된 문제 1-특수문자의 경우, 생각의 사슬을 적용하기 전과 달리 모두 각 행의 회색(5, f, ♧)을 해당 행의 첫 번째 원소로 바꿔주는 접근법으로 문제를 맞혔다. 문제 2-숫자, 문자의 경우, 객체 크기 예측, 각 객체 찾기, 각 객체를 모서리에 위치시키기, 3행 3열 제거하기 등의 단계로 문제를 해결하였다. 그러나 두 표기 방식 모두, 대부분의 풀이에서 중간단계의 결과가 잘못되었지만 최종결과는 맞히는 경우가 많았다. 예를 들어서 객체의 크기는 4인데 3으로 예측하고 크기가 4인 객체를 탐지하는 경우가 있었다. 또 다른 경우에는 중간 단계인 각 객체를 예측하는 부분에서 객체를 잘못 예측하였지만 최종 결과는 올바르게 나온 경우도 있었다. 이와 같은 현상이 발생한 원인은 ChatGPT가 알고리즘을 통해서 특정 원소를 세거나 객체를 찾는 것이 아닌 주어진 프롬프트를 가지고 출력을 생성하는 모델이기 때문에 발생한 것으로 예상된다.

성능이 큰 폭으로 하락한 문제 2-특수문자의 경우, 문제 2-숫자와 같이 중간단계의 결과가 잘못되어 최종 결과 또한 잘못된 결과가 출력된 경우가 많았다. ChatGPT의 구조가 공개되어 있지 않아 정확한 분석은 어렵지만, ChatGPT가 출력한 답변과 풀이 과정을 통해 원인을 분석해 보면 원소를 숫자로 표기했을 경우, 각 객체의 크기가 4보다는 작다는 점을 활용하여 객체를

찾는다. 반면 배열의 원소가 특수문자일 경우, 이와 같은 정보를 활용하지 못한 것으로 보였으며 이로 인해 잘못된 결과를 출력한 것으로 예상된다. 문제 3-특수문자의 경우, 동일한 색(숫자, 문자, 특수문자)으로 이어진 행 혹은 열을 찾고 해당 색을 출력하는 올바른 접근법으로 문제를 풀려고 하였지만 중간에 다른 색이 포함된 잘못된 행 혹은 열을 출력하는 경우가 총 6번 있었다. 이는 문제 2-특수문자의 경우와 같이 특정 원소를 세는 것에 대해서 대규모 언어 모델이 프롬프트와 사전 학습된 파라미터를 이용하여 생성했기 때문에 발생한 것으로 예상된다.

3.3 공통된 생각의 사슬 적용한 실험 및 결과

3.2의 생각의 사슬 적용 실험은 각 문제에 대한 풀이 방식을 프롬프트에 적용한 것이다. 이는 추상화와 추론을 위한 ARC 벤치마크의 취지에 부합하지 않을 수 있다. 그렇기 때문에, 각 문제들과 관련 없는 새로운 유형의 문제를 예제로 사용하고 그에 대한 풀이법을 서술하여 공통적인 프롬프트를 만들어줬다. 또한, 많은 문제에 해당 방법을 적용하여 성능을 평가하고자 100개의 ARC 문제를 가지고 실험을 10회씩 반복해서 진행하였다. 해당 실험에서는 이전 실험 결과 표 2에서 가장 좋은 성능을 낸 숫자 표기 방식으로 배열의 원소를 구성하여 실험을 진행했다.

표 3을 통해 알 수 있듯이, 10번의 반복 실험 결과 평균 10% 정도의 정확도를 보였으며 이전 실험의 문제 4와 같이 단순 패턴 적용 문제에 대해서 주로 맞췄다. 또한 특정 객체를 찾는 문제에 대해서도 맞췄다. 그럼에도 불구하고 틀린 문제 중 대다수가 객체 개념을 활용해야 풀 수 있는 문제였다. 이를 통해 ARC 문제를 해결함에 있어 객체 정보가 중요하다는 것을 알 수 있었다.

표 3 ARC 100문제에 대해서 통일된 생각의 사슬 적용한 10회 반복 실험 및 결과

Table 3 Experiment and results for 10 iterations of applying a same chain of thought to an ARC 100 problem

	Number of correct answers	Number of incorrect answers	Accuracy
1	8	92	8%
2	10	90	10%
3	9	91	9%
4	10	90	10%
5	12	88	12%
6	10	90	10%
7	9	91	9%
8	10	90	10%
9	12	88	12%
10	11	89	11%

4. 최근 관련 연구동향

최근 몇 개월 사이, 대규모 언어 모델을 ARC에 적용해 본 사례가 늘어났으며 이에 대한 여러 논문이 아카이브에 올라왔다[5,8,9].

Moskvichev의 논문[8]은 ARC 문제를 유형별로 정리 및 문제를 추가하여 데이터셋을 공유하였고 Kaggle ARC대회에서 1등 코드와 2등한 코드 그리고 GPT-4.0를 이용하여 공개한 데이터셋에 대해서 평가해 보았다. 대부분의 유형에서 1등의 코드가 GPT-4.0보다 성능이 좋았다.

Xu의 논문[5]은 대규모 언어 모델이 주어진 프롬프트에서 객체를 파악하는 것을 어려워한다는 점을 지적하였고 이를 해소하기 위해 프롬프트를 JSON 파일과 같은 형식으로 줌으로써 객체 정보를 대규모 언어 모델이 잘 파악할 수 있도록 해주었다. 50개의 ARC 문제에 대해서 JSON 형식과 같이 프롬프트를 구성했을 경우에는 23문제를 맞췄으며 JSON 형식을 적용하지 않았을 때는 13문제를 맞췄다.

Mirchandani의 논문[9]에서는 ARC 문제를 해결하기 위해 PaLM모델, GPT-3.0모델 계열(text-davinci-001), 그리고 GPT-3.5모델 계열(text-davinci-002/003)을 사용하였고 이 중에서 text-davinci-003의 성능이 가장 좋았다. 해당 논문에서는 ARC와 이외에도 다양한 벤치마크 데이터셋에 대규모 언어 모델을 적용했다.

이처럼 현재 대규모 언어 모델을 ARC에 적용해 보는 논문들은 대규모 언어 모델의 약점을 파악하고 그것을 보완하기 위한 방향으로 연구가 진행되고 있다.

5. 결론 및 향후 연구

본 논문은 ARC 문제를 풀기 위해 기존 연구들이 활용한 DSL을 적용하는 대신에 프롬프트 엔지니어링 방법을 활용했다. 실험 결과를 통해 ARC 문제를 일부 풀 수 있는 것을 확인했고 문제 유형에 따른 적합한 배열 원소 표기법과 프롬프트가 존재함을 알 수 있었다. 이를 통해 ARC 문제를 풀 새로운 가능성을 확인할 수 있었다.

향후 연구에서는 객체나 대칭 이동, 회전과 같은 사전 개념들을 프롬프트에 사전 학습시키면 성능이 더욱 좋아질 것으로 예상된다. 이 외에도 인간이 이해할 수 있는 형태의 프롬프트가 아닌 모델이 학습을 통해서 만든 프롬프트를 적용 및 프롬프트에 문제 유형과 객체 정보

들을 추가하는 등의 방법으로 더욱 정교한 프롬프트를 구성한다면 성능 향상에 도움이 될 것으로 기대된다.

References

[1] François Chollet, “On the measure of intelligence,” arXiv:1911.01547, 2019.

[2] Raphael Fischer et al., “Solving Abstract Reasoning Tasks with Grammatical Evolution,” LWDA, 2020.

[3] Tuan Dinh et al., “LIFT: Language-Interfaced Fine-Tuning for Non-Language Machine Learning Tasks,” NeurIPS, 2022.

[4] Pengfei Liu et al., “Pre-train, prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing,” ACM Computing Surveys, 2023.

[5] Xu, Yudong et al., "LLMs and the Abstraction and Reasoning Corpus: Successes, Failures, and the Importance of Object-based Representations." arXiv preprint arXiv:2305.18354, 2023.

[6] OpenAI, “GPT-4 Technical Report,” arXiv:2303.0877, 2023.

[7] Wei, Jason et al., "Chain-of-thought prompting elicits reasoning in large language models," NeurIPS, 2022.

[8] Moskvichev, Arseny, Victor Vikram Odouard, and Melanie Mitchell, "The ConceptARC Benchmark: Evaluating Understanding and Generalization in the ARC Domain," *Transactions on Machine Learning Research,* pp. 2835-8856, 2023.

[9] Mirchandani, Suvir et al., "Large Language Models as General Pattern Machines," arXiv preprint arXiv:2307.04721, 2023.

심 우 창

2023년 조선대학교 컴퓨터공학과 졸업(학사). 2023년~현재 GIST AI대학원 석사과정. 관심분야는 인공 일반 지능(Artificial General Intelligence), 표현 학습

진 혜 빈

2020년~현재 GIST 전기전자컴퓨터공학부 학사과정. 관심분야는 인공지능(Artificial Intelligence), 데이터 분석(Data Analytics)

김 세 진

2023년 KAIST 전산학부 졸업(박사). 2023년~현재 GIST AI대학원 Post Doc. 관심분야는 인공 일반 지능(Artificial General Intelligence), 강화 학습(Reinforcement Learning), 메타 학습(Meta-Learning) 등

김 선 동

2019년 KAIST 지식서비스공학대학원(공학박사). 2019년~2022년 기초과학연구원 차세대연구리더. 2022년~현재 광주과학기술원AI대학원 조교수. 관심분야는 강인공지능을 위한 표현 학습, 지식 추론