

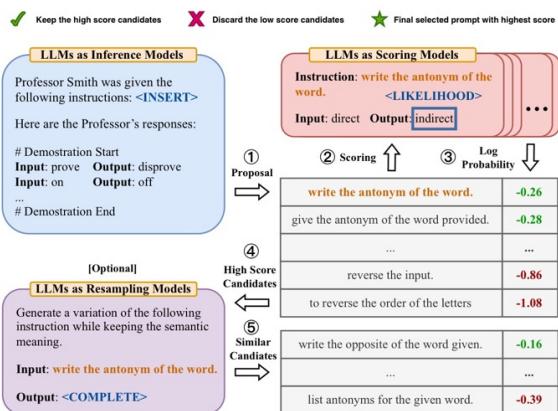


# 1. LARGE LANGUAGE MODELS ARE HUMAN-LEVEL PROMPT ENGINEERS

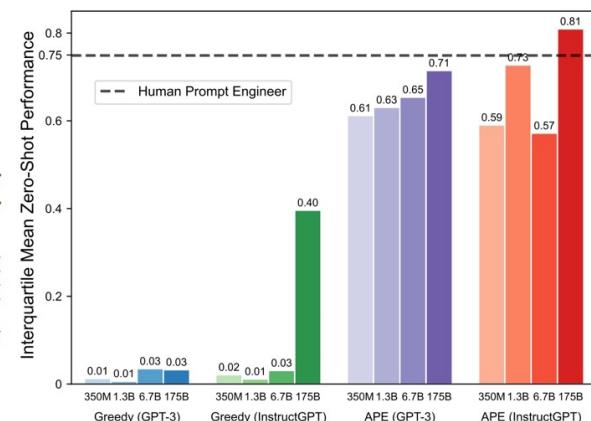
## 0. Abstract

- 자연어 명령을 활용하는 **대규모 언어 모델(LLMs)**은 일반적인 컴퓨터처럼 놀라운 능력을 가지고 있지만, 그 성능은 모델을 조작하는 데 사용되는 프롬프트의 품질에 크게 의존함
  - 효과적인 프롬프트 대부분은 사람에 의해 **수동**으로 만들어짐
- 해당 연구는 **프로그램 합성과 인간의 프롬프트 설계 방식**에서 영감을 받아 자동으로 지침을 생성하고 선택하기 위한 **자동 프롬프트 엔지니어(APE)**를 제안
  - 이 방법에서는 지침을 "프로그램"으로 다루며, 선택된 지침을 최적화하기 위해 LLM이 제안한 다양한 지침 후보를 검색
  - 선택된 지침의 품질은 다른 LLM이 이 지침을 사용하여 테스트됨
- 실험 결과, APE가 **자동**으로 생성한 지침은 이전의 LLM 기준보다 우수한 성능을 보이며, 여러 작업에서 인간이 만든 지침과 비슷하거나 더 나은 성능을 달성함
- 이 연구는 APE를 통해 **프롬프트를 개선**하여 다양한 작업에서 성능 향상을 이끌어내고, 다양한 분야에서 모델을 조작하여 **진실성과 정보성**을 향상시킬 수 있음을 보여줌

## 1. Introduction



(a) Automatic Prompt Engineer (APE) workflow



(b) Interquartile mean across 24 tasks

- 대규모 언어 모델(LLMs)는 놀라울 만한 성과를 달성하였음
  - 범용성과 함께 이를 원하는 방향으로 조작하는 문제에 직면
- LLMs를 원하는 방향으로 동작하기 위해 여러 연구가 진행됨
  - 미세 조정(fine-tuning)
  - 문맥 내 학습(in-context learning)
  - ★ 자연어 프롬프트 엔지니어링
    - 인간과 기계 간 자연스러운 상호작용을 제공
    - 하지만 언어 프롬프트는 항상 원하는 결과를 얻지 못할 때가 있으며, 이는 모델과 프롬프트 간의 호환성 정보가 부족하기 때문임
    - LLMs는 프로그램을 자연어 지침으로 실행하며, 이러한 프로그램이 어떻게 처리되는지는 인간에게 직관적이지 않을 수 있으며, 지침의 품질은 하위 작업에서 실행될 때만 평가할 수 있음

⇒ LLMs를 조작하기 위한 효과적인 지침을 만들고 사용하기에 어려운 경우가 있음
- 효과적인 지침을 생성하고 검증하는 인간의 노력을 줄이기 위해 LLMs를 사용하여 지침을 자동으로 생성하고 선택하는 새로운 알고리즘을 제안  

⇒ 자연어 프로그램 합성 문제
- LLMs의 범용 능력을 다음과 같이 활용
  1. LLM을 추론 모델로 활용하여 입력-출력 쌍을 기반으로 작은 데모를 통해 지침 후보를 생성
  2. 우리가 조종하려는 특정 LLM에 대한 각 지침에 대한 점수를 계산하여 검색 프로세스를 안내

### 3. LLMs가 의미론적으로 유사한 지침 변형을 제안하면서 가장 우수한 후보를 반복적으로 개선하는 몬테 카를로 검색 방법을 도입

- 기본적으로 LLMs에게 데모를 기반으로 지침 후보를 생성하고 어떤 지침이 가장 유망한지를 평가하도록 요청

#### ⇒ 자동 프롬프트 엔지니어(APE)

- 주요 기여

- 지침 생성을 자연어 프로그램 합성으로 정의하고, LLMs의 안내 하에 블랙 박스 최적화 문제로 다루며, 해결 방법 근사화를 위한 간단하고 반복적인 몬테 카를로 검색 방법을 제안

- ▼ 몬테 카를로 검색 방법

- 확률적인 시뮬레이션을 사용하여 다양한 가능성을 탐색하고 최적의 해를 찾는 컴퓨터 알고리즘
  - 모델이 생성한 지침을 사용한 zero-shot 학습에서 24/24 Instruction Induction 및 17/21 Big-Bench 작업에서 인간 수준의 성능을 달성
  - 다양한 질적 및 양적 분석을 제공하며, few-shot 학습 향상, 더 나은 zero-shot chain-of thought 프롬프트 찾기, 그리고 LLMs를 진실성 및/또는 정보성으로 조작하는 다양한 응용 프로그램을 보여줌

- ▼ zero-shot chain-of thought 프롬프트

- 대규모 언어 모델(LLM)을 조작하고 모델로 하여금 특정 작업을 수행하도록 이끌기 위한 프롬프트의 한 유형
    - Zero-shot: 모델이 이전에 학습한 데이터나 지침을 기반으로 하지 않고 처음 보는 작업을 수행하도록 하는 것을 의미
      - 모델은 이 작업에 대한 명시적인 훈련 데이터나 지식이 없는 상태에서 작업을 수행해야 함
    - Chain-of-thought: zero-shot 작업에서 모델이 여러 단계의 연속적인 사고나 추론을 수행하도록 하는 것
      - 모델은 이전 단계의 결과를 기반으로 다음 단계를 수행하며, 연결된 사고 과정을 따름

⇒ 모델에게 이전 학습 없이 처음 보는 주제에 대한 연속적인 사고와 설명을 요청하는 방법을 의미

## 2. 관련 연구

## 2-1. 대규모 언어 모델(LLMs)

- 모델 크기, 학습 데이터 양, 학습 계산 능력 측면에서 변화를 주어 확장하는 것이 다양한 자연어 처리(NLP) 작업에서 성능을 예측 가능하게 했음이 밝혀져 왔음
  - 이런 확장으로 인해 LLMs의 다양한 능력들이 발견됨
- 예시
  - few-shot(in-context) 학습
  - 제로샷 문제 해결
  - 연속적 추론
  - 지침 따르기
  - 지침 유도 등
- 이 논문에서는 LLMs를 자연어 지침에 의해 명시된 프로그램을 실행하는 블랙 박스 컴퓨터로 취급하며, 모델이 생성한 지침을 사용하여 LLM의 동작을 어떻게 제어할 수 있는지에 대해 조사함

## 2-2. Prompt Engineering

- LLMs와 같은 범용 모델과의 상호작용을 향상시키는 중요한 방법임
  - 하지만 LLMs는 프롬프트를 인간과 다르게 이해하는 경향이 있어서 조심스러운 설계가 필요
- 많은 성공적인 프롬프트 튜닝 방법은 그래디언트 기반 방법을 사용
  - 그러나 이 방법은 규모가 커지면 계산 비용이 증가하고 모델 접근 방식이 제한될 수 있음
- 이 논문에서는 프롬프트 검색에 이산 방법의 구성 요소를 활용
  - 지시문을 자연어 가설 공간에서 직접 최적화하는 방법을 제안
  - 이것은 특정 컴포넌트에 특화된 모델을 사용하거나 인간 템플릿에 의존하지 않고도 단일 LLM에 의해 수행될 수 있다는 것을 보여줌

## 2-3. 프로그램 합성

- 특정 명세를 충족하는 프로그램을 자동으로 찾는 프로세스
- 현대 프로그램 합성은 다양한 명세를 사용
  - 입력-출력 예시나 자연어를 활용

- 프로그램 합성의 검색 공간은 도메인 특화 언어에서 일반 프로그래밍 언어로 확장 됨
- 이전 방법과 달리, 우리는 LLMs의 구조를 활용하여 자연어 프로그램을 검색
  - 추론 모델을 사용하여 검색 공간을 제한함으로써 검색 속도를 높임  
⇒ 다양한 작업에 적용 가능한 훈련 없는 모델을 사용하는 것을 의미

### 3. LLMs를 활용한 자연어 처리 프로그램의 합성

- 입력/출력 데모가 모집단  $\chi$ 에서 샘플링된 입력/출력 데모 데이터 집합  $D_{train} = (Q, A)$  와 특정 모델 M을 고려
- 목표
  - M에게 [p, Q] 지시문과 주어진 입력을 제공했을 때 M이 해당하는 출력 A를 생성하는 단일 지침 p를 찾는 것
  - 보다 형식적으로 이를 최적화 문제로 정의
    - 가능한 (Q, A)에 대한 각각에 대한 일련의 점수  $f(p, Q, A)$ 의 기대값을 최대화하는 지침 p를 탐색

$$\rho^* = \arg \max_{\rho} f(\rho) = \arg \max_{\rho} \mathbb{E}_{(Q, A)} [f(\rho, Q, A)]$$

- 입력인 Q가 비어 있는 문자열일 수 있으므로, 일반적으로 최적의 지침 p를 사용하여 바로 출력 {A}를 생성하는 프롬프트로 최적화
- 해당 작업은 주로 사람들에 의해 시도되어왔지만, 어떤 특정 지침이 모델 M과 얼마나 호환되는지에 대한 우리의 지식은 제한적임
  - 그래서 우리는 이 문제를 LLMs가 안내하는 블랙 박스 최적화 과정으로 취급
- **APE 알고리즘**은 두 가지 주요 구성 요소, 즉 제안 및 점수 부여에서 LLMs를 사용
  - 먼저 몇 가지 후보 프롬프트를 제안하고, 선택한 점수 함수에 따라 후보 집합을 필터링하고 정제한 다음, 가장 높은 점수를 가진 지침을 선택

---

**Algorithm 1** Automatic Prompt Engineer (APE)

---

**Require:**  $\mathcal{D}_{\text{train}} \leftarrow \{(Q, A)\}_n$ : training examples,  $f : \rho \times \mathcal{D} \mapsto \mathbb{R}$ : score function

- 1: Use LLM to sample instruction proposals  $\mathcal{U} \leftarrow \{\rho_1, \dots, \rho_m\}$ . (See Section 3.1)
- 2: **while** not converged **do**
- 3: Choose a random training subset  $\tilde{\mathcal{D}}_{\text{train}} \subset \mathcal{D}_{\text{train}}$ .
- 4: **for all**  $\rho$  in  $\mathcal{U}$  **do**
- 5:     Evaluate score on the subset  $\tilde{s} \leftarrow f(\rho, \tilde{\mathcal{D}}_{\text{train}})$  (See Section 3.2)
- 6:     **end for**
- 7: Filter the top k% of instructions with high scores  $\mathcal{U}_k \subset \mathcal{U}$  using  $\{\tilde{s}_1, \dots, \tilde{s}_m\}$
- 8: Update instructions  $\mathcal{U} \leftarrow \mathcal{U}_k$  or use LLM to resample  $\mathcal{U} \leftarrow \text{resample}(\mathcal{U}_k)$  (See Section 3.3)
- 9: **end while**

**Return** instruction with the highest score  $\rho^* \leftarrow \arg \max_{\rho \in \mathcal{U}_k} f(\rho, \mathcal{D}_{\text{train}})$

---

### 3-1. 초기 제안 분포

- 무한한 검색 공간으로 인해 적절한 지침을 찾는 것은 매우 어려운 작업이었음
  - 그러나 최근 NLP 분야의 진전으로 언어 모델이 다양한 자연어 텍스트를 생성하는데 뛰어나다는 것을 알 수 있음
- 이에 따라 미리 학습된 언어 모델을 활용하여 우리의 검색 과정을 안내할 후보 해결책 집합을 제안하는 것을 고려
  - 검색 과정을 안내할 좋은 후보 해결책 집합  $u$ 를 제안하는 것을 고려
  - LLMs에서 무작위 샘플은 원하는  $(Q, A)$  쌍을 생성하기 어려울 것이지만, 대신 LLM에게 입력/출력 데모를 고려하여 가장 가능성이 높은 점수를 가진 지침을 대략적으로 추론하도록 요청할 수 있음  
⇒  $P(p|D_{\text{train}})$  ( $f(p)$ 가 높음)

#### Forward mode 생성

- 우리는  $P(p|D_{\text{train}})$  ( $f(p)$ 가 높음) 분포에서 고품질 후보를 생성하기 위한 두 가지 접근 방식을 고려
  - 먼저, "포워드" 모드 생성에 기반한 방법을 채택  
⇒ 분포  $P(p|D_{\text{train}})$  ( $f(p)$ 가 높음)을 단어로 번역
  - 예를 들어, instruction induction 실험에서는 아래 과정을 통해 LLM에 프롬프트를 제공

### Forward Generation Template

I gave a friend an instruction and five inputs. The friend read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

**Input:**  $[Q_1]$    **Output:**  $[A_1]$   
**Input:**  $[Q_2]$    **Output:**  $[A_2]$

...

The instruction was <COMPLETE>

## Reverse Mode 생성

- "포워드" 모델은 대부분의 사전 학습된 LLMs에서 기본적으로 작동하지만,  $P(p|D_{train})$ ( $f(p)$ 가 높음)을 단어로 번역하려면 다른 작업에서 사용자 정의 엔지니어링이 필요
  - 지시문이 일반적으로 텍스트의 시작 부분에 있지만 "포워드" 모델은 텍스트를 왼쪽에서 오른쪽으로만 생성하므로 지시문이 프롬프트의 끝에서 예측되어야 하는 상황 때문임

⇒ 지시문이 텍스트 어디에 있든지 상관없는 더 유연한 접근 방식이 필요
- 이를 해결하기 위해 우리는 "reverse" 모드 생성을 고려
  - 이 방법은 누락된 지침을 추론하기 위해 infilling 기능을 갖춘 LLM(ex> T5, GLM, InsertGPT 등)을 사용
  - "reverse" 모델은 빈칸을 채우면서  $P(p|D_{train})$ ( $f(p)$ 가 높음)에서 직접 샘플링

### Reverse Generation Template

I instructed my friend to <INSERT>.

The friend read the instruction and wrote an output for every one of the inputs. Here are the input-output pairs:

**Input:**  $[Q_1]$    **Output:**  $[A_1]$   
**Input:**  $[Q_2]$    **Output:**  $[A_2]$

...

## 맞춤화된 지시문

- 위의 예시보다 더 적합한 프롬프트가 존재할 수 있음
  - 예를 들어, 우리의 TruthfulQA 실험에서는 원래 데이터셋의 인간 디자인 지침을 기반으로 시작하고 "reverse" 모델에게 빈칸을 채우는 데 맞는 초기 지침 예시를 제안하도록 요청

### Template for TruthfulQA

Professor Smith was given the following instructions: <INSERT>

Here are the Professor's responses:

Input:  $[Q_1]$  Output:  $[A_1]$

Input:  $[Q_2]$  Output:  $[A_2]$

...

## 3-2. 평가 함수들

- 우리의 문제를 블랙 박스 최적화로 설정하기 위해 데이터셋과 모델이 생성하는 데이터 간의 정확한 일치를 측정하는 점수 함수를 선택
- instruction induction 실험에서는 아래에 설명된 두 가지 잠재적인 점수 함수를 고려
  - TruthfulQA 실험: 주로 Lin 등(2022)에서 제안된 자동화된 메트릭과 유사한 실행 정확도에 중점
  - 각 경우에 대해 생성된 지침의 품질을 Equation(1)을 사용하여 평가하고, 보류 중인 테스트 데이터 집합  $D_{test}$ 에서 기대값을 계산

### 실행 정확도

- 실행 정확도 메트릭을 사용하여 지침 p의 품질을 평가하는 것을 고려( $f_{exec}$ 로 표기)
- 대부분의 경우 실행 정확도는 단순히 0에서 1 사이의 손실로 정의됨
  - $f(p, Q, A) = \mathbb{1}[M([p; Q]) = A]$
- 일부 작업에서는 실행 정확도가 불변성을 고려

### 로그 확률

- 더 부드러운 확률적 점수 함수를 고려하기도 함
  - 저품질의 지침 후보를 검색할 때 더 세밀한 신호를 제공하여 최적화를 개선할 수 있을 것이라고 가정
- 특히, 우리는 목표 모델 M 아래에서 지침과 질문에 따른 원하는 답변의 로그 확률을 고려
  - 각 샘플마다  $\log P(A|p; Q)$

### 효율적인 점수 평가

- 모든 지침 후보에 대해 전체 학습 데이터셋을 사용하여 점수를 계산하는 것은 비용이 많이 듦  $\Rightarrow$  필터링 체계를 채택

- 이 체계에서는 유망한 후보에 대해 더 많은 계산 리소스가 할당되고 품질이 낮은 후보는 더 적은 계산 리소스를 받음
- 이를 위해 **다단계 계산 전략**을 사용
- **다단계 계산 전략**
  1. 작은 학습 데이터셋의 일부를 사용하여 모든 후보를 평가
  2. 일정 임계값을 초과하는 점수를 가진 후보에 대해서만 학습 데이터셋에서 새로운 중첩되지 않는 부분을 샘플링하여 점수의 이동 평균을 업데이트

⇒ 해당 프로세스를 반복하여 낮은 품질의 후보에 대한 계산 비용을 크게 줄이고, 높은 품질의 샘플에 대한 정확한 계산 비용을 유지하면서 최종적으로 소수의 후보만 전체 학습 데이터셋에서 평가하게 됨

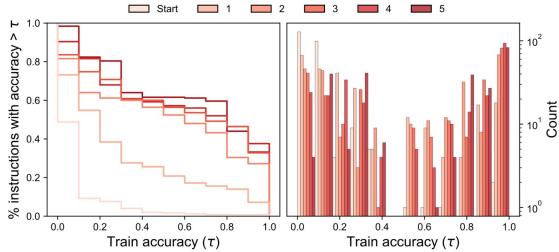
### 3-3. 반복적인 제안 분포

- 고품질 초기 지침 후보를 직접 샘플링하기 위해 시도하였음에도 불구하고, 다양성 부족이나 적절히 높은 점수를 가진 후보를 포함하지 않는 등 좋은 제안 세트  $U$ 를 생성하지 못할 수도 있음
  - 이러한 어려움이 발생하는 경우,  $U$ 를 다시 샘플링하기 위한 반복적인 과정을 탐색

#### 반복적인 몬테 카를로 검색

- 초기 제안에서만 샘플링하는 대신, 현재 최상의 후보 주변에서 검색 공간을 지역적으로 탐색하는 것을 고려
  - 성공 가능성이 더 높은 새로운 지침을 생성할 수 있음 ⇒ **반복 APE**
- 단계
  1. 각 단계에서 일련의 지침을 평가하고 점수가 낮은 후보를 걸러냄
  2. LLM에게 높은 점수를 가진 지침과 유사한 새로운 지침을 생성하도록 요청
- 이 접근 방식은 제안 세트  $U$ 의 전체적인 품질을 향상시키지만, 더 많은 단계에서도 가장 높은 점수의 지침은 일반적으로 동일한 것으로 나타남
  - 반복적인 생성은 경미한 향상을 제공

⇒ 기본적으로 반복적인 검색 없이 APE를 사용



## 4. 대형 언어 모델은 인간 수준의 프롬프트 엔지니어이다.

- APE가 LLMs를 원하는 동작으로 이끌 수 있는 방법을 검토
  - 4가지 관점에서 조사
    - zero-shot 성능
    - few-shot in-context 학습 성능
    - zero-shot chain-of-thought 추론
    - 진실성
- 우리의 실험은 APE가 작업 성능을 향상시킬 수 있는 프롬프트를 찾을 수 있음을 보여주며, 종종 인간이 작성한 것과 동등하거나 더 나은 성능을 발휘함을 입증함
- APE는 또한 종종 언어 모델에 가장 적합한 프롬프트를 성공적으로 새로운 작업으로 전환할 수 있는 통찰력 있는 트릭을 생성함

### 4-1. 지침 생성

- Honovich et al. (2022)에서 제안된 24개의 지침 생성 작업에서 zero-shot 및 few-shot in-context 학습의 효과를 평가
  - 언어 이해의 여러 측면을 포괄하며, 단순한 구문 구조부터 유사성 및 인과 관계 식별 까지 다양함
- 각 작업에 대해 학습 데이터에서 5개의 입력-출력 쌍을 샘플링하고 알고리즘 1을 사용하여 최상의 지침을 선택
  - 그런 다음 해당 지침의 품질을 **InstructGPT**에서 실행하여 평가
  - 다른 무작위 시드로 실험을 다섯 번 반복하여 평균과 표준 편차를 보고

Table 5: Raw templates used for model prompting in our experiments

Usage	Template
Zero-shot Evaluation	<b>Instruction:</b> [INSTRUCTION] <b>Input:</b> [Q <sub>test</sub> ]\n <b>Output:</b> <COMPLETE>
Few-shot Evaluation	<b>Instruction:</b> [INSTRUCTION] <b>Input:</b> [Q <sub>1</sub> ]\n <b>Output:</b> [A <sub>1</sub> ]\n <b>Input:</b> [Q <sub>2</sub> ]\n <b>Output:</b> [A <sub>2</sub> ] ... <b>Input:</b> [Q <sub>test</sub> ]\n <b>Output:</b> <COMPLETE>
Forward Generation	I gave a friend an instruction and five inputs. The friend read the instruction and wrote an output for every one of the inputs.\nHere are the input-output pairs: <b>Input:</b> [Q <sub>1</sub> ]\n <b>Output:</b> [A <sub>1</sub> ]\n <b>Input:</b> [Q <sub>2</sub> ]\n <b>Output:</b> [A <sub>2</sub> ] ... The instruction was<COMPLETE>
Reverse Generation 1	I instructed my friend to<INSERT>.The friend read the instruction and wrote an output for every one of the inputs.\nHere are the input-output pairs: <b>Input:</b> [Q <sub>1</sub> ]\n <b>Output:</b> [A <sub>1</sub> ]\n <b>Input:</b> [Q <sub>2</sub> ]\n <b>Output:</b> [A <sub>2</sub> ] ...
Reverse Generation 2	Professor Smith was given the following instructions:<INSERT>\nHere are the Professor's responses: <b>Q:</b> [Q <sub>1</sub> ]\n <b>A:</b> [A <sub>1</sub> ]\n <b>Q:</b> [Q <sub>2</sub> ]\n <b>A:</b> [A <sub>2</sub> ] ...
Resample Instruction	Generate a variation of the following instruction while keeping the semantic meaning. <b>Input:</b> [INSTRUCTION]\n <b>Output:</b> <COMPLETE>
Zero-shot-CoT	<b>Instruction:</b> Answer the following question. <b>Q:</b> [INPUT]\n <b>A:</b> Let's <INSERT>. [OUTPUT]

## Zero-shot 학습

- 두 가지 베이스라인과 비교
  - 인간 프롬프트 엔지니어(Human)
  - Honovich et al. (2022)에서 제안한 모델 생성 지침 알고리즘(→ "Greedy"로 참조)

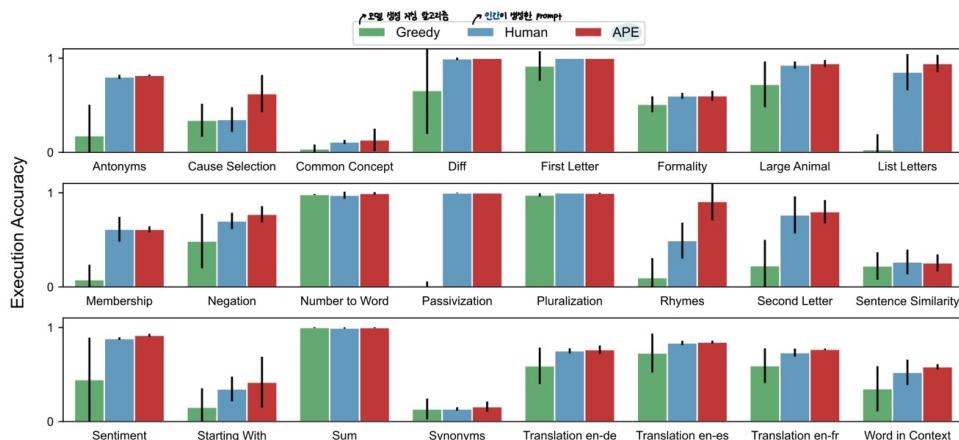


Figure 4: Zero-shot test accuracy on 24 Instruction Induction tasks. APE achieves human-level or better performance on all 24 out of 24 tasks.

- 우리의 알고리즘은 모든 작업에서 "Greedy"보다 우수한 성능을 보이며 24개 중 24개의 작업에서 인간 성능과 동등하거나 더 우수한 성능을 달성
  - 또한 24개 작업 전체에 대한 Interquartile Mean(IQM)은 [InstructGPT](#) 와 함께 **APE** 가 인간이 고안한 프롬프트를 능가한다는 것을 확인(IQM: 0.810 vs 인간: 0.794)

### Few-shot In-context 학습

- APE가 생성한 지침은 few-shot 문맥 내 학습에서 평가됨
  - 이러한 지침은 zero-shot 실행 정확도를 기반으로 선택됨
- 실험 결과, 이러한 지침을 추가하면 24개 작업 중 21개에서 표준 인컨텍스트 학습 성능과 비교 가능하거나 더 나은 테스트 성능을 얻을 수 있었음

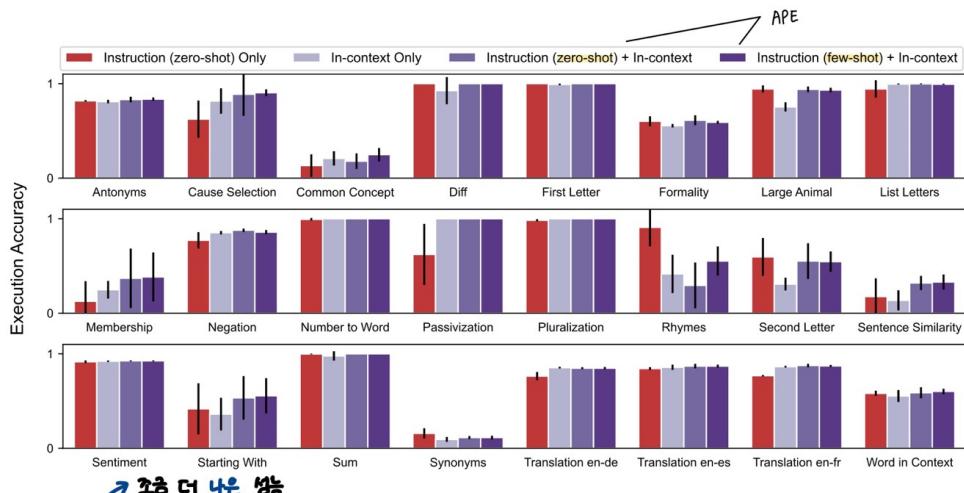


Figure 14: Few-shot in-context test accuracy of best performing instructions selected using few-shot execution accuracy on 24 Instruction Induction tasks.

- 그러나 Rhymes, Large Animal 및 Second Letters 작업에서는 문맥 내 예제 추가가 모델 성능을 저해하는 것으로 나타남

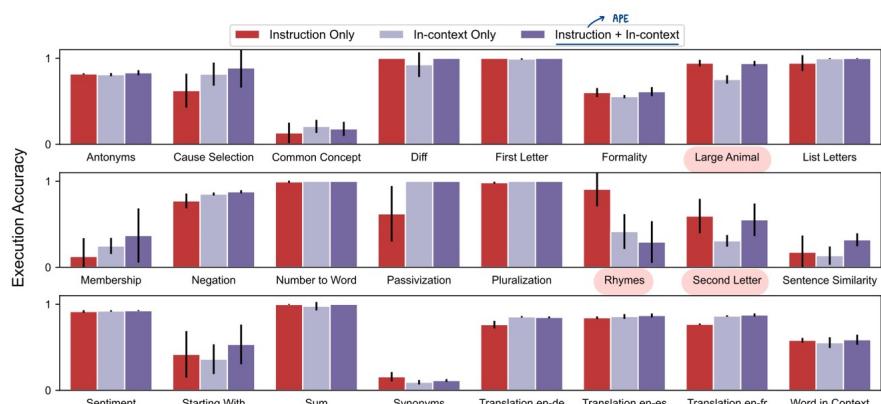


Figure 8: Few-shot in-context test accuracy on 24 Instruction Induction tasks. APE improves the few-shot in-context learning performance on 21 out of 24 tasks.

⇒ 선택된 지침이 제로샷 학습 시나리오에 과적합되어 few-shot 케이스에서 성능이 좋지 않기 때문으로 추측됨

## 4-2. BigBench

- APE는 더 어려운 작업에도 적용 가능한 자동 지침 생성 및 선택 알고리즘  
⇒ 다양한 언어 이해 작업에서 탁월한 성능을 보임
- APE가 성공적으로 작업 수행을 개선하고, 종종 인간이 작성한 것보다 더 나은 지침을 생성한다는 것을 실험적으로 입증  
→ 인간이 생성한 프롬프트와 모델이 생성한 프롬프트의 성능을 비교하고 APE가 모델을 성공적으로 조절하는 방법을 탐구함으로써 확인

Task	Normalized Performance	
	Human	APE
causal judgment	18.0	18.0
disambiguation qa	-0.4	<b>5.6</b>
dyck languages	3.0	<b>18.0</b>
epistemic reasoning	36.0	<b>38.0</b>
gender inclusive sentences german	13.0	<b>22.0</b>
implicatures	60.0	60.0
linguistics puzzles	0.0	0.0
logical fallacy detection	<b>24.0</b>	12.0
movie recommendation	-2.7	<b>12.0</b>
navigate	-8.0	<b>12.0</b>
object counting	2.0	<b>44.0</b>
operators	<b>48.0</b>	47.0
presuppositions as nli	<b>13.0</b>	5.5
question selection	-2.6	<b>-0.9</b>
ruin names	<b>1.3</b>	-14.7
snarks	2.0	<b>4.0</b>
sports understanding	36.0	36.0
tense	84.0	<b>85.0</b>
winowhy	-12.0	<b>12.0</b>
word sorting	11.0	<b>30.0</b>
word unscrambling	10.0	<b>15.0</b>

▲ 21개 작업 중 17개에서 기본 인간 프롬프트와 비교 가능하거나 더 나은 성능을 얻음

## 4-3. ZERO-SHOT CHAIN OF THOUGHT

- 해당 연구는 **자동 프롬프트 엔지니어링(APE)**를 사용하여 프롬프트를 최적화하고 자연어 모델의 성능을 향상시키는 방법을 탐구
  - 먼저 자연어 지침 후보를 생성하고 선택하는 프로세스로 이루어져 있음  
⇒ LLMs(대규모 언어 모델)의 동작을 제어하고 인간이 원하는 작업을 실행하도록 유도
  - 먼저 초기 지침 후보를 제안하고 후보 세트를 평가하여 가장 높은 점수를 가진 지침을 선택
- 실험 결과, 프롬프트 최적화가 LLMs의 성능을 향상시키는데 효과적임을 확인

- 다양한 작업에서 인간이 작성한 지침과 비교하여 **APE**가 더 나은 성능을 달성하거나 동등한 성능을 보임
- 특히, zero-shot 및 few-shot 학습, zero-shot chain of thought, 그리고 진실성 측면에서 APE가 효과적으로 작동하는 것을 확인할 수 있음  
⇒ APE는 자연어 프롬프트 엔지니어링 분야에서 유용한 도구로 나타남

#### 4-4. 진실한 QA

- APE-generated instructions를 사용하여 LLMs의 답변 스타일을 다양하게 조작하고 진실성과 정보성 간의 균형을 조사
  - 진실성, 정보성 및 두 가지를 결합한 메트릭을 최대화하는 지침을 학습하기 위해 APE를 적용
- 인간 평가와 유사한 결과를 얻기 위해 미세 조정된 GPT-judge와 GPT-info를 사용하여 성능을 평가

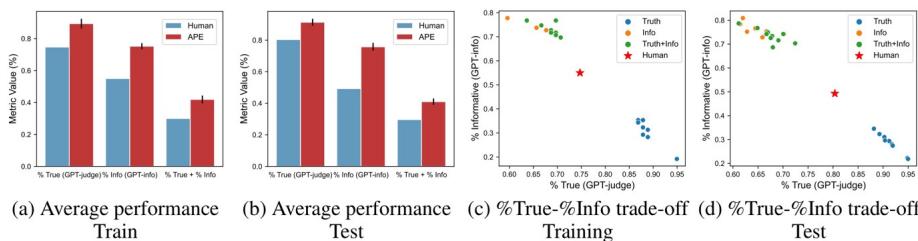


Figure 5: Comparison of APE and “help” (human) prompt on the TruthfulQA task. (a) Percentage of answers that were either true (% True), informative (% Info), or both (% True + % Info) on the 100 training examples. (b) Same data on the 717 test examples. (c) %True-%Info frontier computed on training data with top 10 instructions from each metric. (d) %True-%Info frontier on the test data.

#### TruthfulQA에서의 프롬프트 엔지니어링

- TruthfulQA 데이터셋은 제로샷 설정에서 미세 조정 모델을 테스트하기 위한 것 ⇒ APE로 최적화 된 지침을 사용하여 모델을 평가
- 학습 데모로 사용할 데이터 포인트의 작은 부분을 사용하여 지침을 최적화
- 생성된 프롬프트는 매우 일반적이며 데이터셋의 어떤 예도 포함하지 않는 “여러 질문에 답할 것입니다.”와 같은 내용임

#### 진실성 대 정보성 균형

- APE가 사람이 생성한 지침인 “도움말” 프롬프트보다 우수한 결과를 얻는 것으로 나타남
  - 상위 10개 후보를 선택하여 테스트 세트에서 잘 일반화되는 것으로 나타남

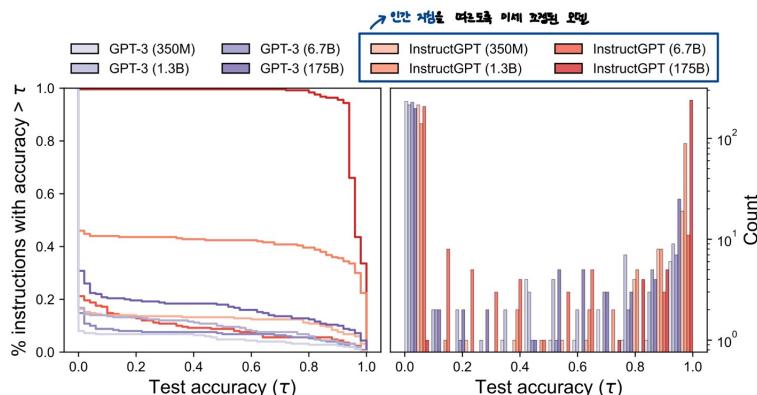
- 진실성과 정보성 사이의 균형을 조사하여 APE가 높은 진실성과 정보성을 동시에 제공하는 답변을 찾는 데 성공한 것으로 나타남

## 5. 양적 분석

- 세 가지 주요 구성 요소( = 제안 분포, 점수 함수 및 반복적인 검색)를 더 잘 이해하기 위해 양적 분석을 수행
  - 더 크고 더 강력한 언어 모델이 더 비용 효율적인 것으로 나타났으며, 토큰당 비용이 더 높더라도 최적의 프롬프트를 생성하는 데 더 비용 효율적임을 확인하기 위해 비용 분석을 수행함

### 5-1. 제안 분포를 위한 LLMs

- 제안 분포를 개선하는 데 모델 크기가 어떻게 영향을 미치는지 이해하기 위해 [OpenAI API](#)를 통해 사용 가능한 여덟 가지 다른 모델을 조사
- 모델 크기를 늘릴수록, 더 큰 모델이 더 좋은 초기 제안 분포를 생성하는 경향이 있음
- 실험 결과, 더 큰 모델과 인간 지침을 따르도록 미세 조정된 모델이 작은 모델보다 훨씬 나은 제안 분포를 생성하는 것으로 나타남
  - 단순한 작업에서는 최고 모델이 모든 지침에 합리적인 테스트 정확도를 제공하지만, 더 어려운 작업에서는 지침의 절반 이상이 부적절하며 성능이 좋지 않음



### 5-2. 선택을 위한 LLMs

- 제안 품질이 모델 크기에 어떻게 영향을 미치는지, 제안 품질이 선택 과정에서 중요한지, 그리고 어떤 점수 함수가 더 좋은지를 이해하기 위한 양적 분석을 수행
- 더 크고 강력한 언어 모델은 훨씬 효과적인 비용으로 최상의 프롬프트를 생성할 수 있다는 것을 관찰

- 제안 품질은 선택 과정에서 중요하며, 더 많은 지침을 샘플링할수록 더 나은 지침을 찾을 가능성이 높아지는 경향이 있음
- 실행 정확도 메트릭이 테스트 성능과 더 일치하는 것으로 나타났으므로, 실행 정확도(exec Acc)를 기본 메트릭으로 선택

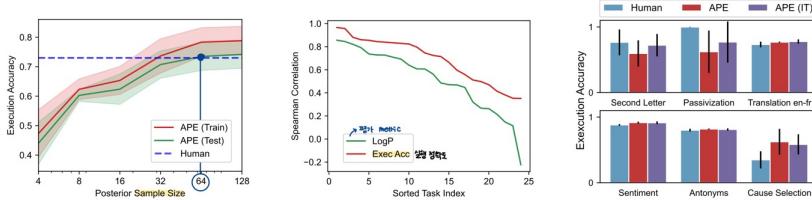
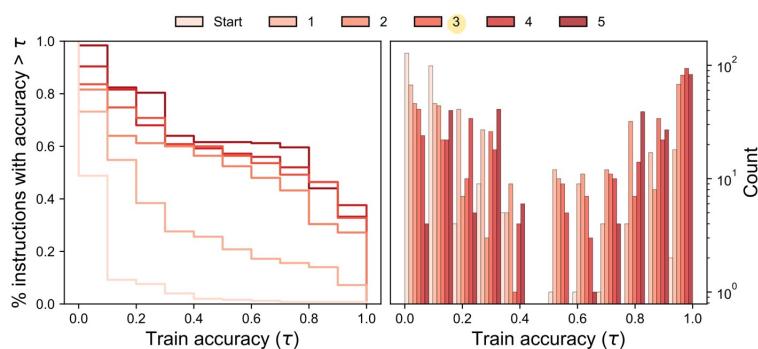


Figure 7: (Left) Test execution of the best instruction as we increase the number of instruction candidates. We report the mean and standard deviation across 6 different tasks. (Middle) Spearman Correlation between the test accuracy and two metrics on 24 tasks. (Right) Test execution accuracy of the best instruction selected using APE and iterative APE (APE (IT)).

### 5-3. 반복적인 Monte Carlo 탐색

- 반복 검색이 지침 품질을 향상시키는지 확인하기 위해 “Passivization” 작업을 포함한 여러 작업에서 생존 함수와 테스트 정확도의 히스토그램을 시각화
  - 반복 검색이 더 높은 품질의 제안 세트로 이어진다는 것을 시사
  - 품질이 세 번의 라운드 후에 안정화되는 것으로 나타남
- APE와 iterative APE를 여섯 가지 작업에서 비교
  - 결과적으로 반복 검색은 APE가 인간을 밀돌기 어려운 작업에서 약간의 성능 향상을 이루며, 다른 작업에서는 유사한 성능을 달성
  - ⇒ 초기 U(= 지시어 명령)를 생성하는 것이 어려운 작업에서 반복 검색이 가장 유용 할 것으로 예상한 가설과 일치



## 6. 결론

- 큰 언어 모델은 자연어 프롬프트로 지정된 프로그램을 실행하는 일반 목적의 컴퓨터로 볼 수 있음

- 이 프롬프트 엔지니어링 프로세스를 블랙 박스 최적화 문제로 자동화하고, LLMs(크고 강력한 언어 모델)의 안내를 받아 효율적인 검색 알고리즘을 사용하여 이를 해결하는 것을 제안
- 우리의 방법은 **최소한의 인간 입력**으로 다양한 작업에서 인간 수준의 성능을 달성
  - 최근의 LLMs는 인간 지침을 따르는 놀라운 능력을 보여주므로, 미래의 프로그램 합성을 포함한 많은 모델이 자연어 인터페이스를 가질 것으로 예상됨