

Prompting Engineering

1. 개념

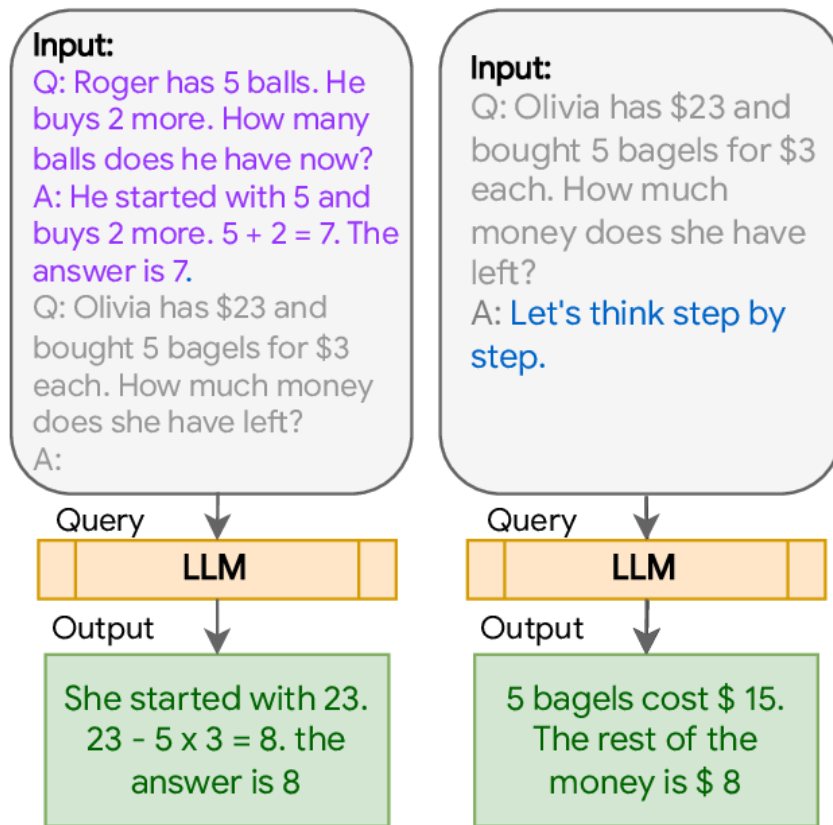
- AI가 원하는 결과를 생성하도록 가이드 하는 **최적의 프롬프트 설계**
- LLM에서 원하는 결과를 얻기 위해 **프롬프트를 정교하게 최적화**하는 과정
- **다양한 종류의 task에서 LLM의 역량을 향상**시킬 수 있음
- 복잡한 task에서도 **hallucination과 같은 오류 해소**에 기여

2. 구성요소

- Instruction 지시 : LLM 모델이 수행해야 하는 과제나 구체적 목표 제시
- Context 문맥 : 모델 정확도를 높이기 위해 사용될 수 있는 문제 해결에 필요한 배경 정보나 추가 문맥
- Input data 입력 데이터 : 모델이 답해야하는 질문
- Output Indicator 출력 지시자 : 출력 유형이나 형식과 같이 모델의 답이 어떤 방식으로 출력되어야 하는지 지정

3. 기법

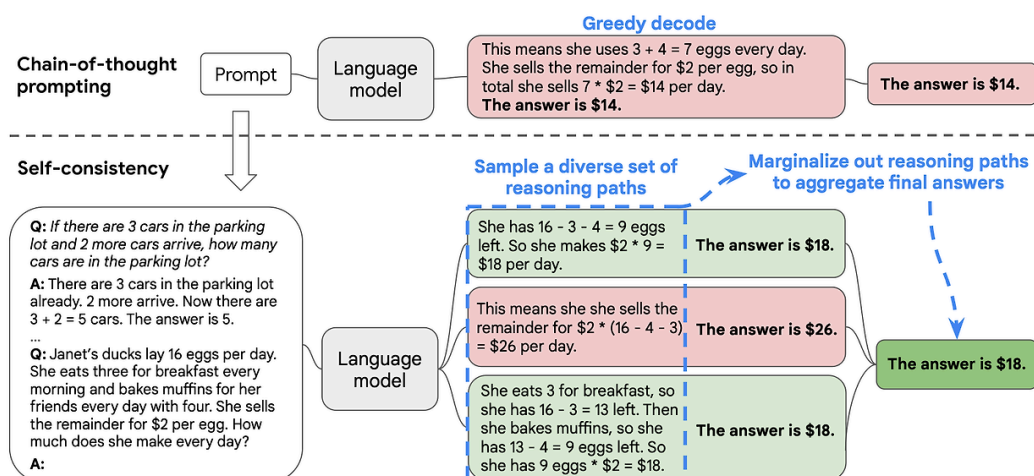
- a. Zero-Shot Prompting : task에 대한 사전 예시 없이 모델이 바로 문제 해결
One-Shot Prompting : task에 대한 1개의 예시를 제시
Few-Shot Prompting : task에 대한 2개 이상의 예시를 제시
- b. Chain of Thought(CoT) : chain of thought 과정으로 문제 해결을 하는 예시를 제시함으로써 복잡한 중간 추론 과정을 사용하여 답을 제시하도록 만듦
- c. Zero Shot Chain of Thought :
별도의 예시 없이 chain of thought 과정을 사용하도록 유도
"Let's think step by step"이라는 trigger 문장을 프롬프트에 추가
[few-shot CoT와 zero-shot CoT 프롬프트 차이]



출처 : https://www.researchgate.net/figure/Few-shot-left-and-zero-shot-right-CoT-Both-prompts-the-LLM-to-output-intermediate_fig3_370981716

- d. Self Consistency : few-shot CoT를 사용해 여러 형태의 추론 과정을 활용한 후 다수에 해당하는 답을 최종 결과로 제시하도록 만들, 모델이 더 일관성 있는 답변을 선택하도록 유도

[b와 d의 차이]



출처 : <https://velog.io/@xuio/SELF-CONSISTENCY-IMPROVES-CHAIN-OF-THOUGHT-REASONING-IN-LANGUAGE-MODELS>

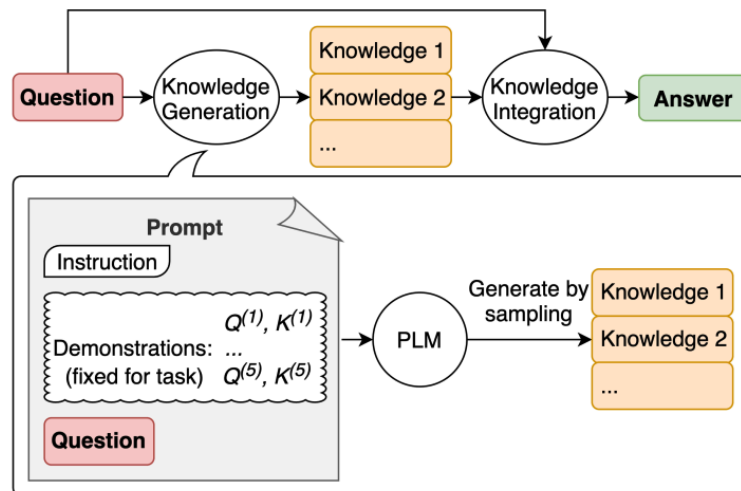
e. **Generated knowledge Prompting :**

질문과 함께 모델에서 생성된 지식이나 정보를 함께 제공하는 방식

지식(Knowledge)을 만드는 과정을 보여주는 프롬프트 예시를 제공하는 Few-Shot Prompting를 통해 원래의 질문과 관련된 Knowledge를 모델이 직접 생성

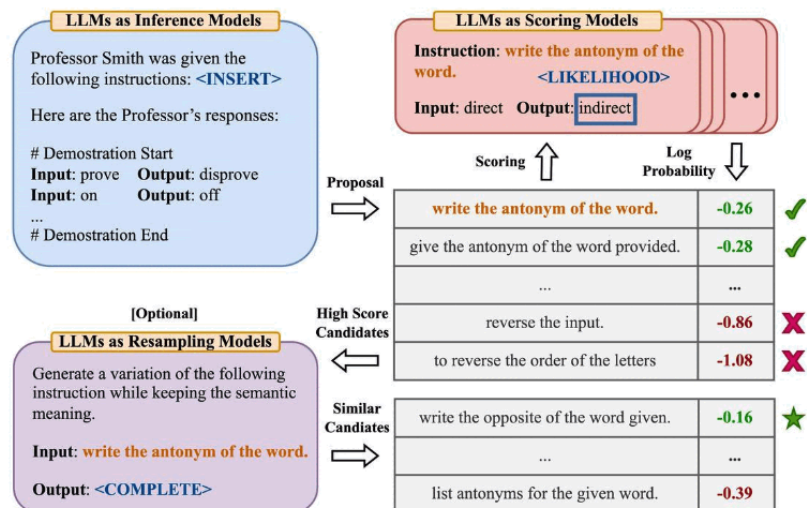
→ 기존의 질문에 생성된 Knowledge를 추가한 프롬프트를 사용함으로써 모델이 Knowledge를 사용해 답을 제시하도록 만듦

[Generated Knowledge Prompting 매커니즘]



f. **Automatic Prompt Engineering :** 사람이 아닌 모델이 스스로 프롬프트를 만들고 이를 활용해 답을 제시하는 방식, 프롬프트 내부에 <INSERT>라는 빈 칸을 만들고 answer를 함께 입력하여 모델이 <INSERT> 부분을 추론함으로써 프롬프트를 생성하도록 만듦

[Automatic Prompt Engineering 매커니즘]



g. ReAct Prompting : '생각-행동-관찰' 과정을 번갈아 수행하는 방식으로 답을 제시하도록 유도하는 방식, '생각-행동-관찰' 과정을 반복하며 주어진 질문에 대한 답을 찾아가는 문제 해결 과정을 예제로 제시하는 Few-Shot Prompting (CoT와 유사하나 생각-행동-관찰이라는 일련의 형식화된 과정을 사용한다는 차이점)

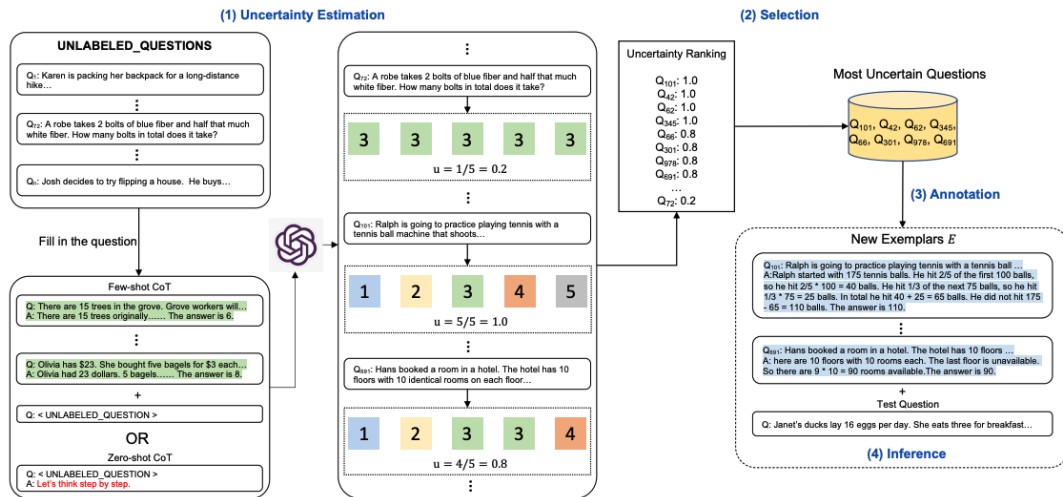
h. Active-Prompt :

여러 개의 질의 예시를 만들어 모델이 답변하게 한 후 모델의 답변에 대한 모호성 (Uncertainty)를 평가

→ 모델이 답하기 어려운 것으로 보이는 질문들만 골라 사람이 직접 답변을 라벨링한 후 Few-Shot CoT의 예제로 사용하여 다시 답변하게 만드는 방식

답변의 모호성은 모델에게 동일한 질문을 여러 번 제시한 후 disagreement를 계산하여 사용함, 더 효율적으로 질의-답변 예제를 만들 수 있음

[Active-Prompt 매커니즘]



4. 예시 [출처 : <https://www.promptingguide.ai/kr/introduction/examples>]

- Text Summarization

Prompt:

항생제는 세균 감염을 치료하기 위해 사용되는 약물의 일종입니다. 세균을 죽이거나 번식을 방지하여 신체의 면역 체계가 감염에 대항할 수 있도록 합니다. 항생제는 보통 알약, 캡슐, 액체의 형태로 경구 복용하거나 때때로 정맥 주사로 투여되기도 합니다. 항생제는 바이러스 감염에는 효과적이지 않고, 남용할 경우 항생제 내성이 생길 수 있습니다.

위 내용을 한 문장으로 요약해 줘:

Output:

항생제는 박테리아를 죽이거나 번식을 막음으로써 감염을 치료하는 데 사용되는 약이지만, 바이러스에 효과적이지 않고 과도한 사용은 항생제 내성을 초래할 수 있습니다.

- Information Extraction

Prompt:

연구 논문의 저자 기여도 진술서와 사사는 저자들이 원고 작성과 분석에 ChatGPT 등 AI 기술을 사용했는지 여부와 범위를 명확하고 구체적으로 명시해야 합니다. 또한 어떤 LLM이 사용되었는지 표기해야 합니다. 이를 통해 편집자와 심사자가 잠재적 편향, 부정확성 및 부적절한 출처 등의 문제를 고려해 원고를 보다 신중하게 검토하도록 유도할 수 있습니다. 마찬가지로, 과학 저널은 투고된 원고 선정 시 LLM 사용 여부를 투명하게 공개해야 합니다.

위 단락에서 언급한 대규모언어모델 기반 제품을 알려줘.

Output:

위 단락에서 언급한 대규모언어모델 기반 제품은 ChatGPT입니다.

- Question Answering

Prompt:

아래 문맥을 고려해서 질문에 답변해 줘. 답변은 짧고 간결하게 해 줘. 답변이 정확하지 않다면, 「확실히 않은 대답」이라고 응답해 줘.

문맥: Teplizumab은 Ortho Pharmaceutical이라는 뉴저지의 제약 회사에서 유래했다. 그곳에서, 과학자들은 OKT3라는 항체의 초기 버전을 만들어 냈다. 원래 쥐에서 유래된 이 분자는 T 세포의 표면에 결합하여 세포를 죽이는 잠재력을 제한할 수 있다. 1986년, 신장 이식 후 장기 거부 반응 예방을 위해 승인되어 인간이 사용할 수 있는 최초의 치료용 항체가 되었다.

질문: OKT3는 어디서 유래했는가?

답변:

Output:

쥐.

- Text Classification

Prompt:

문장을 **neutral**, 부정 혹은 긍정으로 분류해 줘.

문구: 그 음식은 그럭저럭이었어.

감정:

Output:

Neutral

- Reasoning

Prompt:

주어진 수 중에서 홀수를 모두 더해서 짝수를 만들어 줘: 15, 32, 5, 13, 82, 7, 1.

문제를 단계별로 해결해 봐. 먼저 주어진 수 중 홀수를 찾아낸 뒤, 그것들을 합해서, 결과가 홀수인지 짝수인지 판별해 줘.

Output:

홀수: 15, 5, 13, 7, 1
총합: 41
41은 홀수입니다.

- Conversation

Prompt:

다음은 AI 연구 보조원과의 대화입니다. 이 보조원의 어조는 기술적이고 과학적입니다.
인간: 안녕, 너는 누구니?
AI: 반갑습니다! 저는 AI 연구 보조원입니다. 오늘은 무엇을 도와드릴까요?
인간: 블랙홀 생성에 대해 설명해 줄래?
AI:

Output:

물론입니다! 블랙홀은 중력이 너무 강해서 심지어 빛조차도 벗어날 수 없는 시공간의 영역입니다. 아주 거대한 별이 죽고 그 중심핵이 스스로 붕괴하여 무한한 밀도의 특이점을 형성할 때 생성됩니다. 특이점의 강한 중력은 그 주변의 모든 물질과 방사선을 끌어당겨 블랙홀을 만듭니다.

- Code Generation

Prompt:

```
/*  
  유저의 이름을 입력받고 "안녕"이라고 해 줘  
*/
```

Output:

```
let name = prompt("이름이 뭐예요?");  
console.log(`안녕하세요, ${이름}!`);
```

[Reference]

<https://m.blog.naver.com/gptfrontier/223337739409>

<https://working-helen.tistory.com/105>

<https://modulabs.co.kr/blog/prompt-engineering>

<https://blog.naver.com/gptfrontier/223338917032>

<https://modulabs.co.kr/blog/%ED%94%84%EB%A1%AC%ED%94%84%ED%8A%B8-%EC%97%94%EC%A7%80%EB%8B%88%EC%96%B4%EB%A7%81-%EA%B8%B0%EB%B2%95%EA%B3%BC-nlp-%EC%9E%91%EC%97%85-%EB%B6%84%EB%A5%98>