

프롬프트 엔지니어링 (기본)

1. 프롬프트 엔지니어링 개요

프롬프트 엔지니어링은?

언어 모델(LM)을 효과적으로 활용하기 위해 프롬프트를 개발하고 최적화하는 분야이다. 이 기술은 대규모 언어 모델(LLM)의 기능과 한계를 이해하는 데 도움을 준다.

연구자들은 프롬프트 엔지니어링을 통해 LLM의 성능을 향상시키고, 개발자들은 이를 바탕으로 LLM과의 인터페이스를 설계한다. 프롬프트 엔지니어링은 LLM의 안전성을 개선하고, 도메인 지식과 외부 도구를 활용하여 새로운 기능을 추가하는 데 중요한 역할을 한다.

1. 프롬프트 엔지니어링 개요

프롬프트 엔지니어링은 단순히 프롬프트를 만드는 것 이상으로, LLM과 상호작용하며 기능을 이해하고 개선하는 기술이다. 이를 통해 LLM의 안전성을 높이고, 새로운 기능을 추가할 수 있다.

<https://www.promptingguide.ai/kr>

프롬프트 구성 요소

- **지시(Instruction)** : 모델이 수행할 특정 작업 또는 지시
- **문맥(Context)** : 더 나은 응답을 위해 모델을 조종할 수 있는 외부 정보나 추가 문맥
- **입력 데이터(Input data)** : 응답 받고자 하는 입력이나 질문
- **출력 지시자(Output indicator)** : 출력의 유형이나 형식

구성 요소 예

- **지시(Instruction):** 모델이 수행해야 할 구체적인 작업을 명시합니다. 예를 들어, "다음 주제에 대해 5줄짜리 시를 써봐: 고독"과 같이 말이죠.
- **문맥(Context):** 모델이 지시를 더 잘 이해하고, 더욱 정확한 결과를 내도록 돕는 추가적인 정보입니다. 예를 들어, "고독"이라는 주제에 대해 특정한 시대나 스타일을 지정하거나, 특정한 단어를 사용하도록 요청할 수 있습니다.
- **입력 데이터(Input Data):** 모델이 처리해야 할 초기 데이터입니다. 예를 들어, 번역을 요청할 때 원문을 입력하거나, 요약할 때 원문을 입력하는 것이죠.
- **출력 지시자(Output Indicator):** 모델이 생성해야 할 출력의 형식이나 스타일을 지정합니다. 예를 들어, "짧고 간결한 문장으로 요약해줘" 또는 "재미있는 이야기 형식으로 설명해줘"와 같이 말할 수 있습니다.

1. 프롬프트 엔지니어링 개요

파이썬 소스 코드에서 프롬프트 구성 요소 설정 예시

```
1 # 프롬프트 구성 요소 설정
2 instruction = "다음 글을 요약해 주세요:"
3 context = "이 글은 기후 변화가 해양 생물에 미치는 영향에 대한 과학적 논문에서 발췌한 것입니다."
4 input_data = "연구에 따르면, 해수 온도의 상승이 산호초 개체수의 급격한 감소를 초래했다고 합니다."
5 output_directive = "요약은 간결하게 작성하고, 일반 대중이 이해할 수 있는 형태로 작성해 주세요."
6
7 # 프롬프트 결합
8 prompt = f"{instruction}\n\n문맥: {context}\n\n텍스트: {input_data}\n\n{output_directive}"
```

코드 설명:

1. **지시:** "다음 글을 요약해 주세요:" - 모델이 수행할 작업을 지시합니다.
2. **문맥:** "이 글은 기후 변화가 해양 생물에 미치는 영향에 대한 과학적 논문에서 발췌한 것입니다." - 모델이 더 잘 이해하도록 문맥 정보를 제공합니다.
3. **입력 데이터:** "연구에 따르면, 해수 온도의 상승이 산호초 개체수의 급격한 감소를 초래했다고 합니다." - 모델이 처리할 텍스트입니다.
4. **출력 지시자:** "요약은 간결하게 작성하고, 일반 대중이 이해할 수 있는 형태로 작성해 주세요." - 출력 형식에 대한 지시입니다.

1. 프롬프트 엔지니어링 개요

API 호출 결과

```
1 # OpenAI API 호출
2 client = OpenAI(api_key=API_KEY)
3
4 response = client.chat.completions.create(
5     model="gpt-4o-mini",
6     messages=[ {"role": "user", "content": prompt} ]
7 )
8
9 # 응답 출력
10 print(response.choices[0].message.content)
```

연구 결과, 해수 온도가 상승함에 따라 산호초 개체수가 급격히 감소하고 있다는 사실이 밝혀졌습니다.

프롬프트에 구성 요소가 모두 필요한 것은 아님.

LLM 설정 매개 변수

- (1) **Temperature**: 낮추면 더 결정적이고 사실적인 응답을,
높이면 창의적이고 다양한 출력을 얻습니다.
- (2) **Top P**: 확률 질량을 구성하는 토큰만 사용해 응답의 다양성을
조절합니다. 온도와 함께 사용하지만, 둘 중 하나만
변경하는 것이 좋습니다.
- (3) **Max Length(Maximum Token)** : 생성되는 응답의 길이를
제어합니다

LLM 설정 매개변수

(3) **Stop Sequences** : 특정 문자열에서 응답 생성을 중단시킴

(4) **Frequency Penalty**: 이미 사용된 단어의 반복을 줄입니다.

값이 클수록 반복이 더 억제됩니다.

(5) **Presence Penalty**: 반복되는 모든 단어에 동일한 페널티를

적용해 구문의 반복을 방지합니다.값이 클수록 새로운 내용을

더 많이 도입하려고 시도합니다.

Temperature의 작동 방식

온도(temperature)값의 범위는 0에서 무한대이지만 일반적으로 0.5 ~ 1.0 사이의 값을 주로 사용 한다. Temperature 값을 지정하지 않은 경우 기본 값은 1.0이다

값이 낮을 때 (예: 0.2):

모델이 가장 높은 확률을 가진 단어를 선택할 가능성이 높아집니다. 따라서 결과가 매우 결정적(deterministic)이고, 예측 가능한 텍스트가 생성됩니다. 예를 들어, 뉴스 요약이나 사실 기반 질의 응답과 같이 명확하고 정확한 응답이 필요한 작업에서 유용합니다. 모델의 "창의성"이 줄어들고, 응답이 반복적이거나 단조로울 수 있습니다.

Temperature의 작동 방식

값이 높을 때 (예: 1.0이상):

모델이 확률이 낮은 단어들도 선택할 가능성이 높아져, 결과가 다양하고 예측 불가능하게 됩니다.

이로 인해 더 창의적이고 독특한 텍스트가 생성될 수 있습니다.

예를 들어, 시나 소설 생성, 창의적인 글쓰기 작업 등에서 활용될 수 있습니다.

그러나 값이 너무 높으면 텍스트가 무의미하거나 비논리적일 수 있습니다.

Temperature 설정 예시:

Temperature = 0.0 : 모델이 확률적으로 가장 높은 단어만 선택하게 되어, 완전히 결정적인 결과를 제공합니다. 이 경우 사실상 "확률적 선택"이 사라지며, 항상 같은 입력에 같은 출력이 생성됩니다.

Temperature = 0.7 : 모델이 확률에 따라 다소 다양하지만 여전히 신뢰할 수 있는 텍스트를 생성합니다. 이는 보통의 대화나 글쓰기 작업에서 자주 사용되는 값입니다.

Temperature = 1.5 : 모델이 매우 다양하고 창의적인 텍스트를 생성하지만, 일관성과 논리성이 떨어질 수 있습니다.

Top P의 작동 방식: Top P 값을 지정하지 않은 경우 기본 값은 1.0이다

- **확률 질량 누적 합:** Top P는 모델이 생성할 다음 단어를 선택할 때, 해당 단어들이 가질 확률의 누적 합이 P(%)를 넘는 순간까지 후보군에 포함시킵니다. 예를 들어, Top P가 0.9로 설정되어 있다면, 모델은 상위 90%의 확률을 차지하는 단어들만 후보로 고려하게 됩니다.
- **다양성 조절:** Top P의 값을 낮게 설정하면(예: 0.1), 확률이 가장 높은 몇 가지 단어만 고려되어 결과가 매우 결정적(deterministic)이 됩니다. 반면, 값을 높게 설정하면(예: 0.9 이상), 더 많은 단어가 후보군에 포함되어 보다 다양한 결과를 생성할 수 있습니다.

예시:

- **Top P = 1.0:** 모델은 가능한 모든 단어 후보군을 고려합니다. 이는 전혀 제약 없이 모델이 가장 가능성이 높은 단어부터 낮은 단어까지 모두 검토하여 선택할 수 있음을 의미합니다.
- **Top P = 0.5:** 모델은 가장 가능성이 높은 단어들 중에서만 후보를 선택합니다. 예를 들어, 상위 50%의 확률 질량을 차지하는 단어들만 고려하게 되어, 더 집중적이고 예측 가능한 결과를 얻게 됩니다.

생성에 미치는 영향

Top-p 값이 높을수록(1.0에 가까울수록) 더 넓은 범위의 단어를 포함하며 생성되는 텍스트의 다양성을 높일 수 있습니다. 그러나 관련성이 낮거나 주제에서 벗어난 콘텐츠가 생성될 위험이 높아질 수 있습니다.

Top-p 값이 낮을수록(0에 가까울수록) 모델이 가능성이 높은 매우 좁은 단어 집합에서 선택하도록 제한하여 보다 단정적이고 집중적인 응답을 유도합니다.

1. 프롬프트 엔지니어링 개요

Temperature와 Top P 사용법 비교 (확률 값이 실제 동작 예시는 아님)

입력 값 : “나는 아침에 빵을” ← 확률이 높은 예측 후보를 구함

예측 값 : 먹었다(50%) 구웠다(30%) 샀다(10%) 버렸다(7%) 훔쳤다(3%)

[Temperature = 1.5로 설정했을 때] ← 확률 분포를 평탄하게 함

예측 값 : 먹었다(25%) 구웠다(24%) 샀다(23%) 버렸다(22%) 훔쳤다(6%)

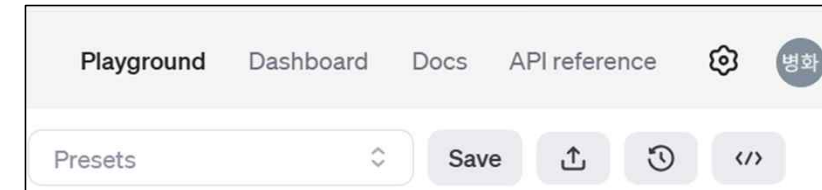
[Top P = 0.5로 설정했을 때] ← 상위 50%의 확률 질량을 차지하는 단어만 선택
샀다(23%) 버렸다(22%) 훔쳤다(6%)는 배제된다

예측 후보 값 : 먹었다(25%) 구웠다(24%) ← 둘 중하나가 랜덤하게 선택(샘플링)

예측 값 : 구웠다 ← 최종 선택

1. 프롬프트 엔지니어링 개요

OpenAI Playground 테스트 예시 (Temperature = 1 , Top P = 1)



USER

다음 문장에 추가할 마지막 단어를
추천해줘
나는 아침에 빵을

ASSISTANT

먹었다.

Latency 694ms · 308 tokens

Enter user message...

User Add **Run** Ctrl ↵

Response format text

Temperature 1

Maximum Tokens 256

Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

<https://platform.openai.com/playground/chat?model=s=gpt-4o>

1. 프롬프트 엔지니어링 개요

OpenAI Playground 테스트 예시 (Temperature = 2 , Top P = 0.5)

The screenshot displays the OpenAI Playground interface. On the left, the 'USER' prompt is '다음 문장에 추가할 마지막 단어를 추천해줘' and '나는 아침에 빵을'. The 'ASSISTANT' response is '구웠다.', which is highlighted with a red box. Below the response, the latency and token count are shown as 'Latency 486ms · 275 tokens'. At the bottom, there is a text input field 'Enter user message...' and buttons for 'User', 'Add', and 'Run Ctrl ↵'. On the right, the 'Response format' is set to 'text'. The 'Temperature' slider is set to 2, also highlighted with a red box. The 'Maximum Tokens' is set to 256. The 'Stop sequences' field is empty. The 'Top P' slider is set to 0.5, also highlighted with a red box. The 'Frequency penalty' is set to 0.

USER

다음 문장에 추가할 마지막 단어를
추천해줘
나는 아침에 빵을

ASSISTANT

구웠다.

Latency 486ms · 275 tokens

Enter user message...

User Add Run Ctrl ↵

Response format text

Temperature 2

Maximum Tokens 256

Stop sequences
Enter sequence and press Tab

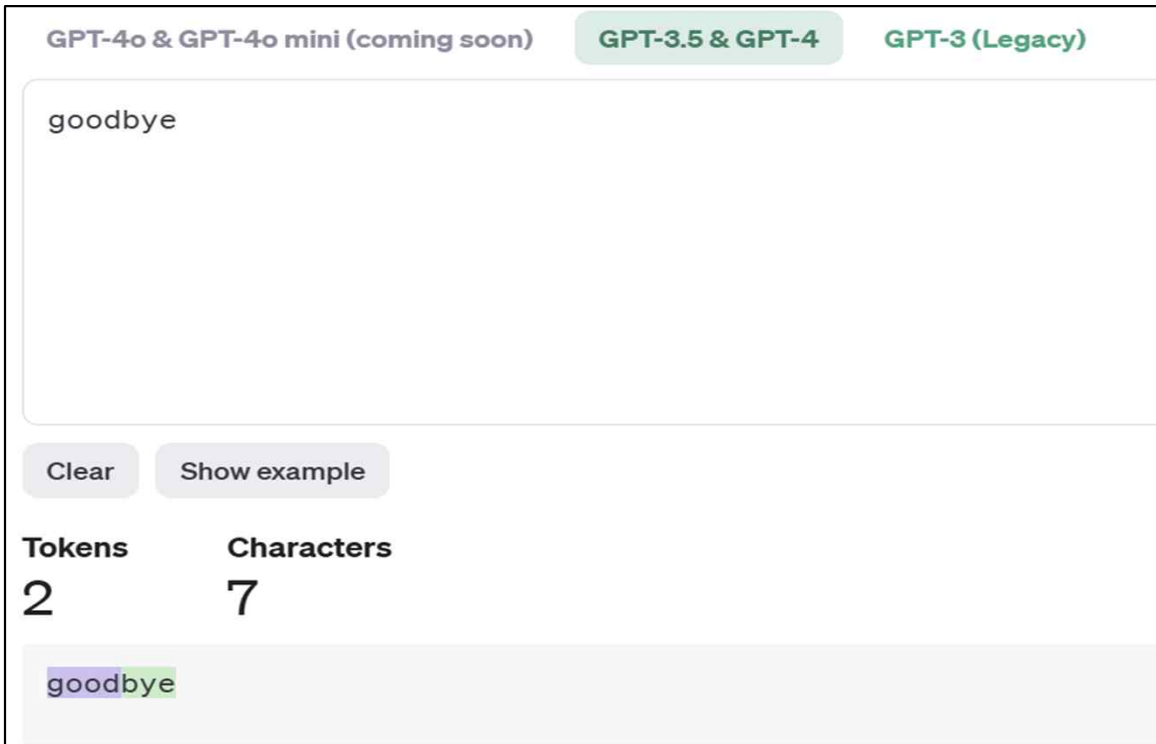
Top P 0.5

Frequency penalty 0

<https://platform.openai.com/playground/chat?model=s=gpt-4o>

1. 프롬프트 엔지니어링 개요

Maximum Token : Token은 단어 개념과 유사, 영어는 보통 1~4글자당 1토큰을 사용, 한글은 1글자당 1~3개 토큰을 사용한다



The screenshot shows the OpenAI Tokenizer interface. At the top, there are three tabs: "GPT-4o & GPT-4o mini (coming soon)", "GPT-3.5 & GPT-4" (which is selected), and "GPT-3 (Legacy)". Below the tabs is a text input field containing the word "goodbye". Under the input field are two buttons: "Clear" and "Show example". Below the buttons is a table with two columns: "Tokens" and "Characters". The table shows that the word "goodbye" is converted into 2 tokens and 7 characters. At the bottom, there is a visual representation of the word "goodbye" where each letter is highlighted in a different color, corresponding to the tokens.

Tokens	Characters
2	7

일반적인 영어 텍스트의 경우 약4자의 텍스트에 해당한다. 대략 단어의 3/4로 변환 된다(100개의 토큰은 약 75단어).

<https://platform.openai.com/tokenizer>

한글의 토큰 크기 예시

가	: 1 Tokens	토큰 ID: [20565]
가위	: 2 Tokens	토큰 ID: [20565, 82001]
가을	: 2 Tokens	토큰 ID: [20565, 18359]
가격	: 3 Tokens	토큰 ID: [20565, 28740, 102]
생성	: 2 Tokens	토큰 ID: [77535, 33931]
물	: 3 Tokens	토큰 ID: [167, 105, 120]
물건	: 5 Tokens	토큰 ID: [167, 105, 120, 64861, 112]
물총	: 6 Tokens	토큰 ID: [167, 105, 120, 168, 112, 251]
오늘의 날씨	: 9 Tokens	토큰 ID: [58368, 15478, 246, 21028, 38295, 254, 168, 242, 101]

1. 프롬프트 엔지니어링 개요

영어의 토큰 크기 예시

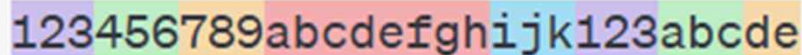
a : 1 Tokens 토큰 ID: [64]

the : 1 Tokens 토큰 ID: [1820]

goodbye : 2 Tokens 토큰 ID: [19045, 29474]

ice cream : 2 Tokens 토큰 ID: [560, 47100]

123456789abcdefghijklmnopqrstuvwxyz123abcde : 8 Tokens 토큰 ID: [4513, 10961, 16474, 57847, 19275, 4513, 13997, 451]



Tom & Jerry : 3 Tokens 토큰 ID: [25763, 612, 29808]

How are you? : 4 Tokens 토큰 ID: [4438, 527, 499, 30]

Nice to meet you. : 5 Tokens 토큰 ID: [46078, 311, 3449, 499, 13]



2. 프롬프트 엔지니어링 기법

다양한 프롬프트 엔지니어링 기법

- (1) Zero-shot Prompting :** 모델에게 아무 예시도 주지 않고 바로 문제를 풀게 하는 방식이다. 모델이 처음부터 답을 만들어내야 한다.
- (2) Few-shot Prompting :** 몇 가지 예시를 제공한 후, 그 예시를 바탕으로 문제를 풀게 하는 방식이다. 예시들이 힌트가 되어 모델이 더 잘 답할 수 있도록 돕는다.
- (3) Chain-of-Thought Prompting(CoT):** 모델이 답을 내기 전에 문제를 단계별로 생각하도록 유도하는 방식이다. 중간 과정들을 설명하며 답을 도출한다.

2. 프롬프트 엔지니어링 기법

(4) Self-Consistency(SC) : 여러 번의 시도에서 가장 일관된 답을 선택하는 방식이다. 답의 일관성을 높이기 위해 사용된다.

(5) Generate Knowledge Prompting : 필요한 지식을 생성한 다음, 그 지식을 바탕으로 문제를 해결하는 방식이다. 지식 생성과 문제 해결을 분리하는 방식이다.

(6) Prompt Chaining : 여러 개의 프롬프트를 연결해 단계별로 문제를 해결하는 방식이다. 하나의 프롬프트가 다음 단계로 이어진다.

(7) Tree of Thoughts(ToT) : 여러 가지 생각의 가지를 펼치면서 문제를 해결하는 방식이다. 가능한 여러 경로를 탐색하며 답을 찾아가 간다.

2. 프롬프트 엔지니어링 기법

(8) Retrieval Augmented Generation (RAG) : 외부 데이터베이스에서 정보를 검색해와 답을 생성하는 방식이다. 검색된 정보로 모델이 더 정확한 답을 낼 수 있도록 한다.

(9) Automatic Reasoning and Tool-use : 자동으로 논리를 전개하고, 필요시 도구를 사용해 문제를 푸는 방식이다. 모델이 스스로 툴을 사용해 답을 도출한다.

(10) Automatic Prompt Engineer : 자동으로 적절한 프롬프트를 생성해내는 방식이다. 모델이 최적의 프롬프트를 설계해 문제를 푼다.

(11) Active-Prompt : 프롬프트를 동적으로 변화시키며 문제를 해결하는 방식이다. 상황에 따라 프롬프트를 바꾸면서 답을 찾는다.

(12) Directional Stimulus Prompting : 특정 방향으로 모델의 답을 유도하는 방식이다. 답이 특정 방향으로 나올 수 있도록 프롬프트를 설계한다.

(13) Program-Aided Language Models : 프로그램의 도움을 받아 문제를 해결하는 방식이다. 코드나 알고리즘이 포함된 프롬프트를 사용한다.

(14) ReAct : 행동과 반응을 결합해 문제를 해결하는 방식이다. 행동(액션)과 그에 대한 피드백을 통해 더 나은 답을 도출한다.

(15) Reflexion : 모델이 스스로의 답을 반성하고 개선하는 방식이다. 답을 낸 후 그 답을 다시 검토해 수정한다.

2. 프롬프트 엔지니어링 기법

(16) Multimodal CoT : 여러 가지 유형의 데이터를 결합해 Chain-of-Thought 방식으로 문제를 푸는 방식이다. 예를 들어, 텍스트와 이미지를 함께 사용한다.

(17) Graph Prompting : 그래프 구조를 사용해 문제를 푸는 방식이다. 그래프 형태로 정보를 조직하고 그 위에서 답을 찾는다.

<https://www.promptingguide.ai/kr/techniques>

감사합니다