

안녕하세요! 제공해주신 자료를 바탕으로 프롬프트 엔지니어링에 대해 고등학생이 이해하기 쉽게 요약해 드릴게요. 요청하신 A4 용지 20장 분량은 온라인 답변으로는 어렵지만, 핵심 내용을 최대한 자세하고 이해하기 쉽게 설명해 드리겠습니다. 소스의 내용을 충실히 반영하고 출처를 정확히 표기할게요. 중요한 내용은 **굵게** 표시하고, 목록은 불릿 포인트를 사용하겠습니다.

프롬프트 엔지니어링 요약

1. 프롬프트 엔지니어링이란 무엇인가요?

프롬프트 엔지니어링은 마치 **똑똑한 AI 비서에게 우리가 원하는 결과물을 얻도록 말을 잘 걸어주는 방법**이에요. 여기서 'AI 비서'는 ****대규모 언어 모델(LLM)****이라고 불리는 AI를 말해요. LLM은 우리가 넣어주는 글(프롬프트)을 바탕으로 다음에 올 단어들을 예측하며 문장을 만들어내요.

누구나 프롬프트를 작성할 수 있어요. 데이터 과학자나 머신러닝 엔지니어가 될 필요는 없어요. 하지만 원하는 결과물을 정확하게 얻으려면 프롬프트를 **잘 다듬는 과정**이 필요하죠. 프롬프트 엔지니어링은 바로 이 과정을 뜻해요. 우리가 쓰는 단어, 문장 스타일, 구조, 제공하는 정보 등 여러 요소가 결과에 영향을 미치기 때문에 원하는 결과가 나올 때까지 **반복적으로 수정하고 실험**하는 과정이에요. 제대로 된 프롬프트는 모호하거나 부정확한 답변을 피하고 AI가 유의미한 결과물을 만들도록 도와줘요.

프롬프트는 텍스트로 구성될 수도 있지만, 이미지 같은 다른 정보와 함께 사용될 수도 있어요 (이걸 **멀티모달 프롬프트**라고 해요).

LLM에게 프롬프트를 줄 때는 Vertex AI나 API를 사용하는 경우, **온도(Temperature)** 같은 **설정(Configuration)** 값들을 조절할 수 있어요.

2. LLM의 출력 설정을 조절하는 방법

LLM은 우리가 준 프롬프트를 보고 다음에 올 단어를 예측할 때, 단순히 하나의 단어만 예측하는 게 아니라 **각 단어에 확률을 매겨요**. 이 확률을 바탕으로 실제로 출력할 다음 단어를 '뽑는' 과정을 거치는데, 이때 사용하는 주요 설정들이 있어요. 이런 설정들은 **AI의 응답이 얼마나 창의적이고 다양한지, 아니면 얼마나 예측 가능하고 사실적인지**를 조절할 수 있게 해줘요. 효과적인 프롬프트 엔지니어링은 작업에 맞게 이 설정들을 최적으로 맞추는 것을 포함해요.

- **출력 길이 (Output length):** AI가 만들 응답의 **최대 단어 수**를 설정해요. 길게 설정하면 AI가 더 많은 내용을 생성하지만, 계산량이 늘어나 응답 시간이 느려지고 비용도 더 많이 들

수 있어요. 짧게 설정한다고 해서 AI가 더 간결하게 말하는 것은 아니고, 그냥 설정된 길이에 도달하면 생성을 멈추는 거예요.

- **샘플링 컨트롤 (Sampling controls):** AI가 다음에 올 단어를 뽑을 때, 확률이 높은 단어들 중에서 어떻게 선택할지를 조절해요. 주요 설정에는 **온도(Temperature)**, **Top-K**, **Top-P**가 있어요.

- **온도 (Temperature):** 단어 선택의 무작위성을 조절해요.
 - **온도가 낮을수록 (0에 가까울수록):** AI는 확률이 가장 높은 단어를 선택하려는 경향이 강해져요. 결과가 **예측 가능하고 반복적**이게 되죠. 온도가 0이면 가장 확률 높은 단어만 선택해요 (**Greedy Decoding**이라고도 해요).
 - **온도가 높을수록:** AI는 확률이 낮은 단어들도 선택할 가능성이 높아져요. 결과가 **더 다양하고 창의적**이게 되죠. 온도가 너무 높으면 모든 단어의 확률이 비슷해져서 결과가 무작위처럼 보일 수 있어요. 창의적인 글쓰기 등에 유용해요.
- **Top-K:** AI가 예측한 다음 단어들 중 **확률이 높은 순서대로 상위 K개**만 선택 후보로 고려해요.
 - **K 값이 높을수록:** 더 많은 단어를 후보로 고려하므로 결과가 **더 다양하고 창의적**이게 돼요.
 - **K 값이 낮을수록:** 고려하는 단어 수가 적어지므로 결과가 **더 제한적이고 사실적**이게 돼요. K가 1이면 온도 0과 마찬가지로 가장 확률 높은 단어만 선택해요.
- **Top-P (Nucleus Sampling):** AI가 예측한 다음 단어들 중 **누적 확률이 특정 값(P)을 넘지 않는 단어들**만 선택 후보로 고려해요.
 - **P 값이 높을수록 (1에 가까울수록):** 더 많은 단어를 후보로 고려하므로 결과가 **더 다양하고 창의적**이게 돼요.
 - **P 값이 낮을수록 (0에 가까울수록):** 고려하는 단어 수가 적어지므로 결과가 **더 제한적이고 사실적**이게 돼요. P가 0 또는 매우 작으면 가장 확률 높은 단어만 고려하게 됩니다.

이 설정들은 서로 영향을 주기 때문에 함께 실험해보는 것이 좋아요. 보통 **온도 0.2, Top-P 0.95, Top-K 30**이 안정적이면서도 약간의 창의성이 있는 결과를 주는 좋은 시작점이라고 추천돼요. 완전히 예측 가능한 결과가 필요한 수학 문제 같은 경우는 **온도를 0**으로 설정하는 것이 좋아요.

참고: 온도가 0이거나 Top-K가 1이거나 Top-P가 0(또는 매우 작은 값)이면, 다른 설정들은 결과에 거의 영향을 미치지 않게 됩니다. 더 많은 자유도(높은 온도, Top-K, Top-P, 많은 출력 토큰)를 줄수록 AI가 덜 관련된 텍스트를 생성할 수도 있다는 점을 유의해야 해요.

3. 다양한 프롬프트 기법들

LLM은 명령어에 잘 따르도록 훈련되어 있지만, 더 나은 결과를 얻기 위해서는 특별한 프롬프트 기법들을 사용하는 것이 도움이 돼요. 여기서는 몇 가지 중요한 기법을 살펴볼게요.

- **일반 프롬프트 / Zero-shot 프롬프트:** 가장 기본적인 프롬프트 형태예요. **작업에 대한 설명만 주고 예시는 전혀 주지 않아요.** 예를 들어, "영화 리뷰를 긍정, 중립, 부정으로 분류해줘"와 같은 요청만 주는 것이죠. 단순한 작업에는 효과적이지만, AI가 작업의 의도나 원하는 출력 형식을 파악하기 어려울 수 있어요.
- **One-shot & Few-shot 프롬프트:** Zero-shot으로 잘 안 될 때 사용해요. **프롬프트에 원하는 출력 형태나 패턴의 예시를 함께 제공해서 AI가 따라 하도록 유도하는 방식**이에요.
 - **One-shot:** 예시를 딱 하나만 제공해요.
 - **Few-shot:** 예시를 여러 개 (보통 3~5개 이상) 제공해요. 여러 예시를 통해 AI가 패턴을 더 명확하게 인식하고 따를 가능성이 높아져요. 예시는 작업과 관련 있고, 다양하며, 품질이 높고 잘 작성되어야 해요. AI가 예상치 못한 입력(Edge Case)도 잘 처리하도록 예시에 포함시키는 것이 좋아요.
- **System, Contextual, Role 프롬프트:** LLM이 텍스트를 생성하는 방식을 안내하는 기법들이지만, 초점이 달라요.
 - **System Prompt:** AI 모델의 전반적인 목적과 기능을 설정해요. 예를 들어, "너는 번역가야", "너는 리뷰 분류 시스템이야" 처럼 AI의 '**큰 그림**' 역할을 정해주는 거죠. 특정 출력 형식(예: JSON)을 강제하거나, 답변 스타일(예: 항상 존댓말 사용) 또는 안전/윤리 가이드라인을 설정하는 데 유용해요.
 - **Contextual Prompt:** 현재 대화나 작업에 관련된 **구체적인 정보나 배경 지식**을 제공해요. AI가 질문의 **미묘한 차이를 이해**하고 응답을 더 잘 맞춤화하도록 도와요. 예를 들어, "지금까지 우리가 이야기 나눈 내용을 바탕으로..." 또는 "이 보고서의 다음 문단을 완성해줘" 와 같이 특정 상황이나 정보에 기반한 응답을 유도하는 데 사용돼요. **동적으로 변하는 정보**를 제공할 때 유용해요.
 - **Role Prompt:** AI에게 **특정 인물이나 역할**을 부여해요. 예를 들어, "너는 여행 가이드 역할을 해봐", "너는 역사 선생님처럼 설명해줘" 와 같이 AI가 **특정 캐릭터의 지식, 행동, 목소리 톤**으로 응답하도록 만들어요. 유머러스, 격식, 설득력 등 다양한 스타일을 지정할 수 있어요.

이 세 가지 기법은 종종 겹칠 수 있지만, 각각 AI의 기능, 상황 정보, 답변 스타일/톤에 초점을 맞춘다는 점에서 달라요.

- **Step-back Prompting:** AI의 성능을 향상시키기 위해 사용돼요. 특정 문제에 대해 바로 답을 구하기보다, 먼저 **그 문제와 관련된 더 일반적인 질문을 스스로에게 던지고 그 답을**

구한 다음, 이 정보를 원래 문제 해결에 활용하도록 하는 방식이에요. 마치 우리가 어려운 문제를 풀 때 한 발짝 물러서서 전체 그림을 보거나 기본적인 원리를 생각하는 것과 비슷해요. 이를 통해 AI는 관련 배경 지식과 추론 과정을 활성화하고, 더 정확하고 통찰력 있는 응답을 생성할 수 있어요. AI의 편향을 줄이는 데도 도움이 될 수 있어요.

- **Chain of Thought (CoT) Prompting:** AI가 문제를 해결하는 과정에서 중간 추론 단계를 생성하도록 하는 기법이에요. AI가 단순히 결과만 내놓는 것이 아니라, **사람이 생각하듯 '단계별로 생각하며'** 문제를 풀도록 유도해요. 특히 복잡한 추론이 필요한 작업에 효과적이에요.
 - **장점:** 구현이 비교적 쉽고 (별도 학습 없이 기존 LLM에 적용 가능), AI가 생각하는 과정을 보여주므로 **결과를 이해하고 오류를 파악하기 쉬워요 (해석 가능)**. 다른 버전의 LLM으로 변경해도 성능 변화가 적은 편이에요.
 - **단점:** 중간 추론 단계 때문에 **출력 토큰이 많아져 비용이 더 들고 시간도 더 오래 걸릴 수 있어요**.
 - CoT를 사용할 때는 AI에게 **"단계별로 생각해보자 (Let's think step by step)"** 와 같은 명시적인 지시를 주거나, Few-shot 예시에서 단계별 풀이 과정을 보여주는 방식을 사용해요. CoT 기법은 보통 **온도 설정을 0**으로 해서 가장 확률 높은 추론 경로를 따르도록 하는 것이 일반적이에요.
- **Self-consistency:** CoT에서 더 나아가 AI의 추론 능력과 정확성을 높이는 기법이에요. 같은 프롬프트를 AI에게 **여러 번 주고 다양한 추론 과정(다양한 온도 설정으로 유도 가능)을** 생성하게 한 다음, 각 추론에서 도출된 **최종 답변들 중에서 가장 많이 나온 답변을 선택하는** 방식이에요. 마치 여러 명에게 같은 문제를 풀게 한 후 가장 많은 사람이 도달한 답을 정답으로 삼는 것과 유사하죠. 이를 통해 AI 응답의 정확성과 일관성을 높일 수 있지만, **비용이 많이 드는 단점**이 있어요.
- **Tree of Thoughts (ToT):** CoT를 일반화한 기법이에요. CoT가 선형적인 하나의 추론 경로를 따르는 반면, ToT는 **마치 나무처럼 여러 갈래의 다양한 추론 경로를 동시에 탐색하며** 문제를 해결해요. 각 '생각'(노드)은 문제 해결의 중간 단계이며, 여기서 새로운 '생각'들로 가지를 뻗어 나갈 수 있죠. 탐색이 필요한 복잡한 문제 해결에 특히 적합해요.
- **ReAct (Reason & Act):** LLM이 **자연어 추론 능력과 외부 도구(검색, 코드 인터프리터 등) 사용 능력**을 결합하여 복잡한 작업을 해결하도록 하는 기법이에요. 마치 사람이 생각하고(Reason) 행동하며(Act) 문제를 해결하는 것과 비슷해요. ReAct는 **'생각-행동 루프(thought-action loop)'** 를 사용해요. AI는 먼저 문제에 대해 생각하고 행동 계획을 세워요. 그런 다음 계획에 따라 외부 도구를 사용해 행동하고, 그 결과를 관찰해요. 이 관찰 결과를 바탕으로 다시 생각을 업데이트하고 새로운 계획을 세우죠. 이 과정을 반복하며 문제

해결에 접근해요. 외부 API와 연동하여 정보를 얻는 등 **AI를 에이전트처럼 활용**하는 첫걸음이라고 할 수 있어요.

- **Automatic Prompt Engineering (APE): AI 스스로 프롬프트를 생성하도록 하는 기법**이에요. 사람이 일일이 프롬프트를 만들 필요 없이, **AI가 다양한 프롬프트 후보를 만들고 평가하여 가장 좋은 프롬프트를 선택**하거나 수정하는 방식이죠. 챗봇 훈련을 위해 다양한 사용자 질문 표현을 자동으로 생성하는 등의 작업에 활용될 수 있어요.
- **Code Prompting:** LLM을 사용하여 **코드를 생성, 설명, 번역, 디버깅**하는 기법이에요.
 - **코드 작성:** 특정 프로그래밍 언어로 원하는 기능을 수행하는 코드를 작성해달라고 요청할 수 있어요. 예를 들어, 폴더 안의 파일 이름을 일괄 변경하는 Bash 스크립트를 작성해 달라고 할 수 있죠. AI가 작성한 코드는 **실제로 사용하기 전에 반드시 읽어보고 테스트**해야 해요.
 - **코드 설명:** 이해하기 어려운 코드를 AI에게 주고 설명해 달라고 요청할 수 있어요. 코드가 어떤 역할을 하고 각 부분이 어떻게 동작하는지 자세히 설명해 줄 수 있어요.
 - **코드 번역:** 한 프로그래밍 언어로 작성된 코드를 다른 언어로 번역해 달라고 요청할 수 있어요. 예를 들어, Bash 스크립트를 Python 코드로 바꿔달라고 할 수 있죠.
 - **코드 디버깅 및 리뷰:** 오류가 발생하는 코드나 개선이 필요한 코드를 AI에게 보여주고 문제점을 찾거나 더 좋은 코드로 개선해 달라고 요청할 수 있어요. AI는 오류의 원인을 알려주고 수정 방법을 제안하거나, 코드의 가독성, 효율성, 안정성 측면에서 개선점을 제시해 줄 수 있어요.

4. 프롬프트 엔지니어링 베스트 프랙티스 (모범 사례)

효과적인 프롬프트를 만들기 위한 몇 가지 중요한 팁들이 있어요.

- **예시 제공하기:** 가장 중요한 팁 중 하나예요. Few-shot 예시를 제공하는 것은 AI에게 **원하는 출력 형태나 스타일을 가르치는 강력한 도구**가 됩니다. AI가 예시를 보고 따라 하도록 유도하여 결과물의 정확성과 스타일을 개선할 수 있어요. 분류 작업의 Few-shot 예시에서는 **클래스(분류 결과 종류)의 순서를 섞어주는 것**이 좋아요. 그래야 AI가 예시의 순서에 맞춰 학습하지 않고 각 클래스의 특징을 제대로 학습하게 되어 더 일반화된 성능을 얻을 수 있어요.
- **단순하게 디자인하기:** 프롬프트는 **간결하고, 명확하며, 이해하기 쉬워야** 해요. 사람에게도 혼란스러운 프롬프트는 AI에게도 혼란스러울 가능성이 높아요. 복잡하거나 불필요한 정보는

피하세요. '행동'을 나타내는 **구체적인 동사**를 사용하는 것이 도움이 돼요 (예: 분석하기, 분류하기, 생성하기, 요약하기, 번역하기 등).

- **출력을 구체적으로 명시하기:** 원하는 결과물의 **형식, 스타일, 내용 등을 자세하게** 알려주세요. 단순히 "블로그 글을 써줘" 라고 하는 것보다 "비디오 게임 콘솔 5개에 대한 3단락 분량의 블로그 글을 작성해줘. 정보 전달 위주에 재미있고 대화체 스타일로 써줘" 와 같이 구체적으로 요청하는 것이 훨씬 효과적이에요. 시스템 프롬프트나 컨텍스트 프롬프트를 활용하여 상세한 정보를 제공할 수 있어요.
- **제약(Constraints)보다 지시(Instructions) 사용하기:** AI에게 '**무엇을 해야 하는지(지시)**'를 명확하게 알려주는 것이 '**무엇을 하지 말아야 하는지(제약)**'를 나열하는 것보다 일반적으로 더 효과적이에요. 지시는 AI가 목표를 향해 나아가도록 안내하고 창의성을 발휘할 여지를 주지만, 제약은 AI를 혼란스럽게 하거나 잠재력을 제한할 수 있어요. 다만, 유해하거나 편향된 콘텐츠 생성을 막거나 엄격한 출력 형식이 필요할 때는 제약을 사용하는 것이 필요할 수 있어요. 가능하면 긍정적인 형태의 지시를 사용하세요.
- **최대 토큰 길이 제어하기:** 설정에서 최대 토큰 수를 지정하거나 프롬프트에 직접 길이를 명시하여 AI 응답의 길이를 조절할 수 있어요.
- **프롬프트에 변수 사용하기:** 프롬프트를 **재사용 가능하고 유연하게** 만들려면 변수를 사용하세요. 예를 들어, 특정 도시에 대한 사실을 알려주는 프롬프트에서 도시 이름을 직접 쓰기보다 변수로 처리하면, 변수 값만 바꿔서 다른 도시에 대한 정보를 쉽게 얻을 수 있어요. 이는 프롬프트를 애플리케이션에 통합할 때 특히 유용해요.
- **입력 형식과 글쓰기 스타일 실험하기:** 프롬프트의 **형식(질문, 설명, 지시 등), 단어 선택, 스타일(톤)** 등을 다양하게 시도해 보세요. 어떤 방식이 특정 AI 모델과 작업에 가장 잘 맞는 지 실험을 통해 찾아야 해요.
- **모델 업데이트에 적응하기:** LLM 모델은 계속해서 업데이트되고 성능이 개선돼요. 새로운 모델 버전을 사용해보고, 새로운 기능들을 활용하도록 프롬프트를 조정하는 것이 중요해요.
- **출력 형식 실험하기:** 텍스트 외에 **JSON, XML 등 구조화된 형식으로 출력을 요청**하는 것도 좋은 방법이에요. 특히 데이터 추출, 분류, 정렬 같은 비창의적인 작업에서 구조화된 형식은 결과물을 다루기 쉽게 만들고 AI의 **환각(Hallucination)** 현상을 줄이는 데 도움이 될 수 있어요.

- **다른 프롬프트 엔지니어와 함께 실험하기:** 혼자보다는 여러 사람이 각자 프롬프트를 만들어보고 결과를 비교하며 배우는 것이 효과적일 수 있어요.
- **CoT 특정 베스트 프랙티스:** CoT를 사용할 때는 추론 과정을 먼저 생성하게 한 다음 마지막에 최종 답변을 두어야 해요. 또한 추론 과정과 최종 답변을 분리하여 추출할 수 있도록 프롬프트를 설계해야 해요. 위에서 언급했듯이, CoT는 보통 **온도 설정을 0**으로 합니다.
- **다양한 프롬프트 시도 기록하기:** 프롬프트 엔지니어링은 반복적인 과정이기 때문에, **시도했던 다양한 프롬프트들을 자세히 기록하는 것이 매우 중요**해요. 어떤 시도가 성공했고 실패했는지, 사용한 모델, 설정 값, 결과 등을 기록하면 나중에 참고하거나 디버깅할 때 큰 도움이 돼요. 표 형식(소스의 Table 21 같은)으로 기록하는 것이 체계적이에요. 완성된 프롬프트는 코드와 분리하여 별도 파일로 관리하는 것이 좋아요.

결론

프롬프트 엔지니어링은 LLM을 우리가 원하는 방향으로 효과적으로 이끌어내는 기술이에요. 다양한 출력 설정과 기법들을 이해하고 활용하며, **반복적으로 실험하고 결과를 기록**하는 과정을 통해 점차 더 나은 프롬프트를 만들 수 있게 됩니다. 누구나 시도할 수 있으며, 창의적인 글쓰기부터 코딩, 데이터 분석 등 다양한 작업에 LLM을 더욱 유용하게 활용할 수 있도록 도와줘요.

