

Building LLM applications for production

by Chip Huyen

원본: <https://huyenchip.com/2023/04/11/llm-engineering.html>

번역: DeepL + mnncat@gmail.com

최근 제가 많이 받은 질문 중 하나는 대규모 언어 모델(LLM)이 머신 러닝 워크플로우를 어떻게 변화시킬 것인가 하는 것입니다. LLM Application을 사용하는 여러 회사와 함께 일해보고 개인적으로도 Application을 구축하면서 두 가지 사실을 깨달았습니다:

1. LLM으로 뭔가 멋져보이는 것을 만들기는 쉽지만, 프로덕션에 바로 사용할 수 있는 무언가를 만들기는 매우 어렵다는 것입니다.
2. LLM의 한계는 프롬프트 엔지니어링에 대한 공학적 엄격성 부족으로 더 나빠집니다. 이는 자연어 자체의 모호함이라는 특성 때문이기도 하고, 한편으로는 이 분야가 아직 발생 초기 단계이기 때문이기도 합니다.

이 글은 세 부분으로 구성되어 있습니다.

1부에서는 LLM Application을 제작할 때 마주치는 주요한 어려움들과 제가 경험한 해결책에 대해 설명합니다.

2부에서는 제어 흐름(예: if 문, for 루프)을 사용하여 여러 작업을 구성하고 보다 복잡하고 강력한 Application을 위한 도구(예: SQL 실행기, bash, 웹 브라우저, 타사 API)를 통합하는 방법에 대해 설명합니다.

3부에서는 기업들이 LLM을 기반으로 구축한 유망한 사용 사례들과 그것을 구축하는 방법을 다룹니다.

LLM에 대해서는 이미 많은 글이 작성되었으므로, 이미 잘 알고 있는 부분은 건너뛰셔도 좋습니다.

1부: 프롬프트 엔지니어링 제품화의 어려움

자연어의 모호성

프롬프트 평가

프롬프트 버전 관리

프롬프트 최적화

비용 및 지연 시간

비용

지연 시간

LLM 비용 + 지연 시간 분석의 불가능성

프롬프트 vs. 파인튜닝 vs. 대안

- [프롬프트 튜닝](#)
 - [distillation을 적용한 파인튜닝](#)
 - [임베딩 + 벡터 데이터베이스](#)
 - [이전 및 이후 버전과의 호환성](#)
- [2부. 작업 결합성](#)
 - [여러 작업으로 구성된 Application](#)
 - [에이전트, 도구 그리고 제어 흐름](#)
 - [도구 vs. 플러그인](#)
 - [제어 흐름: 순차, 병렬, if, for 루프](#)
 - [LLM 에이전트를 사용한 제어 흐름](#)
 - [에이전트 테스트하기](#)
- [3부. 유망한 사용 사례](#)
 - [AI 어시스턴트](#)
 - [챗봇](#)
 - [프로그래밍과 게임](#)
 - [학습에 활용하기](#)
 - [내 데이터와 대화하기](#)
 - [LLM이 데이터 분석을 대신 해줄 수 있나요?](#)
 - [검색과 추천](#)
 - [세일즈](#)
 - [SEO](#)
- [결론](#)

1부: 프롬프트 엔지니어링 제품화의 어려움

자연어의 모호성

컴퓨터 역사의 대부분을 엔지니어들은 프로그래밍 언어로 명령어를 작성해 왔습니다. 프로그래밍 언어는 '대부분' 정확합니다. 모호함은 개발자에게 좌절감을 안겨주고 심지어 열렬한 증오를 불러일으키기도 합니다 (파이썬이나 자바스크립트의 동적 타이핑을 생각해보세요).

프롬프트 엔지니어링에서는 프로그래밍 언어보다 훨씬 더 유연한 자연어로 명령어를 작성합니다. 이는 사용자 경험은 좋지만 개발자 경험은 상당히 나쁠 수 있습니다.

유연성은 사용자가 명령어를 정의하는 방식과 LLM이 이러한 명령어에 응답하는 방식이라는 두 가지 방향에서 비롯됩니다.

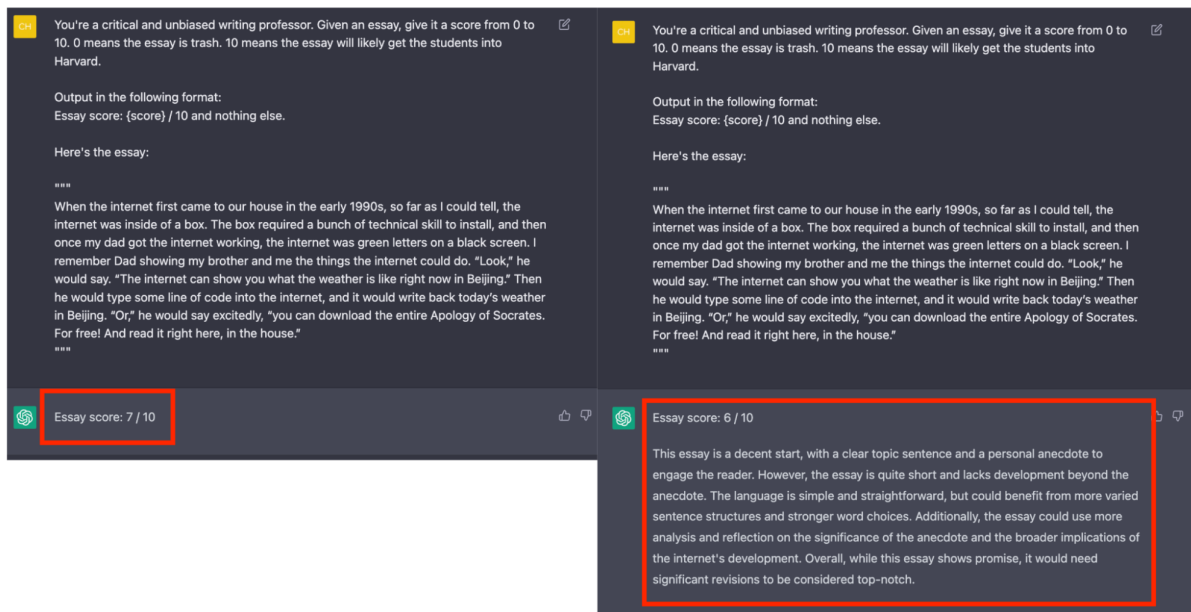
첫째, 사용자 정의 프롬프트의 유연성은 조용한 실패로 이어집니다. 누군가 실수로 임의의 문자를 추가하거나 줄을 제거하는 등 코드를 일부 변경하면 오류가 발생할 가능성이 높습니다. 하지만 사용자 정의 프롬프트는 실수로 프롬프트를 변경하더라도 계속 실행되지만 매우 다른 출력을 제공합니다.

사용자 정의 프롬프트의 유연성은 귀찮을 뿐이지만, LLM이 생성하는 응답의 모호성은 문제를 일으킬 수 있습니다. 이는 두 가지 문제로 이어집니다:

1. 모호한 출력 형식: LLM 위에 구축된 다운스트림 **Application**은 구문 분석이 가능하도록 특정 형식의 출력을 기대합니다. 출력 형식에 대해 명시적으로 프롬프트를 만들 수는 있지만 출력이 항상 이 형식을 따른다는 보장은 없습니다. 아래 예시를 보면 “**Essay score: {점수}**” 형태로 출력하도록 지시했지만 매 번 그 형식으로만 출력하지는 않는 걸 볼 수 있습니다.

Expected output format

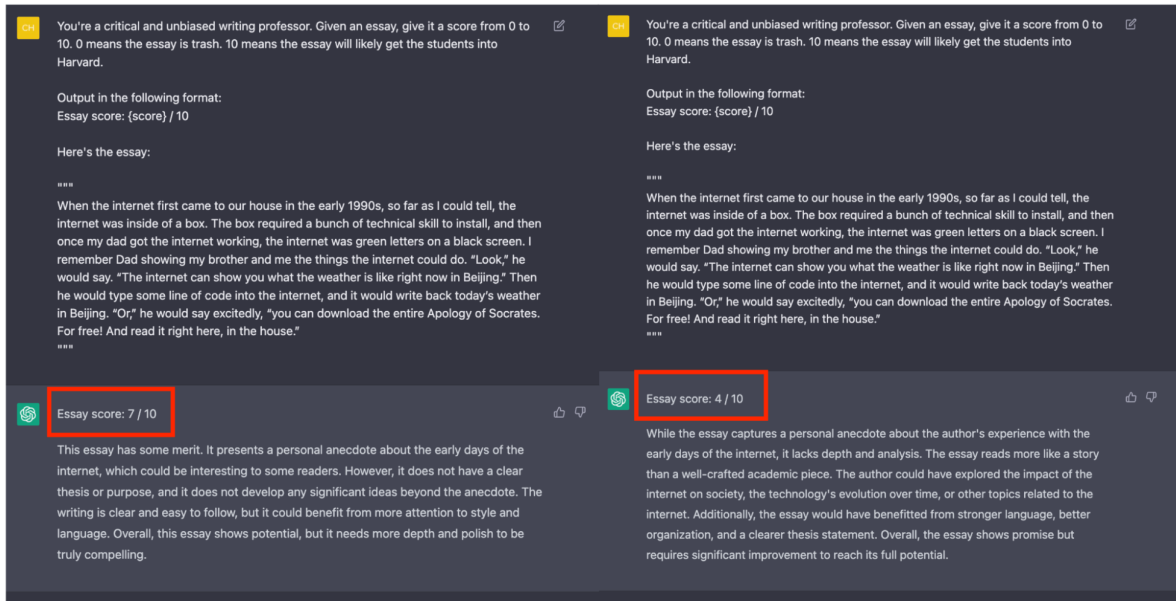
Essay score: {score} / 10 and nothing else.



2. 사용자 경험의 일관성 결여: **Application**을 사용할 때 사용자는 일정한 일관성을 기대합니다. 보험 회사에서 웹사이트를 확인할 때마다 다른 견적을 제공한다고 상상해 보세요. LLM은 확률적이기 때문에 매번 동일한 입력에 대해 동일한 결과를 제공한다는 보장은 없습니다.

Temperature = 0으로 설정하여 LLM이 동일한 응답을 제공하도록 강제할 수 있고 일반적으로 좋은 방법입니다. 이 방법은 일관성 문제를 대부분 해결하지만 시스템에 대한 신뢰를 줄 수는 없습니다. 특정 교실에 앉은 학생에게만 일관된 점수를 주는 교사가 있다고 상상해 보세요. 만약 그 교사가 다른 교실에 앉아 있다면 그 교사의 점수는 들쭉날쭉할 것입니다. 아래 예시에서는 동일한 프롬프트를 주어도 매 번 점수가 다른 것을 볼 수 있습니다.

Same prompt, different scores



이 모호성 문제를 어떻게 해결할 수 있을까요?

이 문제는 **OpenAI**가 적극적으로 해결하려고 노력하는 문제인 것 같습니다. 그들은 모델의 신뢰성을 높이는 방법에 대한 팁이 담긴 노트북을 가지고 있습니다.

수년간 **LLM**을 사용해 온 몇몇 사람들은 이러한 모호성을 받아들이고 이를 중심으로 워크플로를 구축했다고 말했습니다. 결정론적 프로그램을 개발하는 것과는 다른 사고방식이지만, 익숙해지는 것이 불가능한 것은 아닙니다.

이러한 모호성은 최대한 공학적 엄격성을 적용함으로써 완화할 수 있습니다. 이 글의 나머지 부분에서는 결정론적이지는 않더라도 신속한 엔지니어링을 체계적으로 만드는 방법에 대해 논의하겠습니다.

프롬프트를 평가하기

프롬프트 엔지니어링의 일반적인 기법은 프롬프트에 몇 가지 예제를 제공하고 이 예제를 통해 학습자가 일반화하기를 바라는 것입니다 (**Fewshot learning**).

예를 들어, 텍스트에 “논란 여지 점수”를 부여하는 것을 생각해 보겠습니다. (이것은 제가 실제로 트윗의 인기도와 논란의 여지 사이의 상관관계를 찾기 위해 수행한 재미있는 프로젝트였습니다.)

다음은 4개의 짧은 예시와 함께 짧게 작성된 프롬프트입니다:

예: “논란 여지 점수”

None

Given a text, give it a controversy score from 0 to 10.

Examples:

1 + 1 = 2

Controversy score: 0

Starting April 15th, only verified accounts on Twitter will be eligible to be in For You recommendations

Controversy score: 5

Everyone has the right to own and use guns

Controversy score: 9

Immigration should be completely banned to protect our country

Controversy score: 10

The response should follow the format:

Controversy score: { score }

Reason: { reason }

Here is the text.

Fewshot learning을 할 때는 두 가지 질문을 염두에 두어야 합니다:

1. **LLM**이 프롬프트에 제시된 예제를 이해하는지 여부. 이를 평가하는 한 가지 방법은 동일한 예제를 입력하고 모델이 예상 점수를 출력하는지 확인하는 것입니다. 프롬프트에 제공된 동일한 예제에서 모델이 제대로 작동하지 않는다면 프롬프트가 명확하지 않기 때문일 가능성이 높으므로 프롬프트를 다시 작성하거나 작업을 더 작은 작업으로 나누고(이 글의 파트 2에서 자세히 설명합니다.) 함께 결합하는 것이 좋을 수 있습니다.
2. **LLM**이 이러한 몇 가지 예제에 과도하게 적합한지 여부. 별도의 예제에서 모델을 평가할 수 있습니다.

제가 유용하다고 생각한 한 가지 방법은, 모델에 특정 레이블을 부여할 수 있는 예제를 만들도록 요청하는 것입니다. 예를 들어, 모델에 4점을 줄 수 있는 텍스트의 예시를 제공하도록 요청할 수 있습니다. 그런 다음 이 예시를 **LLM**에 입력하여 실제로 4점을 출력하는지 확인합니다.

None

```
from llm import OpenAILLM

def eval_prompt(examples_file, eval_file):
    prompt = get_prompt(examples_file)
    model = OpenAILLM(prompt=prompt, temperature=0)
    compute_rmse(model, examples_file)
    compute_rmse(model, eval_file)
    eval_prompt("fewshot_examples.txt", "eval_examples.txt")
```

프롬프트 버전 관리

프롬프트의 작은 변경은 매우 다른 결과를 초래할 수 있습니다. 따라서 각 프롬프트의 버전을 관리하고 성능을 추적하는 것이 필수적입니다. **git**을 사용하여 각 프롬프트와 그 성능을 버전 관리할 수도 있고, 프롬프트 실험을 위한 **MLflow**나 **Weights & Biases**와 같은 도구가 나온다고 해도 놀라운 일은 아닙니다.

프롬프트 최적화

프롬프트를 최적화하는 방법에 대한 많은 논문과 블로그 게시물이 작성되었습니다. 프롬프트 엔지니어링에 관한 대부분의 논문은 몇 문장으로 설명할 수 있는 트릭들이라는 [릴리안 웡의 유용한 블로그 게시물](#)에 동의합니다. **OpenAI**에는 [예제와 함께 많은 팁을 설명하는 훌륭한 노트](#)가 있습니다. 다음은 그 중 일부입니다:

- 모델이 어떻게 답변에 도달하는지 단계별로 설명하거나 설명하도록 지시하기 ([Chain-of-Thought](#) 또는 **COT**라고 알려진 기법)([Wei et al., 2022](#)). 단점: COT는 출력 토큰의 수가 증가하기 때문에 지연 시간과 비용이 모두 증가할 수 있습니다 [비용 및 지연 시간 섹션 참조].
- 동일한 입력에 대해 여러 출력을 생성하기. 다수결로 최종 출력을 선택하거나([self-consistency technique](#) 이라고도 알려진 [Wang et al., 2023](#)), LLM에 가장 좋은 출력을 선택하도록 요청할 수 있습니다. **OpenAI API**에서는 인자 `n`을 전달하여 동일한 입력에 대해 여러 개의 응답을 생성할 수 있습니다 (이상적인 API 설계는 아니라고 생각합니다).
- 하나의 큰 프롬프트를 더 작고 간단한 프롬프트로 나누세요.

많은 도구들이 프롬프트를 자동으로 최적화해 준다고 약속하지만, 비용이 꽤 비싸고 보통 이러한 트릭들을 적용하는 데 그칩니다. 이러한 도구의 한 가지 좋은 점은 코딩이 필요 없기 때문에 코딩을 모르는 분들에게도 매력적으로 다가갈 수 있다는 점입니다.

비용 및 지연 시간

비용

프롬프트에 더 명확한 세부 사항과 예제를 입력할수록 모델 성능이 더 좋아지며, 추론 비용이 더 많이 듭니다.

OpenAI API는 입력 토큰과 출력 토큰 모두에 대해 요금을 부과합니다. 작업에 따라 간단한 프롬프트에는 300~1000개의 토큰이 필요할 수 있습니다. 프롬프트에 자체 문서나 인터넷에서 검색한 정보를 추가하는 등 더 많은 컨텍스트를 포함하려는 경우 프롬프트에만 1만 토큰까지 쉽게 올라갈 수 있습니다.

긴 프롬프트의 비용은 실험이 아니라 추론에 있습니다.

실험 측면에서 프롬프트 엔지니어링은 무언가를 저렴하고 빠르게 실행할 수 있는 방법입니다. 예를 들어, 다음과 같은 설정으로 GPT-4를 사용하더라도 실험 비용은 300달러가 조금 넘습니다. 데이터 수집과 모델 학습에 드는 기존 ML 비용은 일반적으로 훨씬 더 비싸고 훨씬 더 오래 걸립니다.

- 프롬프트: 10,000토큰 (\$0.06/1,000토큰)
- 출력: 200토큰 (\$0.12/1천 토큰)
- 20개의 예시로 평가
- 25가지 버전의 프롬프트로 실험

LLMOps의 비용은 추론에 있습니다.

- 입력에 1만 토큰, 출력에 200 토큰이 있는 GPT-4를 사용하는 경우 예측당 \$0.624입니다.
- 입력과 출력 모두에 4k 토큰이 있는 GPT-3.5-터보를 사용하는 경우 예측당 \$0.004 또는 예측당 \$4/1k가 됩니다.
- 2021년에 DoorDash ML 모델은 하루에 100억 건의 예측을 수행했습니다. 각 예측에 \$0.004의 비용이 든다면 하루에 4천만 달러가 드는 셈입니다!
- 이에 비해 AWS Personalize 비용은 [예측 1천건당 약 \\$0.0417](#), AWS Fraud Detector 비용은 [예측 1천건당 약 \\$7.5](#) (한 달에 10만 건 이상 예측 시)입니다. AWS 서비스는 일반적으로 적당한 규모의 회사에서는 엄청나게 비싸고 유연성이 떨어지는 것으로 간주됩니다.

지연 시간

입력 토큰들은 병렬로 처리할 수 있으므로 입력 길이가 지연 시간에 큰 영향을 미치지 않습니다.

그러나 출력 토큰이 순차적으로 생성되기 때문에 출력 길이가 지연 시간에 큰 영향을 미칩니다.

매우 짧은 입력(51개 토큰)과 출력(1개 토큰)의 경우에도 gpt-3.5 터보의 지연 시간은 약 500ms입니다. 출력 토큰이 20개 이상으로 증가하면 지연 시간은 1초가 넘습니다.

다음은 제가 실행한 실험으로, 각 설정을 20회 실행했습니다. 모든 실험은 2분 이내에 이루어집니다. 실험을 다시 해보면 지연 시간은 매우 달라지겠지만 세 가지 설정 간의 관계는 비슷할 것입니다.

OpenAI와 같은 API를 사용하여 LLM Application을 제작할 때의 또 다른 어려움은 API의 신뢰성이 매우 낮고, SLA가 언제 제공될지에 대한 확약이 아직 없다는 점입니다.

# tokens	p50 latency (sec)	p75 latency	p90 latency
input: 51 tokens, output: 1 token	0.58	0.63	0.75
input: 232 tokens, output: 1 token	0.53	0.58	0.64
input: 228 tokens, output: 26 tokens	1.43	1.49	1.62

이러한 지연 시간이 모델 때문인지, 네트워킹 때문인지 (실행 간 편차가 크기 때문에 이 영향이 매우 클 것으로 예상됨) 또는 비효율적인 엔지니어링 오버헤드 때문인지 여부는 명확하지 않습니다. 가까운 시일 내에 지연 시간이 크게 줄어든 가능성은 매우 높아 보입니다.

0.5초의 지연시간은 많은 사용 사례에서 길어 보이지만, 모델의 크기와 API가 사용되는 규모를 고려하면 이 수치는 매우 인상적입니다. gpt-3.5-turbo의 파라미터 수는 공개되지 않았지만 약 1,500억 개로 추정됩니다. 이 글을 쓰는 현재로서는 이 정도 규모의 오픈 소스 모델은 없습니다. Google의 T5는 매개변수 수가 110억 개이고 Facebook의 가장 큰 LLaMA 모델은 매개변수 수가 65억 개입니다. 사람들은 [이 깃허브 스레드](#)에서 LLaMA 모델을 작동시키기 위해 어떤 구성이 필요한지 논의했는데, 30억 개의 파라미터 모델을 작동시키는 것은 충분히 어려운 일인 것 같았습니다. 가장 성공한 사람은 랜달러로, (토큰 하나를 생성하는데만 몇 초가 걸리는) [128GB RAM에서 동작하는 30B 파라미터 모델](#)을 만들었습니다.

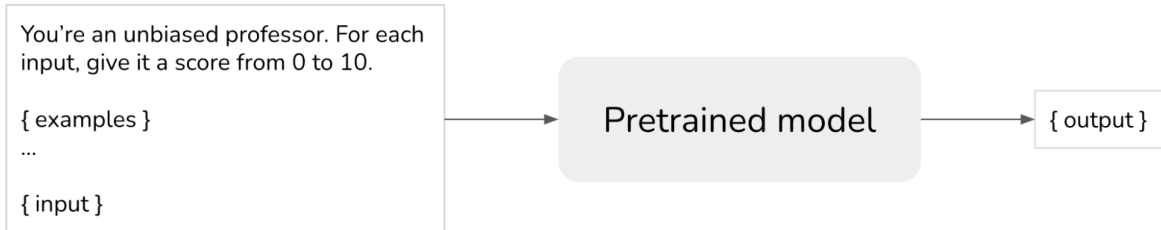
LLM 비용 + 지연 시간 분석의 불가능성

LLM Application 세계는 너무나 빠르게 변화하고 있기 때문에 비용 + 지연 시간 분석은 금방 구식이 될 수밖에 없습니다. Scribd의 응용 연구 선임 관리자인 Matt Ross는 불과 6개월 동안 자신의 사용 사례에 대한 API 비용이 두 자리수가 감소했다고 말했습니다. 지연 시간도 크게 줄었습니다. 마찬가지로 많은 팀에서 타당성을 추정하고 구매(유료 API 사용하기) 대 구축(오픈 소스 모델로 직접 구축)에 대한 결정을 매 주 다시 해야 할 필요를 느낀다고 말합니다.

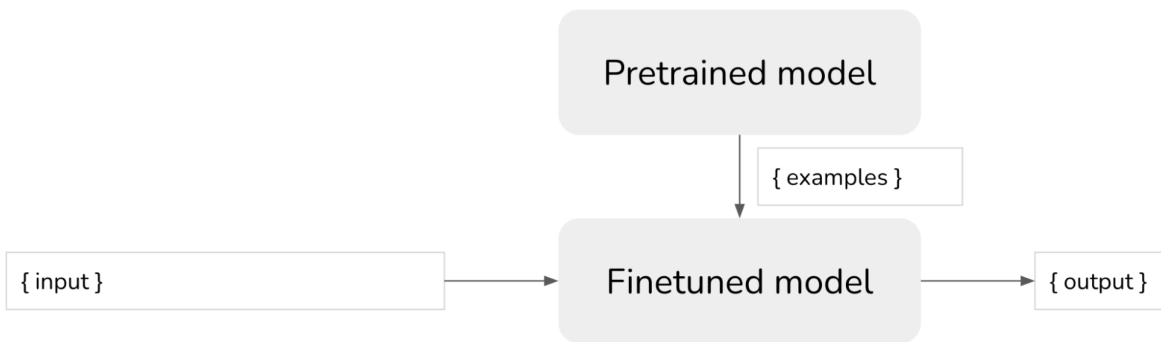
프롬프트 vs. 파인튜닝 vs. 대안

- 프롬프트: 각 샘플에 대해 모델이 어떻게 응답해야 하는지 명시적으로 알려줍니다.
- 파인튜닝: 모델을 학습해서 응답 방법을 가르치므로 프롬프트에 명시하지 않아도 됩니다.

Prompting



Finetuning



프롬프트와 파인튜닝을 고려할 때 데이터 가용성, 성능, 비용이라는 세 가지 주요 요소가 있습니다.

예시가 몇 개만 있는 경우 프롬프트를 사용하면 빠르고 쉽게 시작할 수 있습니다. 최대 입력 토큰 길이로 인해 프롬프트에 포함할 수 있는 예제 수에는 제한이 있습니다.

파인튜닝에 필요한 예시의 수는 물론 작업과 모델에 따라 다릅니다. 하지만 제 경험에 따르면 100여개의 예제를 파인튜닝하면 모델 성능에 눈에 띄는 변화를 기대할 수 있었습니다. 그러나 결과는 프롬프트보다 크게 나아지지 않을 수도 있습니다.

[프롬프트는 얼마나 많은 데이터 포인트의 가치가 있을까?](#) (2021)에서 Scao와 Rush는 프롬프트의 가치가 약 100개의 예제라는 사실을 발견했습니다(주의: 작업과 모델에 따라 편차가 크다는 점 - 아래 이미지 참조). 일반적인 추세는 예제 수를 늘릴수록 프롬프트보다 파인튜닝을 통해 더 나은 모델 성능을 얻을 수 있다는 것입니다. 모델을 파인튜닝하는 데 사용할 수 있는 예제 수에는 제한이 없습니다.

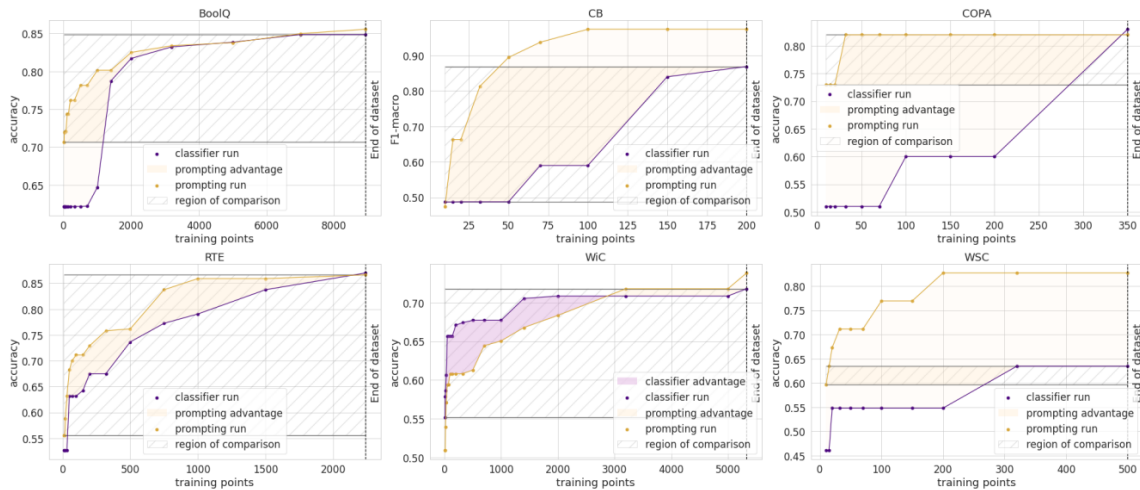


Figure 1: Prompting vs head (classifier) performance across data scales, up to the full dataset, for six SuperGLUE tasks. Compares the best prompt and head performance at each level of training data across 4 runs. Highlighted region shows the accuracy difference of the models. Cross-hatch region highlights the lowest- and highest- accuracy matched region in the curves. The highlighted area in this region is used to estimate the data advantage.

파인튜닝의 이점은 두 가지입니다:

1. 더 좋은 모델 성능을 얻을 수 있습니다. 더 많은 예제를 사용할 수 있고, 예제가 모델 내부 지식의 일부가 될 수 있습니다.
2. 예측 비용을 줄일 수 있습니다. 모델에 더 많은 지시사항을 구워 넣을수록 프롬프트에 입력해야 하는 지시사항이 줄어듭니다. 예를 들어, 각 예측에 대해 프롬프트에서 1,000개의 토큰을 줄일 수 있다면 gpt-3.5-turbo에서 1백만 개의 예측 당 2,000달러를 절약할 수 있습니다.

프롬프트 튜닝

프롬프트와 파인튜닝 사이에 있는 멋진 아이디어는 2021년에 Leister et al.이 소개한 [프롬프트 튜닝](#)입니다. 프롬프트를 변경하는 대신 프롬프트의 임베딩을 프로그래밍 방식으로 변경하는 것입니다. 프롬프트 튜닝이 작동하려면 프롬프트의 임베딩을 LLM 모델에 입력하고 이 임베딩에서 토큰을 생성할 수 있어야 하는데, 현재 이 작업은 오픈 소스 LLM에서만 가능하며 OpenAI API에서는 불가능합니다. T5에서는 프롬프트 튜닝이 프롬프트 엔지니어링보다 훨씬 더 나은 성능을 보이며 모델 튜닝을 따라잡을 수 있습니다 (아래 이미지 참조).

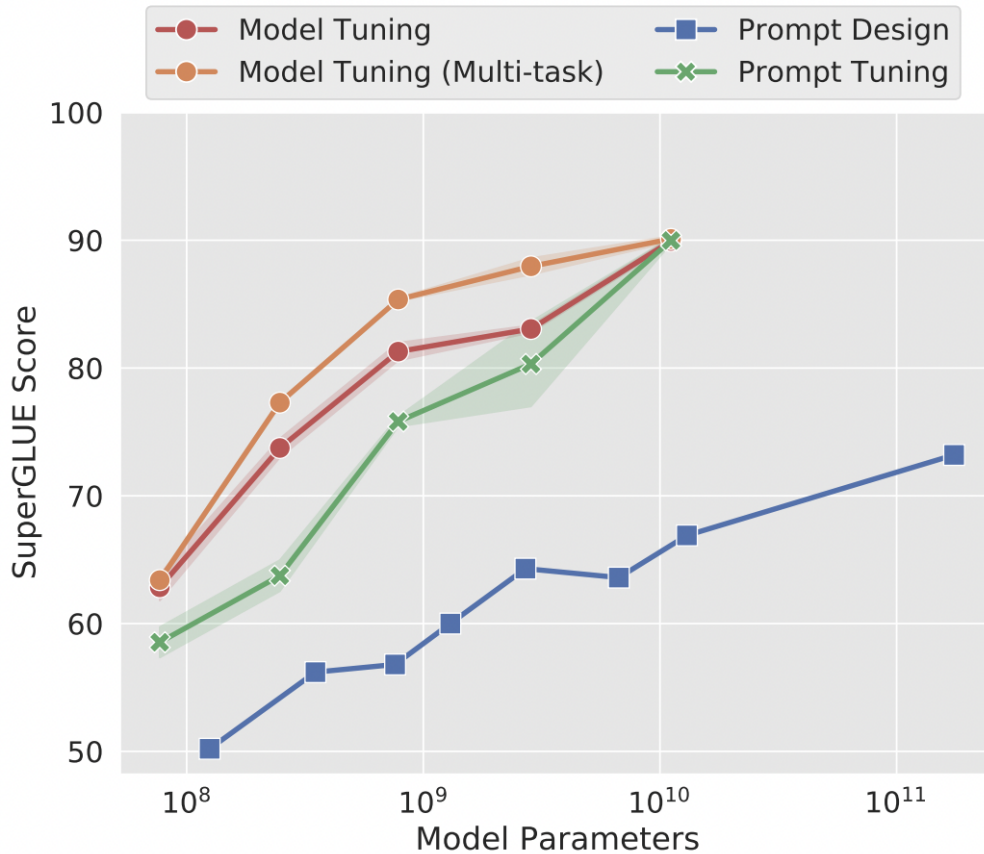
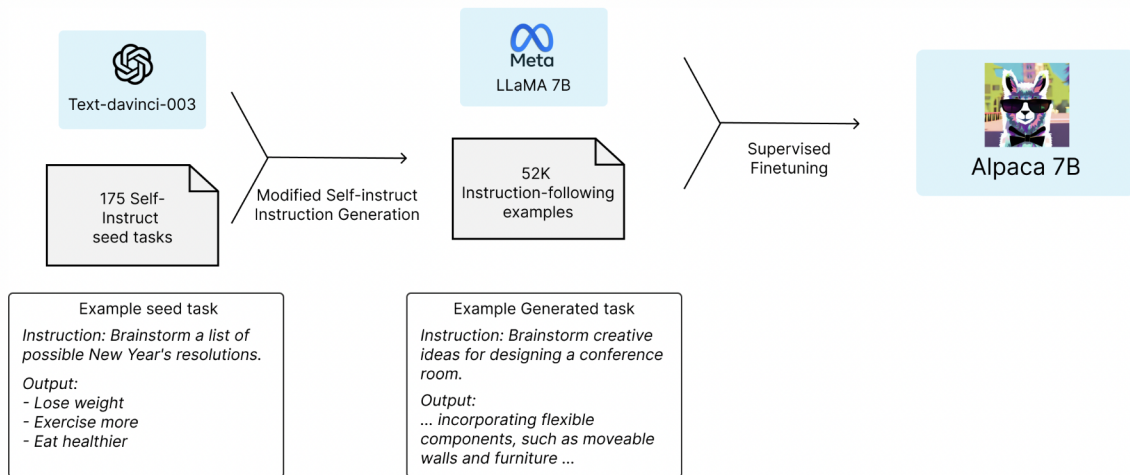


Figure 1: Standard **model tuning** of T5 achieves strong performance, but requires storing separate copies of the model for each end task. Our **prompt tuning** of T5 matches the quality of model tuning as size increases, while enabling the reuse of a single frozen model for all tasks. Our approach significantly outperforms few-shot **prompt design** using GPT-3. We show mean and standard deviation across 3 runs for tuning methods.

distillation을 적용한 파인튜닝

2023년 3월, 스탠포드 학생 그룹은 더 큰 언어 모델(text-davinci-003 - 1750억 개의 매개변수)로 생성된 예제에 대해 더 작은 오픈 소스 언어 모델(LLaMA-7B, 70억 개의 매개변수 버전)을 파인튜닝는 유망한 아이디어를 발표했습니다. 더 큰 모델의 동작을 모방하도록 작은 모델을 훈련하는 이 기법을 **distillation**이라고 합니다. 그 결과 파인튜닝된 모델은 text-davinci-003과 유사하게 작동하지만 훨씬 더 작고 실행 비용이 저렴합니다.

파인튜닝을 위해 52,000개의 명령어를 text-davinci-003에 입력하여 출력을 얻은 다음, 이를 사용하여 LLaMa-7B를 파인튜닝했습니다. 이를 생성하는 데 500달러 미만의 비용이 들었습니다. 학습에는 100달러 미만의 비용이 듭니다. [Stanford Alpaca: An Instruction-following LLaMA Model](#) (Taori et al., 2023)을 참조하세요.



이 접근 방식의 매력은 분명합니다. 3주 만에 GitHub 리포지토리의 별 수가 거의 2만 개에 달했습니다! 이에 비해 HuggingFace의 트랜스포머 리포지토리는 비슷한 수의 별을 달성하는 데 1년이 넘게 걸렸고, TensorFlow 리포지토리는 4개월이 걸렸습니다.



임베딩 + 벡터 데이터베이스

제가 매우 유망하다고 생각하는 한 가지 방향은 LLM을 사용하여 임베딩을 생성한 다음, 이 임베딩을 기반으로 검색 및 추천시스템 등의 ML Application을 구축하는 것입니다. 2023년

4월 현재, 더 작은 모델 **text-embedding-ada-002**를 사용하는 임베딩 비용은 토큰 **1,000개당 0.0004달러**입니다. 각 항목이 평균 **250토큰(187단어)**인 경우, 이 가격은 항목 1만 개당 **1달러** 또는 항목 **100만 개당 100달러**를 의미합니다.

기존 오픈소스 모델에 비해서는 여전히 비싸지만, 다음을 감안하면 여전히 매우 저렴한 가격입니다:

- 일반적으로 각 아이템에 대한 임베딩은 한 번만 생성하면 됩니다.
- OpenAI API를 사용하면 임베딩을 실시간으로 쉽게 생성할 수 있습니다.

GPT 임베딩 사용에 대해 자세히 알아보려면 [SGPT\(Niklas Muennighoff, 2022\)](#) 또는 [GPT-3 임베딩의 성능과 비용에 대한 이 분석\(Nils Reimers, 2022\)](#)을 확인하세요. Nils의 게시물에 있는 일부 수치는 이미 구식이지만 (이 분야는 매우 빠르게 발전하고 있습니다!!), 그 방법은 훌륭합니다!

임베딩 모델의 실시간 사용 사례에서, 가장 큰 비용은 이러한 임베딩을 벡터 데이터베이스에 로드하는 것입니다. 그러나 어떤 임베딩을 사용하든 이 비용은 발생하게 됩니다. **Pinecone**, **Qdrant**, **Weaviate**, **Chroma**와 같은 새로운 벡터 데이터베이스는 물론 기존의 **Faiss**, **Redis**, **Milvus**, **ScaNN** 등 수많은 벡터 데이터베이스가 꽃을 피우는 것을 보는 것은 매우 흥미롭습니다.

2021년이 그래프 데이터베이스의 해였다면, **2023**년은 벡터 데이터베이스의 해입니다.

이전 및 이후 버전과의 호환성

파운데이션 모델은 많은 태스크에 재학습 없이 바로 사용할 수 있습니다. 하지만 파운데이션 모델이 구식이 되면 때때로 재학습하거나 파인튜닝해야 합니다. 릴리안 왕의 [프롬프트 엔지니어링](#)에 따르면:

SituatedQA 데이터 세트에서 관찰한 한 가지 사실은, 서로 다른 날짜에 기반한 질문에 대해 *LM*(사전 학습 컷오프는 2020년)이 *Google* 검색을 통해 최신 정보에 액세스할 수 있음에도 불구하고 2020년 이후 질문에 대한 성능이 2020년 이전 질문에 비해 여전히 훨씬 떨어졌다는 것입니다. 이는 문맥 정보와 모델 내부 지식 사이에 약간의 불일치 또는 상충되는 매개변수가 존재한다는 것을 시사합니다.

기존 소프트웨어에서는 소프트웨어가 업데이트 되더라도 이전 버전용으로 작성된 코드도 계속 작동하는 것이 이상적입니다. 그러나 프롬프트 엔지니어링에서는 최신 모델을 사용하려는 경우 모든 프롬프트가 최신 모델에서도 의도한 대로 작동한다고 보장할 수 없으므로 프롬프트를 다시 작성해야 할 가능성이 높습니다. 사용하는 모델이 변경될 것으로 조금이라도 예상되는 경우, 평가 예제를 사용하여 모든 프롬프트를 단위 테스트하는 것이 중요합니다.

제가 자주 듣는 주장 중 하나는 프롬프트 재작성은 문제가 되지 않는다는 것입니다. 왜냐하면:

1. 최신 모델은 기존 모델보다 더 잘 작동할 것이기 때문에. 저는 이에 대해 확신이 없습니다. 최신 모델이 전반적으로 더 좋을 수도 있지만, 최신 모델이 더 나쁜 사용 사례도 있을 것입니다.
2. 비용 섹션에서 설명한 것처럼 프롬프트를 사용한 실험은 빠르고 저렴하기 때문에. 이 주장에 동의하지만, 오늘날 **MLOps**에서 제가 보는 가장 큰 문제는 모델 로직, 기능 로직, 프롬프트 등에 대한 중앙 집중식 지식이 부족하다는 것입니다. **Application**에는 복잡한 로직이 포함된 여러 프롬프트가 포함될 수 있습니다(2부. 작업 결합성에서 논의). 원래 프롬프트를 작성한 사람이 퇴사하면 원래 프롬프트의 의도를 파악하여 업데이트하기 어려울 수 있습니다. 이는 아무도 감히 건드릴 수 없는 700줄짜리 SQL 쿼리를 남기고 떠나는 상황과 비슷해질 수 있습니다.

또 다른 문제는 프롬프트 패턴이 변경에 강하지 않다는 것입니다. 예를 들어, 제가 본 많은 게시된 프롬프트는 "I want you to act as XYZ"로 시작합니다. OpenAI가 언젠가 "I'm an AI assistant and I can't act like XYZ."와 같은 문구를 출력하기로 결정하면 이러한 모든 프롬프트를 업데이트해야 합니다.

2부. 작업 결합성

여러 작업으로 구성된 Application

위의 “논쟁 여지 점수” 예시는 입력이 주어지면 논쟁 여지 점수를 출력하는 하나의 단일 작업으로 구성되어 있습니다. 하지만 대부분의 **Application**은 더 복잡합니다.

데이터베이스에 연결하여 자연어로 데이터베이스를 쿼리하려는 "내 데이터와 대화하기" 사용 사례를 생각해 보겠습니다. 신용카드 거래 테이블을 상상해 보세요. 다음과 같은 질문을 하고 싶다고 가정해 보겠습니다: "피닉스에는 몇 개의 고유한 판매자가 있으며 그 이름은 무엇인가요?"라고 질문하면 데이터베이스가 반환됩니다: "피닉스에는 9개의 고유한 판매자가 있으며 그 이름은 ..."입니다.

이를 수행하는 한 가지 방법은 다음과 같은 일련의 작업을 수행하는 프로그램을 작성하는 것입니다:

1. 작업 1: 사용자의 자연어 입력을 SQL 쿼리로 변환 [LLM]
2. 작업 2: SQL 데이터베이스에서 SQL 쿼리 실행 [SQL 실행기]
3. 작업 3: SQL 결과를 자연어 응답으로 변환하여 사용자에게 표시 [LLM]

에이전트, 도구 그리고 제어 흐름

제 지인들을 대상으로 소규모 설문조사를 실시한 결과 아직 용어에 대한 합의가 이루어지지 않은 것 같습니다.

에이전트라는 단어는 주어진 제어 흐름에 따라 여러 작업을 실행할 수 있는 **Application**을 지칭하기 위해 많이 사용되고 있습니다(제어 흐름 섹션 참조). 작업은 하나 이상의 도구를 활용할 수 있습니다. 위 예제에서 **SQL** 실행기는 도구의 한 예입니다.

참고: 제 지인 중 일부는 에이전트라는 용어가 이미 다른 맥락(예: 강화 학습에서 정책을 지칭하는 에이전트)에서 남용되고 있기 때문에 이 맥락에서 에이전트라는 용어를 사용하는 것을 거부합니다.

도구 vs. 플러그인

SQL 실행기 외에 다른 도구의 예는 다음과 같습니다:

- 검색 (예: **Google** 검색 **API** 또는 **Bing API** 사용)
- 웹 브라우저 (예: **URL**이 주어지면 해당 콘텐츠 가져오기)
- **bash** 실행기
- 계산기

도구와 플러그인은 기본적으로 같은 것입니다. 플러그인은 **OpenAI** 플러그인 스토어에 기여한 도구라고 생각하면 됩니다. 이 글을 쓰는 현재 **OpenAI** 플러그인은 아직 공개되지 않았지만, 누구나 도구를 만들고 사용할 수 있습니다.

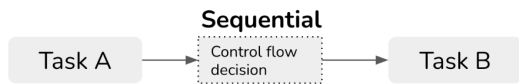
제어 흐름: 순차, 병렬, **if**, **for** 루프

위의 예에서 순차는 제어 흐름의 한 예로, 한 작업 씩차례로 실행되는 제어 흐름입니다. 병렬, **if** 문, **for** 루프와 같은 다른 유형의 제어 흐름도 있습니다.

- 순차: 작업 **A**가 완료된 후 작업 **B**를 실행하는 것으로, 작업 **B**가 작업 **A**에 종속되어 있기 때문입니다. 예를 들어, **SQL** 쿼리는 사용자 입력에서 변환된 후에만 실행할 수 있습니다.
- 병렬: 작업 **A**와 **B**를 동시에 실행합니다.
- **If** 문: 입력에 따라 작업 **A** 또는 작업 **B**를 실행합니다.
- **For** 루프: 특정 조건이 충족될 때까지 작업 **A**를 반복 실행합니다. 예를 들어 브라우저 작업을 사용하여 웹페이지의 콘텐츠를 가져온 다음 에이전트가 원래 질문에 대한 충분한 정보를 얻었다고 판단할 때까지 브라우저 작업을 계속 사용하여 해당 웹페이지에 있는 링크의 콘텐츠를 가져온다고 가정해 보겠습니다.

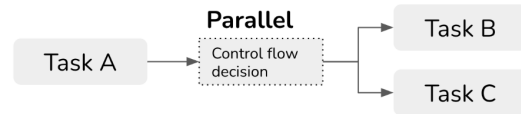
참고: 병렬 처리는 분명 유용할 수 있지만 이를 사용하는 **Application**을 많이 보지 못했습니다.

Control flows with LLMs



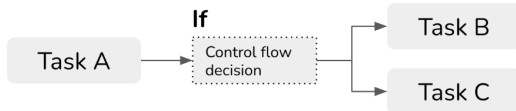
Example: talk-to-your-data

- Task A: translate user input into SQL query
- Task B: execute SQL query



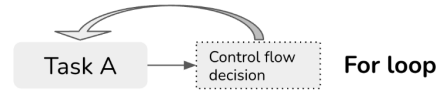
Example: project-based search

- Task A: translate user input into a list of items needed for this project
- Task B/C/...: for each item, retrieve all relevant products from your inventory



Example: AI assistant

- Task A: generate a response given a user input
- If response is a SQL query, execute task B
- If response is an idea, execute task C



Example: browsing

- Task A: keep browsing links found in a webpage

LLM 에이전트를 사용한 제어 흐름

기존 소프트웨어 엔지니어링에서는 제어 흐름의 조건이 정확합니다. **LLM Application**(에이전트라고도 함)에서는 프롬프트에 따라 조건이 결정될 수도 있습니다.

예를 들어 에이전트가 검색, **SQL** 실행기 및 채팅이라는 두 가지 작업 중 하나를 선택하도록 하려면 다음과 같이 (매우 근사적으로) 설명할 수 있습니다. 즉, **LLM**을 사용하여 제어 흐름의 조건을 결정할 수 있습니다!

None

You have access to three tools: Search, SQL executor, and Chat.

Search is useful when users want information about current events or products.

SQL executor is useful when users want information that can be queried from a database.

Chat is useful when users want general information.

Provide your response in the following format:

Input: { input }

Thought: { thought }

```
Action: { action }
Action Input: { action_input }
Observation: { action_output }
Thought: { thought }
```

에이전트 테스트하기

에이전트를 신뢰할 수 있으려면 각 작업을 결합하기 전에 개별적으로 빌드하고 테스트할 수 있어야 합니다. 실패 모드에는 크게 두 가지 유형이 있습니다:

1. 하나 이상의 작업이 실패. 잠재적 원인:
 - a. 제어 흐름이 잘못됨: 선택 사항이 아닌 작업이 선택됨
 - b. 하나 이상의 작업이 잘못된 결과를 생성함
2. 모든 작업이 올바른 결과를 생성하지만 전체 솔루션이 잘못됨. **Press et al. (2022)**은 이를 "결합성 격차"라고 하는데, 이는 모델이 하위 질문에 정답을 맞춘 모든 구성 문제 중에서 모델이 오답을 낸 구성 문제의 비율입니다.

소프트웨어 엔지니어링과 마찬가지로 각 구성 요소와 제어 흐름을 단위 테스트할 수 있고 또 해야 합니다. 각 구성 요소에 대해 프롬프트 또는 제어 흐름을 업데이트할 때마다 **Application**을 평가하는 데 사용할 수 있는 평가 예제로 (입력, 예상 출력) 쌍을 정의할 수 있습니다. 전체 **Application**에 대한 통합 테스트도 수행할 수 있습니다.

3부. 유망한 사용 사례

인터넷에는 LLM으로 구축된 멋진 **Application** 데모가 넘쳐나고 있습니다. 제가 본 가장 일반적이고 유망한 **Application** 몇 가지를 소개합니다. 제가 놓친 것들도 많을 겁니다.

더 많은 아이디어를 원하시면 제가 본 두 해커톤의 프로젝트를 확인해 보세요:

- [GPT-4 Hackathon Code Results](#) [Mar 25, 2023]
- [Langchain / Gen Mo Hackathon](#) [Feb 25, 2023]

AI 어시스턴트

가장 인기 있는 소비자 사용 사례입니다. 일정 관리, 메모 작성, 페어 프로그래밍, 이메일 응답, 부모님 도우미, 예약, 항공편 예약, 쇼핑 등 다양한 사용자 그룹을 위한 다양한 작업을 위해 만들어진 AI 어시스턴트가 있습니다. - 물론 궁극적인 목표는 모든 것을 도와줄 수 있는 비서가 되는 것입니다.

이는 모든 대기업이 수년 동안 노력해 온 성배이기도 합니다: 구글은 구글 어시스턴트와 바드, 페이스북은 M과 블렌더, 오픈AI(그리고 더 나아가서는 마이크로소프트)는 ChatGPT를 선보였습니다. AI에 의해 대체될 위험이 매우 높은 Quora는 여러 LLM과 채팅할 수 있는 자체

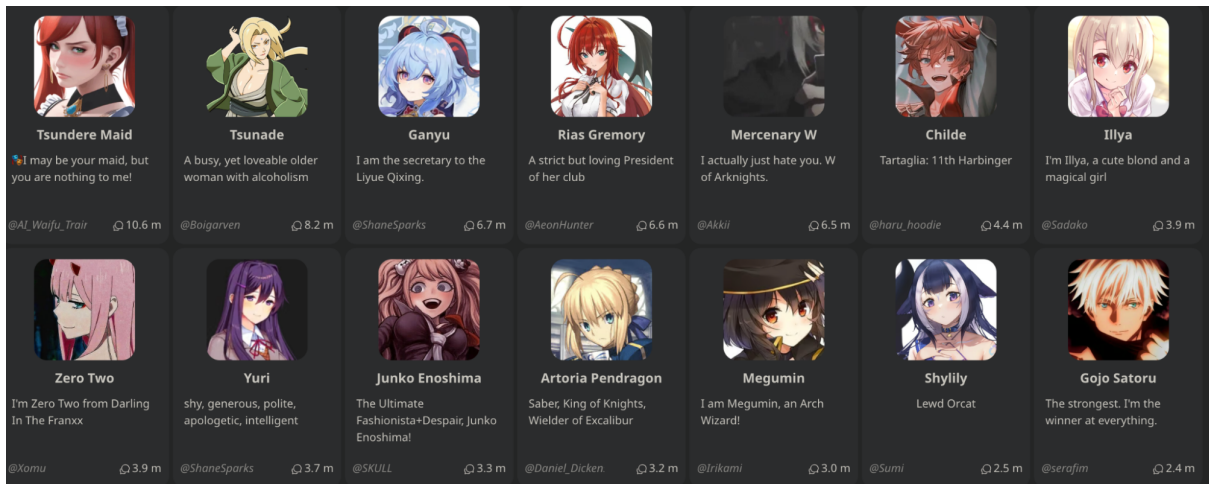
앱 Poe를 출시했습니다. 애플과 아마존이 아직 이 경쟁에 합류하지 않은 것은 놀라운 일입니다.

챗봇

챗봇은 API 측면에서 AI 비서와 유사합니다. AI 비서의 목표가 사용자가 지정한 작업을 수행하는 것이라면, 챗봇의 목표는 동반자 역할에 더 가깝습니다. 예를 들어 유명인, 게임/영화/책 캐릭터, 사업가, 작가 등과 같이 대화하는 챗봇을 만들 수 있습니다.

미셸 황은 [어린 시절 일기장을 GPT-3가 내면의 아이와 대화할 수 있도록 하는 프롬프트의 일부로 사용](#)했습니다.

소비형 챗봇 분야에서 가장 흥미로운 회사는 아마도 **Character.ai**일 것입니다. 사람들이 챗봇을 만들고 공유할 수 있는 플랫폼입니다. 이 플랫폼에서 가장 인기 있는 챗봇 유형은 애니메이션과 게임 캐릭터이지만 심리학자, 페어 프로그래밍 파트너 또는 언어 연습 파트너와 대화할 수도 있습니다. 대화하고, 행동하고, 그림을 그리고, 텍스트 기반 게임(예: AI 던전)을 플레이할 수 있으며, 캐릭터의 음성을 활성화할 수도 있습니다. 저도 몇 가지 인기 있는 챗봇을 사용해 보았습니다. 아직 대화를 지속할 수 있는 챗봇은 없었지만 이제 시작 단계에 불과합니다. 챗봇 제작자가 수익을 얻을 수 있도록 수익 공유 모델이 도입되면 더욱 흥미로운 일이 벌어질 수 있습니다.



프로그래밍과 게임

프로그래밍과 게임은 LLM이 코드 작성과 디버깅에 매우 능숙하기 때문에 또 다른 인기 있는 LLM Application 카테고리입니다. 이 분야의 선구자라고 할 수 있는 **GitHub Copilot**(이 글을 쓰는 시점 기준으로 5백만 다운로드를 기록한 **VSCode** 확장 프로그램)가 대표적입니다. LLM을 사용하여 코드를 작성하는 꽤 멋진 데모도 있습니다:

1. [자연어로 웹 앱 만들기](#)
2. 보안 위협 찾기: **Socket AI**는 [코드베이스의 npm 및 PyPI 패키지에 보안 위협이 있는지 검사](#)합니다. 잠재적인 문제가 발견되면 **ChatGPT**를 사용하여 결과를 요약합니다.

3. 게임

- a. 게임 만들기: 예를 들어 Wyatt Cheng은 [ChatGPT를 사용하여 Flappy Bird를 복제하는 방법](#)을 보여주는 멋진 영상을 공개했습니다.
- b. 게임 캐릭터 생성하기
- c. 게임 캐릭터와 더욱 사실적인 대화 나누기: [Convai의 멋진 데모](#)를 확인하세요!

학습에 활용하기

ChatGPT가 다운될 때마다 숙제를 완료할 수 없다는 불만을 토로하는 학생들로 OpenAI에 대한 불만이 쏟아졌습니다. 일부는 학교에서 ChatGPT 사용을 완전히 금지하는 것으로 대응했습니다. 일부는 훨씬 더 나은 아이디어, 즉 ChatGPT를 통합하여 학생들이 더 빠르게 학습할 수 있도록 돕는 방법을 제시했습니다. 제가 아는 모든 에듀테크 회사들은 ChatGPT를 본격적으로 연구하고 있습니다.

몇 가지 사용 사례를 소개합니다:

- 책 요약하기
- 학생들이 책이나 강의를 이해했는지 확인하기 위해 자동으로 퀴즈를 생성하기.
ChatGPT는 문제를 생성할 수 있을 뿐만 아니라 학생이 입력한 답이 맞는지 평가할 수도 있습니다.
 - 제가 사용해 본 결과 ChatGPT는 머신러닝 시스템 설계를 위한 퀴즈를 생성하는 데 꽤 능숙해 보였습니다. 곧 생성된 퀴즈를 게시할 예정입니다!
- 에세이 채점/피드백 제공하기
- 수학 솔루션 살펴보기
- 토론 파트너가 되기: ChatGPT는 같은 토론 주제에 대해 서로 다른 입장을 취하는 데 정말 능숙합니다.

홈스쿨링이 증가함에 따라 부모님들의 홈스쿨링을 돕기 위해 ChatGPT가 많이 활용될 것으로 예상됩니다.

내 데이터와 대화하기

제가 보기에 이것은 (지금까지) 가장 인기 있는 엔터프라이즈 Application입니다. 많은 스타트업이 기업 사용자가 자연어 또는 Q&A 방식으로 내부 데이터와 정책을 쿼리할 수 있는 도구를 만들고 있습니다. 일부는 법률 계약, 이력서, 재무 데이터 또는 고객 지원과 같은 업종에 초점을 맞추고 있습니다. 기업의 모든 문서, 정책, FAQ가 제공되면 고객 지원 요청에 응답할 수 있는 챗봇을 구축할 수 있습니다.

이 Application을 구축하는 주요 방법은 일반적으로 다음 4단계로 이루어집니다:

1. 내부 데이터를 데이터베이스(SQL 데이터베이스, 그래프 데이터베이스, 임베딩/벡터 데이터베이스 또는 단순한 텍스트 데이터베이스)로 구성합니다.

2. 자연어로 된 입력이 주어지면 이를 내부 데이터베이스의 쿼리 언어로 변환합니다. 예를 들어, **SQL** 또는 그래프 데이터베이스인 경우 이 프로세스는 **SQL** 쿼리를 반환할 수 있습니다. 임베딩 데이터베이스인 경우, **ANN**(근사 최인접 이웃) 검색 쿼리가 될 수 있습니다. 순전히 텍스트만 있는 경우, 이 프로세스는 검색 쿼리를 추출할 수 있습니다.
3. 데이터베이스에서 쿼리를 실행하여 쿼리 결과를 얻습니다.
4. 이 쿼리 결과를 자연어로 바꿉니다.

이렇게 하면 정말 멋진 데모를 만들 수 있지만, 이 범주가 얼마나 방어 가능한지는 잘 모르겠습니다. 스타트업이 **Google** 드라이브나 **Notion**과 같은 데이터베이스를 기반으로 사용자가 쿼리할 수 있는 **Application**을 만드는 것을 본 적이 있는데, 이는 **Google** 드라이브나 **Notion**이 일주일 안에 구현할 수 있는 기능인 것 같습니다.

OpenAI에는 [벡터 데이터베이스와 대화하는 방법에 대한 훌륭한 튜토리얼](#)이 있습니다.

LLM이 데이터 분석을 대신 해줄 수 있나요?

gpt-3.5-turbo에 몇가지 데이터를 입력해보니 일부 패턴을 감지할 수 있는 것 같습니다. 하지만 이는 입력 프롬프트에 들어갈 수 있는 작은 데이터에 대해서만 작동합니다. 대부분의 프로덕션 데이터는 그보다 더 큼니다.

CH

Given some data samples, tell me what insights and patterns you see. I care about signup start rate 

Here's the data

	device	screen_size (inches)	total	signup_start	signup_complete
0	mobile	<5	693	3	1
1	mobile	5-6	1498	47	23
2	mobile	6-7	703	23	11
3	web	<13	1395	8	2
4	web	13-14	2821	136	76
5	web	14-15	2297	121	67
6	web	>15	1492	69	38



Looking at the data, it appears to be related to user signups on a platform, with columns indicating device, screen size, total number of users, signup start, and signup completion times.  

Focusing on the column "signup_start", it seems that the signup start rate (the proportion of users who started the signup process) is generally low across all device types and screen sizes.

- For mobile devices with screens smaller than 5 inches, only 0.43% of total users started the signup process.
- For mobile devices with screens between 5-6 inches and 6-7 inches, the signup start rate is slightly higher at 3.14% and 3.27%, respectively.
- For web users, the signup start rate is highest for users with screen sizes between 13-14 inches, with 4.82% of total users starting the signup process.
- Users with screen sizes greater than 14 inches have lower signup start rates, with a rate of 3.36% for screens between 14-15 inches and 4.62% for screens greater than 15 inches.

Overall, the signup start rate appears to be quite low, indicating that there may be some friction in the signup process that is preventing users from completing the process.

However, it's also possible that some users may be browsing or exploring the platform without necessarily intending to sign up, which could be contributing to the low signup start rates.

검색과 추천

검색과 추천은 항상 엔터프라이즈 사용 사례의 핵심이었습니다. LLM과 함께 르네상스를 맞이하고 있습니다. 검색은 대부분 키워드 기반이었습니다. 텐트가 필요하다면 텐트를 검색하는 식이었죠. 하지만 아직 필요한 것이 무엇인지 모른다면 어떨까요? 예를 들어 11월에 오레곤의 숲으로 캠핑을 가려고 한다면 다음과 같은 일을 하게 될 겁니다:

1. 다른 사람들의 경험을 읽어보기 위해 검색하기
2. 검색된 블로그 게시물을 읽고 필요한 항목의 목록을 수동으로 추출하기
3. Google이나 다른 웹사이트에서 이러한 각 항목을 다시 검색하기

Amazon이나 전자상거래 웹사이트에서 직접 '11월에 오레곤에서 캠핑할 때 필요한 것들'을 검색하면 다음과 같은 결과를 얻을 수 있습니다:

The screenshot shows the Amazon search results page for the query "things you need for camping in oregon in november". The search bar at the top displays the query. Below the search bar, there are several product recommendations. The first row features three multi-tools: a Roxon CM1349 SPARK Multitool Plier, a ROXON S801 16-in-1 STROM Multitool, and a Roxon 5501 Folding Pocket Knife. The second row shows four items: a Bestway SaluSpa Miami Inflatable Hot Tub, a Paula's Choice exfoliant, a Coleman SaluSpa Inflatable Hot Tub, and a book titled "I'M DEAD. NOW WHAT?". Each product listing includes an image, a title, a rating, and a price.

만약에 아마존에서 "11월 오레곤에서 캠핑할 때 필요한 것들"을 검색하면 실제로 캠핑 여행에 필요한 물품 목록이 표시된다면 어떨까요?

이제는 LLM을 통해 가능해졌습니다. 예를 들어, Application을 다음과 같은 단계로 나눌 수 있습니다:

1. 작업 1: 사용자 쿼리를 제품 이름 목록으로 변환 [LLM]
2. 작업 2: 목록의 각 제품 이름에 대해 제품 카탈로그에서 관련 제품을 검색

CH

things i need for camping in the woods in oregon in november.

output should be in the format:

* {{ item type }}: {{ why }}.

Recommended products for this item type:

{{ product name }}: {{ cost range }}: {{ where to find it }}



- Tent: To provide shelter and protection from the elements.

Recommended products for this item type:

REI Co-op Half Dome 2 Plus: \$200-\$250: REI.com or local REI store.

- Sleeping Bag: To keep you warm during the cold nights.

Recommended products for this item type:

The North Face Eco Trail Down 20: \$180-\$220: TheNorthFace.com or local outdoor store.

- Sleeping Pad: To provide insulation and cushioning for a comfortable night's sleep.

Recommended products for this item type:

Therm-a-Rest NeoAir XLite: \$130-\$170: Thermarest.com or local outdoor store.

- Backpack: To carry all of your camping gear and essentials.

Recommended products for this item type:

Osprey Atmos AG 65: \$270-\$310: Osprey.com or local outdoor store.

- Camping Stove: To cook meals and boil water.

Recommended products for this item type:

MSR PocketRocket 2: \$40-\$50: MSRgear.com or local outdoor store.

이것이 성공한다면 LLM이 제품을 추천하도록 하는 기술인 LLM SEO가 등장할지 궁금합니다.

세일즈

세일즈에 LLM을 사용하는 가장 확실한 방법은 세일즈 이메일을 작성하는 것입니다. 하지만 누구도 더 많은 또는 더 나은 세일즈 이메일을 원하지는 않습니다. 하지만 제 네트워크에 속한 여러 회사에서 LLM을 사용하여 회사에 대한 정보를 종합하여 필요한 것이 무엇인지 파악하고 있습니다.

SEO

SEO는 곧 매우 이상해질 것입니다. 오늘날 많은 기업들이 Google에서 높은 순위를 차지하기 위해 많은 콘텐츠를 제작하는 데 의존하고 있습니다. 하지만, LLM이 콘텐츠 제작에 정말

능숙하고, 특정 키워드에 대해 **SEO**에 최적화된 콘텐츠를 무제한으로 제작하는 서비스를 제공하는 스타트업이 몇 개 있다는 점을 감안하면 검색 엔진에 콘텐츠가 넘쳐날 것입니다. 검색 엔진은 **AI**가 생성한 콘텐츠를 감지하는 새로운 알고리즘을 개발하고, 기업은 이러한 알고리즘을 우회하는 데 더 능숙해지기 때문에 **SEO**는 고양이와 쥐의 게임이 될 수도 있습니다. 또한 사람들은 검색보다는 브랜드에 더 의존하게 될 수도 있습니다(예: 특정 사람이나 회사가 만든 콘텐츠만 신뢰하는 경우).

결론

우리는 아직 **LLM Application**의 초기 단계에 있으며 모든 것이 매우 빠르게 발전하고 있습니다. 최근에 **LLM**에 관한 책 제안서를 읽었는데, 가장 먼저 든 생각은 '이 내용 대부분이 한 달 안에 구식이 될 것'이라는 것이었습니다. **API**는 하루가 다르게 변화하고 있습니다. 새로운 **Application**이 발견되고 있습니다. 인프라는 공격적으로 최적화되고 있습니다. 비용 및 지연 시간 분석은 매주 수행해야 합니다. 새로운 용어가 도입되고 있습니다.

이러한 변화가 모두 중요한 것은 아닙니다. 예를 들어, 많은 프롬프트 엔지니어링 논문을 보면 가중치를 초기화하는 다양한 방법을 설명하는 수천 개의 논문이 있던 딥 러닝 초창기 시절이 떠오릅니다. 프롬프트를 조정하는 트릭은 다음과 같습니다: "정직하게 대답하세요", "나는 당신이 ...처럼 행동하기를 바랍니다.", "질문:" 대신 "Q:"를 쓰는 것과 같은 트릭은 장기적으로는 중요하지 않을 것입니다.

LLM이 스스로 프롬프트를 작성하는 데 꽤 능숙해 보인다는 점을 감안할 때([Large Language Models Are Human-Level Prompt Engineers](#)(Zhou et al., 2022)) 프롬프트를 조정하는 데 사람이 필요할지 누가 알겠습니까?

그러나 너무 많은 일이 일어나고 있기 때문에 어떤 것이 중요하고 어떤 것이 중요하지 않은지 알기가 어렵습니다.

저는 최근 **LinkedIn**에서 사람들이 이 분야에 대한 최신 정보를 얻는 방법에 대해 질문했습니다. 방법들은 **hype**을 무시하는 것부터 모든 도구를 다 사용해 보는 것까지 다양했습니다.

1. hype을 (대부분) 무시하기

비키 보이키스(**Duo Security**의 선임 **ML** 엔지니어): 저는 엔지니어링이나 데이터 프레임워크들에 대해서와 마찬가지로, 매일 뉴스를 대강 훑어보고 대부분을 무시한 채 6개월을 기다렸다가 무엇이 중요한지 확인합니다. 중요한 것은 여전히 존재할 것이고, 무슨 일이 일어나고 있는지 맥락을 파악하는 데 도움이 되는 더 많은 설문조사 논문과 검증된 구현이 있을 것입니다.

2. 요약만 읽기

Shashank Chaurasia(**Microsoft** 엔지니어링): 저는 **BingChat**의 크리에이티브 모드를 사용하여 **Gen AI**와 관련된 새로운 기사, 블로그 및 연구 논문을 빠르게 요약해 볼 수 있습니다. 세부 사항을 이해하기 위해 연구 논문과 깃허브 리포지토리에 자주 채팅을 합니다.

3. 최신 도구들의 최신 정보를 따라가기

크리스 알렉시옥(창립 ML 엔지니어, Ox): 저는 새로운 도구가 나올 때마다 그 도구로 만들어보려고 노력합니다. 그래야 다음 단계가 나왔을 때 차이만 살펴볼 수 있기 때문입니다.

여러분의 전략은 무엇인가요?