

인수인계서

AI연구파트

배성수

9103496

1. large context LLM으로의 Finetune

1. 배경

LLM 모델이 Large context length를 가지고 있다면, 추가 학습을 할 필요 없이 필요한 정보를 통째로 입력해 줌으로써 필요한 결과 혹은 개인화된 답변을 얻는 것이 가능하다. 여기서 더 나아가 prompt에 쓰인 데이터가 학습에 사용된다면 intensive한 prompt engineering 없이도 답변이 가능할 수 있다. (하지만 pretraining부터 진행한다면 많은 시간이 오래 걸릴 것으로 예상된다.)

커스텀 LLM 개발을 위해서 다음 세 개의 단계를 거친다:- 1) large context로의 Finetuning이 가능한 **모델 구조** 2) 한국어 데이터셋으로 **사전학습이 된 모델** 3) 풀고자 하는 Task에 맞는 **데이터**와 이를 사용한 **Finetuning 기법**이 필요하다.

2. 모델 구조

1) 문제

대부분의 LLM API와 오픈소스 LLM은 각자 학습된 context length로 제한되어 있다. 최대 2K의 context length로 학습된 모델을 사용하면 context length가 2K로 제한되고, 해당 모델은 100K context length로 추론했을 때 의미없는 결과를 얻게 된다. 이러한 이유는 학습 데이터의 분포와 추론할 때의 데이터 분포의 거리가 멀기 때문이다.

단순하게 100K length를 가진 학습 데이터로 바닐라 트랜스포머를 학습시키기 위해서는 2K context length 학습에 필요한 비용의 50배가 필요로 하게 된다. 이러한 문제를 해결하기 위해 1) *요약 및 chained prompts*, 2) 원하는 문서의 임베딩을 저장하는 벡터 데이터베이스를 관리하고 유사도 등을 사용해 검색 그리고 3) 사전학습된 LLM 모델을 커스텀 데이터를 사용해 Fine-tuning하는 방법 등이 존재한다. 이 중 3) **Fine-tuning하는 방향**을 알아보자.

2) 해결 방법

Fine-tuning이 가능한 모델을 위해서는 두가지 문제를 해결해야 한다.

a. **Length Extrapolation:** 학습 데이터에서 보지 못한 Context length에 따른 일반화(필수)

b. Computing Complexity(Time & Memory): 입력 길이 n 에 **제곱**으로 증가하는 Computing and Memory Complexity

a. Length Extrapolation 문제: 기존 트랜스포머의 sinusoidal positional encoding은 Extrapolation 능력이 매우 취약하다. 학습된 context length보다 긴 입력이 주어지면 성능이 빠르게 떨어지는 모습을 보여준다. 이를 보완하기 위해 몇 가지 positional encoding 방법이 고안되었다. 또한 몇 논문에 따르면 length extrapolation이 실패하는 큰 이유는 "방해 토큰"이 추론을 방해하기 때문이라고 말한다.

1) ALiBi(Attention with Linear Bias): sin positional encodings을 토큰 pair의 거리에 비례해 attention score에 패널티를 부여하는, 학습이 필요 없는 bias를 추가해준다. 해당 방법은 토큰끼리의 거리가 멀어질 수록 서로 간의 관련성 혹은 중요성이 떨어질 거라는 직관에 기반한 방법이다.

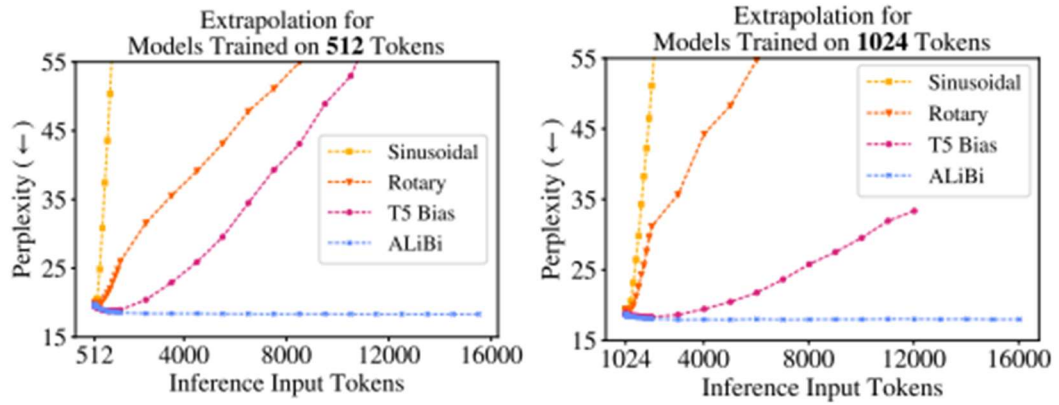


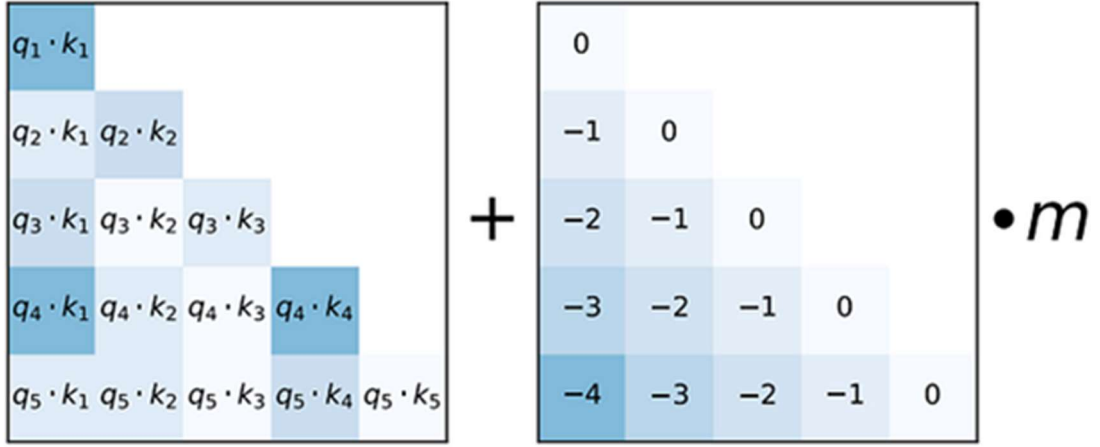
Figure 1: Extrapolation: as the (validation-set's) input sequence gets longer (x -axis), current position methods (sinusoidal, rotary, and T5) show degraded perplexity (y -axis, lower is better), but our method (§3) does not. Models were trained on WikiText-103 with sequences of $L = 512$ (left) or $L = 1,024$ (right) tokens. T5 ran out of memory on our 32GB GPU. For more detail on exact perplexities and runtimes, see Tables 2 and 3 in the appendix.

Sinusoidal과 다르게 네트워크의 시작점에서 적용되지 않고 attention head에 적용되고, QK attention score에 각각의 Query와 Key쌍에 거리에 비례하는 편향을 추가한다.

$$\text{softmax}(\mathbf{q}_i \mathbf{K}^\top + m \cdot [-(i-1), \dots, -2, -1, 0]),$$

where scalar m is a head-specific slope fixed before training. Figure 3 offers a visualization.

[Figure 2]



[Figure 3]

모델이 n 개의 head를 가지고 있으면 $2-8/n$ 으로 시작한다. 예를 들어 8개의 head가 있는 경우 $1/2$ 로 시작해 $1/22, 1/23, \dots, 1/28$ 이런 식으로 적용된다.
이렇게 구한 slope는 여러 도메인, 여러 모델사이즈에서 다양하게 적용할 수 있으며, 해당 논문의 저자들은 새로운 모델에 따라 slope를 구할 때 튜닝이 필요하지 않다고 믿는다.

2) RoPE(Rotary Embedding)[\[참고\]](#): ALiBi와 마찬가지로 sinusoidal 을 대체할 수 있는 positional embedding 방법이다. RoPE는 rotation matrix를 사용해 절대적 위치 정보를 인코딩하는 Position Embedding이다. 입력의 길이에 상관 없이 유연하게 적용 가능하고, 토큰 간의 상대적 거리가 커질 수록 서로의 의존성을 붕괴시켜준다.

where

$$f_{\{q,k\}}(\mathbf{x}_m, m) = \mathbf{R}_{\Theta, m}^d \mathbf{W}_{\{q,k\}} \mathbf{x}_m$$

$$\mathbf{R}_{\Theta, m}^d = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, \dots, d/2]\}$$

RoPE를 적용한 self-attention 수식은 다음과 같다.

$$\mathbf{q}_m^\top \mathbf{k}_n = (\mathbf{R}_{\Theta, m}^d \mathbf{W}_q \mathbf{x}_m)^\top (\mathbf{R}_{\Theta, n}^d \mathbf{W}_k \mathbf{x}_n) = \mathbf{x}^\top \mathbf{W}_q \mathbf{R}_{\Theta, n-m}^d \mathbf{W}_k \mathbf{x}_n$$

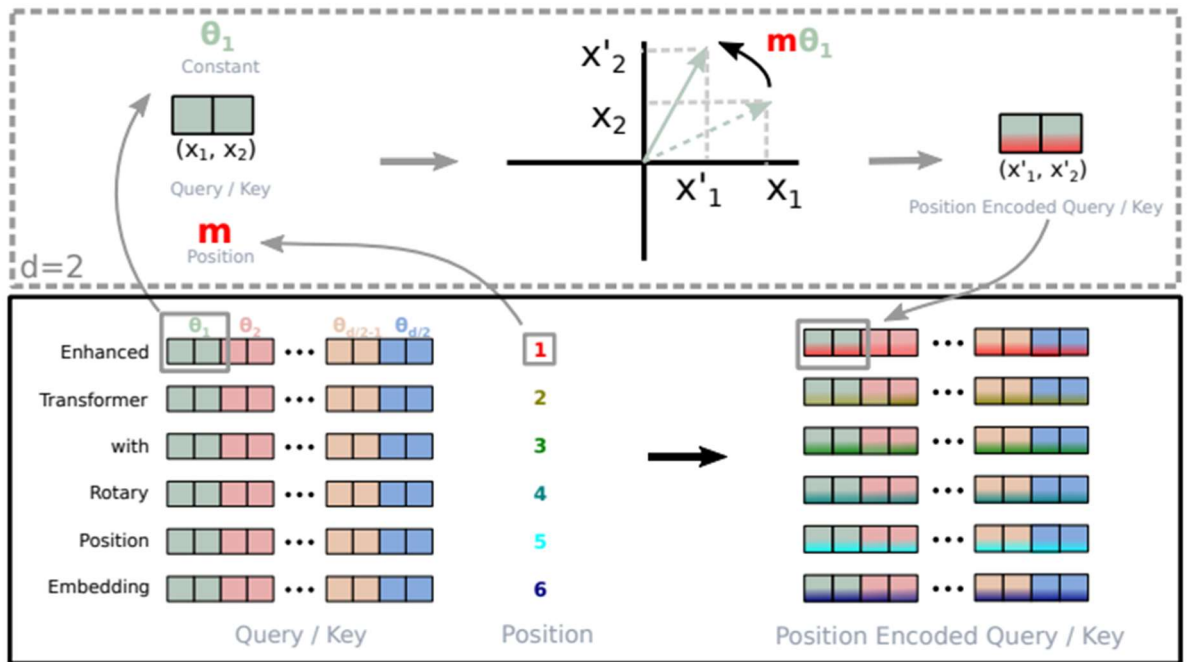


Figure 1: Implementation of Rotary Position Embedding(RoPE).

RoPE와 Sinusoidal embedding과의 차이점은 두가지가 있다.

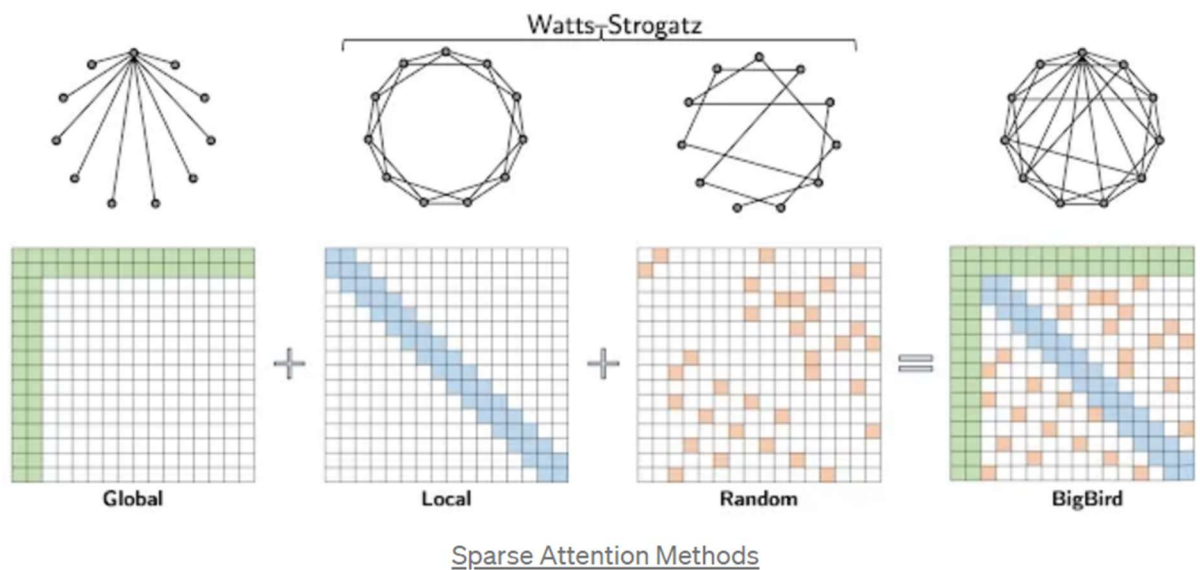
1. Sinusoidal 임베딩은 각 좌표별로 따로 적용되지만, Rotary 임베딩은 좌표 쌍을 혼합하여 적용한다.
2. Sinusoidal 임베딩은 코사인과 사인을 더하여 사용하지만 Rotary 임베딩은 이를 곱해 사용한다.
단점은 RoPE를 계산하고 적용하는 시간이 Sinusoidal보다 2~4배정도 느리다는 점이다.

3) “방해토큰” 완화를 위한 방법[\[참고\]](#): 입력 길이가 길어지면 추론을 저해하는 방해토큰이 존재하는데, 이 방해토큰을 모두 없앨 수 있다면 length generalization이 가능할 것이다. LLaMa에 Landmark 토큰 도입과 windowed-attention을 사용해 LLaMA 7B 모델을 32K context length에 fine-tuning 했다.

b. Computing Complexity(Time & Memory): 컴퓨팅 복잡도가 입력 길이의 제곱으로 커지므로 이 또한 문제가 된다. 따라서 Attention을 계산하는 다양한 방법을 고안하여 제곱을 선형적으로 만들거나, Attention 계산을 GPU에서 최적화를 하는 방법이 존재한다.

1) **Sparse Attention:** 100K의 토큰이 입력으로 주어질때 모든 토큰이 서로 관련이 있다고 볼 수 없다. 이러한 계산 복잡도를 줄이기 위해서 어텐션 스코어를 계산할 때 일부 토큰만 고려하는 방법이 있다. 해당 방법은 학습과 추론 둘 다의 속도를 개선해준다.

Sparse Attention 방법은 희소성을 추가해 줌으로써 Attention의 계산 복잡도를 n 의 제곱이 아닌 n 에 선형적으로 비례할 수 있게.

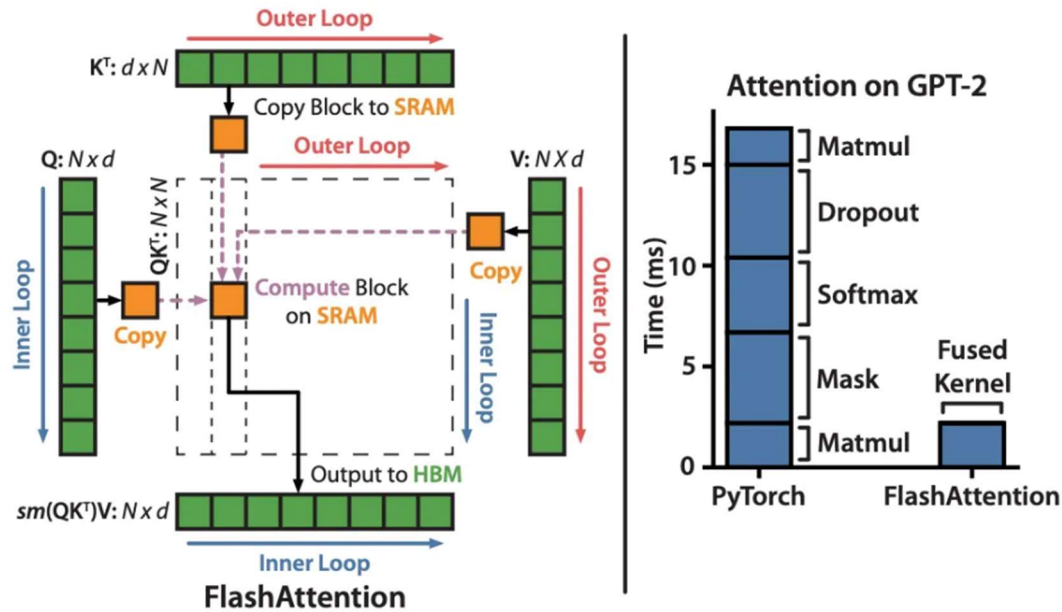


Sliding Window Attention(also called as Local)의 경우 윈도우의 크기 w (각 토큰마다 양옆의 $w/2$ 개의 토큰들을 어텐션 스코어 계산에 사용한다. 그러면 계산 복잡도는 $O(n \cdot w)$ 가 되고 입력 크기 n 에 대해 선형 복잡도를 가지게 된다. (효율을 위해 w 는 n 보다 작아야 함)

Big Bird Attention은 세가지 sparse attention method를 모두 합친 방법이다.

2) FlashAttention: attention layer에서 계산하기 위한 중간 결과의 크기는 $(n \times n)$ 으로 매우 크다. 이를 계산하려면 중간 결과 저장과 계산을 하기 위해 **HBM**(large but slower)과 **SRAM**(small but faster)를 왕복 해야 한다. 이 둘의 속도와 크기의 차이가 존재하므로 **bottleneck**이 발생한다.

FlashAttention 알고리즘은 Q, K, V 행렬들을 블록으로나눠서 SRAM으로 로딩한 뒤, 해당 블록에 대한 어텐션 결과를 계산해 최종 결과 값을 SRAM에 저장한다. 이러한 프로시저를 **tiling** 이라고 부른다. FlashAttention은 학습과 추론의 속도를 향상시켜준다. (Pytorch 2.0에는 flash-attention이 내장되어 있다.)

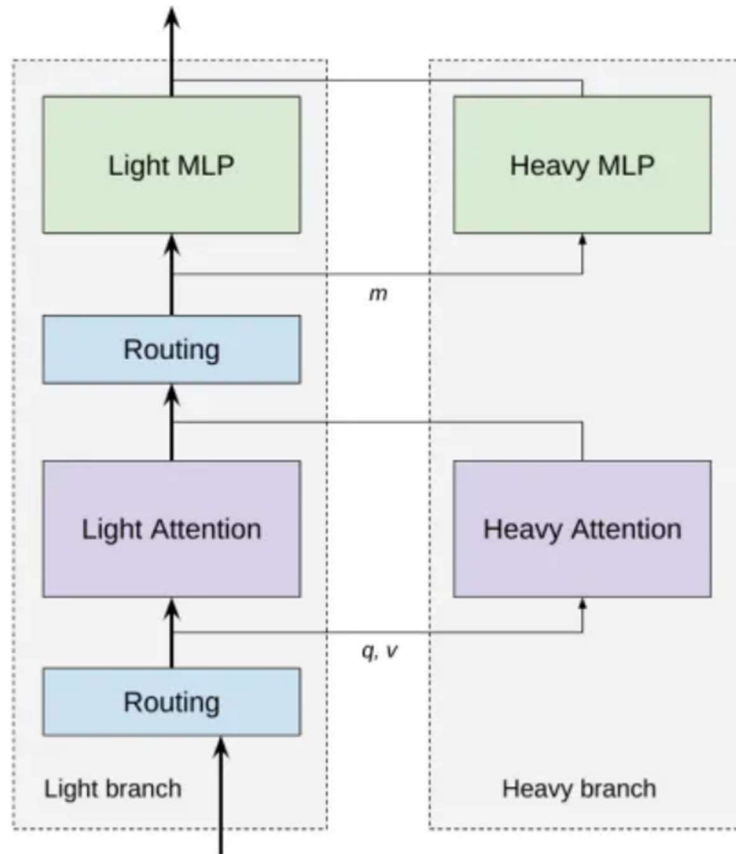


Left: FlashAttention uses tiling to prevent materialization of the large $n \times n$ attention matrix (dotted box) on HBM. In the outer loop (red arrows), FlashAttention loops through blocks of the K and V matrices and loads them to SRAM. In each block, FlashAttention loops over blocks of Q matrix (blue arrows), loading them to SRAM, and writing the output of the attention computation back to HBM. Right: 7.6x speedup. [FlashAttention paper](#)

3) Multi-Query attention(MQA)[\[참고\]](#): 추론 속도를 향상시켜준다. 기존의 Multi-Head Attention은 각 K 와 V 행렬마다 다른 선형 레이어를 가진다. MQA는 K 와 V 를 선형투영할 때 동일한 가중치를 모든 attention heads와 공유해 사용하는 방법이다. 따라서 (n, k) 와 (n, v) 크기의 가중치 행렬 두 개만 둔다. (추가 공부 필요)

4) Conditional Computation: 입력 길이가 임베딩크기보다 적으면 feedforward와 선형투영 시 bottleneck이었다. 이를 해결하기 위한 일반적인 방법은 모델 모든 파라미터를 중요한 토큰에만 전부 적용하고, 모든 토큰은 일부 파라미터만 적용하는 방법이다. 해당 방법은 64K와 같이 긴 입력에도 정확도와 속도가 좋았다.

다음 그림과 같이 Light와 Heavy 두 브랜치를 나눠서 적용하는 방법이다. Light 레이어는 모든 토큰에 적용되고, Heavy 레이어는 중요한 일부 토큰에만 적용된다.



An overview of a COLT5 Transformer layer with conditional computation. All tokens are processed by light attention and MLP layers, while q routed query tokens perform heavier attention over v routed keyvalue tokens and m routed tokens are processed by a heavier MLP. [CoLT5 paper](#)

5) **Large RAM GPUS:** 필수는 아니지만 large context와 LLM을 로드하고 학습하기 위해서는 많은 RAM이 필요하다. (몇 가지 오픈소스 LLM 모델들은 A100 GPU 80GB 4개를 사용해 학습되었다.)

3. 사전 학습된 모델 선택 기준

1) 모델 구조

앞서 언급한 두 가지 문제의 해결법들 중 positional encoding 기법(ALiBi, RoPE) 중 하나를 적용한 모델에 선택적으로 attention 계산의 복잡도를 줄여줄 수 있는 방법 한가지를 적용한 모델이라면 사용 가능하다.

2) Pre-train 된 학습 데이터

첫번째로 pretrain 시점에 한국어 데이터셋이 학습된 모델이어야 한다. 한국어 데이터셋이 포함되는 것도 중요하지만, 더 중요한 것은 한국어 데이터셋의 품질이다. 오픈 데이터셋들을 구성할 때 품질을 위해 보완 작업을 진행하지만, [Korean Hate Speech](#) 데이터셋과 같이 목적이 toxic speech detection인 데이터를 사전학

습된 모델이라면 파인튜닝 후에 성능이 좋지 않거나, 예상치 못한 이상한 답변을 생성해낼 수 있다.

사용 가능한 언어에 한국어가 없어도 한국어 데이터셋에 학습된 모델도 존재하므로 model card와 description를 잘 확인해야 한다.

3) 라이선스

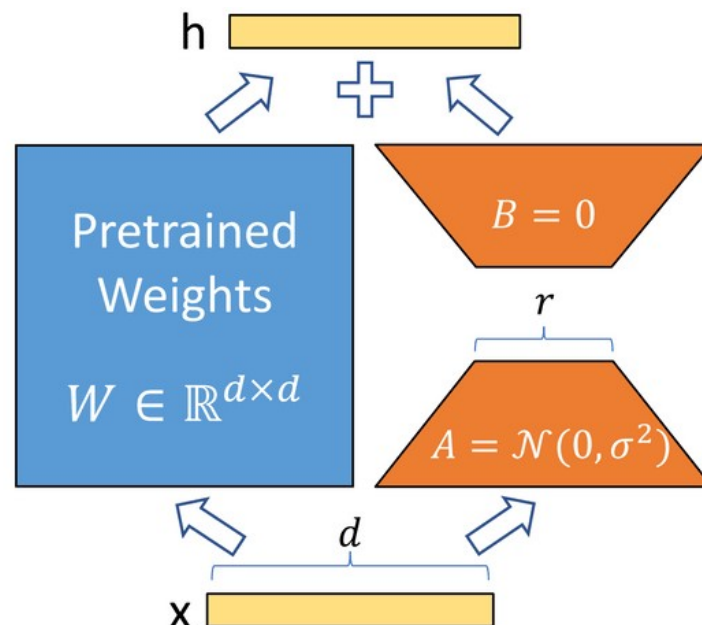
상업적 이용이 가능한 라이선스인지 확인이 필요하다.(e.g. apache 2.0 등)

[후보: KULLM, XGen-base, Polyglot-ko]

4. 파인튜닝 기법 및 태스크 설정

1) QLoRA 기법을 사용한 Finetuning[\[참고\]](#) [\[다른기법\]](#)

시간과 자원을 적게 사용하면서 실제 Finetuning을 진행해 테스트할 수 있는 기법이다. LoRA(Low-Rank Adaptation of Large Language Models) 기법은 LLM의 많은 파라미터 수로 인한 메모리 부족 문제를 해결하기 위해 고안 되었다. LoRA는 사전학습된 모델의 파라미터 행렬의 저 차원 근사치(row rank approximation)로, 새로운 행렬을 훈련 대상으로 함으로써 학습에 필요한 메모리 소비를 줄일 수 있다.



위 그림과 같이 거대한 행렬을 비교적 작은 행렬 2개로 분해 가능하다. 저 차원 근사한 행렬은 랭크수를 $[r]$ 로서 $[d] \times [r]$ 과 $[r] \times [d]$ 라는 두 개의 행렬이 된다. 이렇게 하면 트레이닝 대상의 파라미터 수를 d 의 제곱개에서 $2 \times d \times r$ 개 까지 줄이는데 가능하다.

QLoRA는 16bit 양자화, 양자화에 사용하는 정수의 이중 양자화, 그리고 페이지 최적화(일반 메모리 활용)을

통해 메모리를 획기적으로 줄여준다. QLoRA를 Guanaco 33B 모델에 적용해 48GB GPU 한개만으로 학습시키는 것이 가능하다. 또한 QLoRA는 적은 데이터를 사용해 학습하더라도 괜찮은 성능을 보여준다.

따라서 QLoRA는 back-bone 모델 선택과 Finetune 데이터셋 평가에도 사용될 수 있다. Huggingface에서 공개한 [peft](#) 라이브러리에 있는 일반 lora를 적용시킬 수 있다.

2) Instruction-tuning

GPT-3 모델을 pretraining 후 Instruction(과 선택: Input)과 Answer 페어 형식으로 학습시키는 instruction tuning을 했을 때, Zero-shot의 성능이 좋아진다는 구글의 [논문](#)이 있다. Finetune 시점에서 보지 못한 생소한 task도 해결할 수 있다는 것을 보여줬다.

다르지만 비슷한 케이스로 [phi-1](#) 모델에서 파이썬 교재 형태의 데이터로 사전학습한 뒤 Exercise과 Answer code 페어로 finetune한 모델이다. phi-1은 학습 데이터셋이 작음에도 불구하고 성능이 모델 크기와 학습 데이터 크기가 훨씬 큰 모델의 성능보다 좋았다고 한다. 해당 케이스는 좋은 품질의 데이터의 중요성도 보여주지만, 이러한 형태의 tuning을 통해 생소한 task에 대한 generalization이 되는 모습을 보여준다.

5. 파인튜닝 데이터셋 구성

1) 데이터셋 구성 - Instruction-following 데이터

오픈된 한국어 LLM인 KoGPT, Ko-vicuna 등의 모델들이 한국어 데이터셋을 구성한 방법은 다른 언어로 된 기존 데이터를 한국어로 번역한(DeepL 사용) 뒤 학습에 사용하는 방법이다. 이들의 오픈된 데이터셋이 존재하므로 그대로 사용해 볼 수 있고, 혹은 '2) Self-Instruct'에서 활용하기 위해 품질 개선 후 시드 데이터로 사용할 수 있다.

2) Self-Instruct[[논문](#)]

Stanford-Alpaca는 Self-Instruct를 통해 데이터를 생성하고 Instruction tuning이 된 모델이다. 논문에서는 데이터를 사람에게 의존하는 방식이 답변의 다양성, 창의성, 데이터의 양에서 제한적이어서 파인튜닝된 모델의 일반화를 방해한다고 말한다.

이들은 원하는 형식의 커스텀 데이터를 직접 구성한 뒤, 더 나은 LLM(ChatGPT, davinci-3 모델 사용)의 시드로 사용하여 추가 데이터를 생성하는 방법이다.(Alpaca에서 사용한 방식) 해당 방식에 대한 설명은 논문에 있고, 시드 데이터의 예제는 Stanford-Alpaca에서 찾아볼 수 있다.

2. OpenAI Embedding 분석

1. 학습 방식

- OpenAI의 최근 임베딩 모델인 'text-embedding-ada-002'은 성능을 제외하고, 학습 방법이나 데이터에 대해서 공개된 것이 없어 정확한 분석이 어렵다. 이전 임베딩 모델의 경우 쌍으로 나뉜 학습 데이터로 Contrastive Pre-Training을 진행했다. 하지만 성능과 가격에 차이가 있기 때문에 text-embedding-ada-002가 동일한 방식으로 학습되었다고 보기는 어렵다.
- OpenAI 커뮤니티에서 임베딩 모델과 관련된 실험을 많이 진행해본 [raymonddavey](#)의 말에 의하면 OpenAI의 임베딩은 영어-영어 유사도가 한글-영어의 유사도보다 높게 나오도록 학습되었다고 한다.

2. cosine similarity의 범위(-1~1)

- 여러 랜덤 문장으로 임베딩을 진행해본 결과 cosine similarity의 범위가 0.7~1의 범위로 계산이 된다. 임베딩 사이 각도의 범위가 매우 작다고 할 수 있다. 이는 한국어 문장 뿐만 아니라 영어 문장도 비슷한 결과를 보인다.
- 이러한 이유는 위에서 같은 언어일 수록 유사도가 높게 나오도록 학습했기 때문이라고 볼 수 있다.

```
irb(main):013:0> Embeddings.test_strings("I need to solve the problem with money","Anything you would like to share?")
=> 0.7614775318811315

irb(main):014:0> Embeddings.test_strings("I need to solve the problem with money","What is your financial situation?")
=> 0.8475256263838489

irb(main):015:0> Embeddings.test_strings("I need to solve the problem with money","Fraud")
=> 0.7632965853455049

irb(main):016:0> Embeddings.test_strings("I need to solve the problem with money","CitiBank")
=> 0.7823379047316411
```

3. text-embedding-ada-002의 영어 성능에는 문제가 없다

임베딩 모델 리더보드

- 태스크: Information Retrieval(right document retrieval for a given query.)
- 언어: 영어
- Metric: Normalized Discounted Cumulative Gain@k (k=10)
- 같은 언어가 높은 유사도를 가졌어도 정보 검색 태스크의 성능에는 문제가 없는 것으로 확인됐다.
- 예전 임베딩 모델 중 가장 성능이 좋은 모델인 davinci-001 보다 Retrieval 성능이 더 좋고 가격도 싸다.
- Huggingface의 Retrieval Task에서 sota는 아니지만 7등으로 좋은 성능을 보여주고 있다. Retrieval에서 가장 좋은 성능을 보여주는 모델인 multilingual-e5-large와 2.1정도 차이밖에 나지 않는다.
- 한글 임베딩이 가능함과 동시에 최대 길이가 8k가 넘는 오픈 임베딩 모델은 존재하지 않는다.

4. 학습 데이터로 인해 text-embedding-ada-002에 bias가 존재

- 아프리카계 미국인보다 유럽계 미국인 이름에 긍정적 감정을 가지는 경향이 있다.
- 흑인 여성에게 부정적인 고정관념을 가지는 경향이 있다.
- 이 외에 보이지 않는 편향이 존재할 수 있다.

5. 한국어 질문으로 세션이 잘 안걸리는 이유

1. BPE(Byte Pair Encoding)

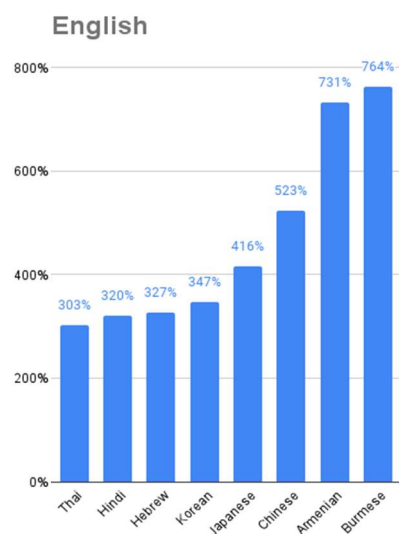
- 한국어와 영어를 BPE로 토큰나이징을 할 경우가 다르다.
- 예를 들어 **웹투비**의 경우 웹->토큰 2개, 투->토큰 3개, 비->토큰 1개로 나뉘지게 된다. 따라서 **웹툰**과의 유사도가 높아진다.
- WebtoB로 토큰나이징 할 경우 Web, to, B로 토큰나이징 된다.

2. 영어로 이루어진 주요 Keyword들

- WebtoB, Jeus 등 주요 키워드 들이 영어로 이루어져 있고, 임베딩을 할 때 위에서 말했던 것처럼 [Web, to, B]와 [Je,us] 등의 토큰으로 나뉘지게 된다.
- 그래서 키워드를 한글 웹투비나 제우스로 검색을 하게 되면 영어로된 키워드를 잘 찾지 못하고, 주변에 있는 단어들에 초점을 더 준다.
- 이러한 고유 명사는 단어의 변형 없이 웹투비 또는 WebtoB를 그대로 쓰기 때문에 후처리로 웹투비를 찾아 WebtoB로 변형해주면 큰 문제가 되지 않는다.

3. 외래어(스택트레이스, 스레드 etc.)

- OpenAI의 gpt3.5, gpt4의 토큰나이저의 사전에 한국어가 많지 않다. 아래 그림에서 알 수 있듯이, gpt4가 한국어를 처리하는 데 드는 비용은 영어의 3.5배이다. 같은 문장이어도 한국어로 하면 토큰이 3.5배정도 더 많다고 대략적으로 알 수 있다.



- '스레드' 같은 외래어의 경우 4개의 토큰으로 나뉜다. 이렇게 하나의 뜻을 가진 단어를 여러개의 토큰들로 나뉜 토큰들을 임베딩 했을 때 가지는 의미는 같은 단어를 영어'thread'로 임베딩 했을 때 가지는 의미와 거리가 멀어진다.

- **[가설]** 외래어같이 여러개의 토큰으로 나뉘는 경우 어텐션 구조에서 주변 단어들의 어텐션 스코어가 더 커져서 결과적으로 주변 단어의 중요도가 높아지는 것 같다.

한국어 세션에 이러한 외래어가 많아지면 설정이나 방식같이 주변 단어의 주요도가 높아지고, 마찬가지로 질문의 임베딩에서도 '설정'의 주요도가 높아져 '세션'보다 '설정'이 들어간 문서의 유사도가 높게 나오는 것 같다. 그래서 문서에 나와있는 형식 그대로 질문을 하거나 외래어가 주요 키워드인 경우 주변에 추가 단어 없이 외래어 키워드만으로 검색해야 외래어 키워드에 집중해서 결과를 반환하는 것 같다.

'JEUS에서 중앙집중식 세션방식을 설정하는 방법 알려줘'과의 유사도

- 설정
- 세션

```
•[14]: # 설정
      query_emb@kor_emb.T
```

```
[14]: 0.8156044460032976
```

```
•[15]: # 세션
      query_emb@kong_emb.T
```

```
[15]: 0.7795651342871465
```

6. 결론

- text 임베딩 모델들이 영어 데이터셋으로만 테스트 되어 다른 언어에서의 성능은 확실하지 않지만, 전체적인 성능 지표를 봤을 때 영어에서 좋은 성능을 내는 모델인 것은 확실하다.

- 길이와 한국어가 지원이 되는 모델로는 OpenAI 임베딩 모델을 대체할 공개 모델이 없다.

- BPE가 다른 언어를 쪼갰을 때 쪼개진 의미가 서로 다르기 때문에 문서와 검색문장이 서로 같은 언어로 임베딩 됐을 때 retrieval 성능이 유지된다.

- 외래어가 포함된 문서나 질문의 경우 주변 단어의 의미에 의존하게 되어 이들이 가지는 임베딩이 의도한 바와 다르게 작동할 수 있다.

3. OpenAI API의 functions 사용법

functions는 ChatCompletion API의 optional 파라미터로, 모델에게 함수의 기능을 설명할 때 사용할 수 있다. 해당 파라미터는 모델이 함수의 기능에 준수하면서 들어갈 인자(input 혹은 argument)를 생성할 목적으로 만들어졌다. 주어진 함수에 맞는 입력을 생성해 줄 뿐 함수를 실행해 주지는 않는다. 개발자의 입맛에 맞게 ChatGPT가 생성한 입력으로 함수를 실행할지 정할 수 있다.

여러 개의 함수가 **functions**에 주어지면 모델이 함수를 실행할 필요가 있는지 먼저 결정한다. 함수의 입력이 생성되었으면 **response**에 "finish_reason": "function_call" 이 포함된다.

모델이 항상 함수를 사용하도록 강제하려면 **function_call** 파라미터에 실행을 강제하고자 하는 함수 이름을 다음 형식으로 넣으면 된다.

```
{"name": "FUNCTION_NAME"}
```

반대로 모델이 함수를 사용하지 않도록 강제하려면 다음과 같이 **function_call** 파라미터에 "none"을 주면 된다.

```
{"name": "none"}
```

functions 의 형식

```
function = [
  {
    "name": "함수이름",
    "description": "함수의 설명, 언제 불러야 하는 지, 출력에 대한 설명",
    "parameters": {
      "type": "object",
      "properties": {
        "파라미터이름1": {
          "type": "파라미터타입1",
          "description": "파라미터에 대한 설명, 어떻게 사용되어야 하는지 등1",
        },
        "파라미터이름2": {
          "type": "파라미터타입2",
          "description": "파라미터에 대한 설명, 어떻게 사용되어야 하는지 등2",
        },
      },
    },
    "required": ["필수입력", "파라미터 리스트"]
  }
]
```

functions 예

Context와 질문을 찾을 때 답변과 답변을 어디서 찾았는지 answer: string, citation: string 형태로 반환하는 함수를 사용해보자.

function의 명세

```
function = [
  {
    "name": "AnswerCitation",
    "description": "답변과 인용글 입력으로 받습니다. 질문에 대한 답변과 답변에 대한 인용글을 출력합니다.",
    "parameters": {
      "type": "object",
      "properties": {
        "answer": {
          "type": "string",
          "description": "질문에 대한 답변",
        },
        "citation": {
          "type": "string",
          "description": "답변에 대한 인용 글",
        }
      }
    },
    "required": ["ciation"]
  }
]
```

OpenAI API call

```
MODEL = "gpt-3.5-turbo-0613"
response = openai.ChatCompletion.create(
    model = MODEL,
    messages=[
        {"role": "system", "content": "당신은 먼저 질문에 대한 답을 합니다. 그 다음에 답변에 대한 글의 내용을 그대로 인용하시오."},
        {"role": "user", "content":
            """다음 글을 사용해 질문에 답하세요.

제 이름은 Jason Liu이고 캐나다 토론토에서 자랐지만 중국에서 태어났습니다.
예술고를 나왔지만 대학에서는 전산수학과 물리학을 전공했다.
coop의 일환으로 저는 Stitchfix, Facebook을 비롯한 많은 회사에서 일했습니다.
또한 University of Waterloo에서 Data Science 동아리를 시작했고 2년 동안 동아리 회장을 역임했습니다.
질문: 저자는 대학시절 어떤 것들을 했나요?"""},
        {"role": "assistant", "content": '''대학시절 저자는 전산수학과 물리학을 전공했으며, University of Waterloo에서 Data Science
동아리를 시작하고 2년 동안 동아리 회장을 역임했습니다. (인용: "대학에서는 전산수학과 물리학을 전공했다.
또한 University of Waterloo에서 Data Science 동아리를 시작했고 2년 동안 동아리 회장을 역임했다.'''},
        {"role": "user", "content": """질문: 저자는 어느 회사를 다녔나요?
답: 답변에 대한 인용을 반드시 작성하고, 인용의 단어를 변형 없이 사용하세요."""}
    ],
    temperature=0,
    functions=function
)
```

```

{
  "id": "chatcmpl-7ZUNgKlPOmhPZxwpOtnrRoRFHe710",
  "object": "chat.completion",
  "created": 1688692594,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": null,
        "function_call": {
          "name": "AnswerCitation",
          "arguments": "{\n  \"answer\": \"저자는 Stitchfix, Facebook을 비롯한 많은 회사에서 일했습니다.\",\n  \"citation\": \"저는 Stitchfix, Facebook을 비롯한 많은 회사에서 일했습니다.\"\n}"
        },
        "finish_reason": "function_call"
      }
    ]
  },
  "usage": {
    "prompt_tokens": 543,
    "completion_tokens": 67,
    "total_tokens": 610
  }
}

```

예제에 대한 response

Limitations

1. **Hallucination**: functions에 주어지지 않은 함수를 지어내어 이상한 답변을 출력할 수 있다.
 - 해결 방법: "주어진 function만 사용하라."라는 시스템 메시지를 줄 수 있다.
2. **프롬프트 의존성**: 주어진 function이 실행될 만한 답변이 출력되어야만 function의 입력을 생성해준다. functions는 모델의 출력을 원하는 형태로 후처리하는 용도로만 사용될 수 있으며, 이는 결국 모델에 전달되는 프롬프트에 의존할 수밖에 없다. 하지만 함수에 필요한 내용이 답변에 존재한다면 이를 바탕으로 정확한 SQL 문을 짜주거나, 필요한 내용을 용도별로 나눠 여러 개의 리스트 형태로 받는 formatting이 가능하다.
3. **Context length 제한**: API에 넣은 function의 명세는 프롬프트의 system 메시지로 들어간다. 따라서, function 또한 컨텍스트의 길이에 포함되어 예상치 못한 context length limit 오류가 발생할 수 있다.

참고 링크

1. [Openai Function calling 독스](#)
2. [Openai cook-book Functions 주피터 예제](#)

4. 반사실적 태스크를 통한 언어모델의 능력과 한계 탐구[논문]

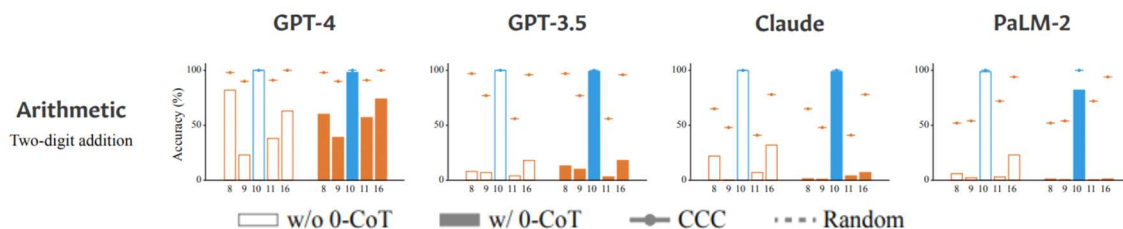


[그림 1] 산수에서 사실적과 반사실적의 예.

왼쪽부터 문제, 사실적, 반사실적 태스크의 예시와 [0-shot chain-of-thought](#) 프롬프트에 대한 GPT-4의 성능을 보여준다.(파란색이 사실적, 주황색이 반사실적 태스크)

반사실적 태스크란?:

- 과거에 일어난 일을 사실이라고 한다. '반사실적'은 과거에 일어난 사실을 반대로 생각하는 것이 된다.
- 사람의 경우 가진 지식을 새로운 문제로 이전하여 유연하게 적용할 수 있다.
- 여기서 언어모델에 적용되는 **사실적 태스크**는 언어모델이 **학습 과정에서 봤을 유형**의 태스크이고, **반사실적 태스크**는 **학습 과정에서 겪어보지 못한 태스크**이고 답을 추론할 때 사실적 태스크에서 배운 지식을 응용해서 적용해야 하는 문제이다.
- 해당 논문에서는 언어모델이 추론 기능을 가지고 새로운 유형의 태스크 또한 풀 수 있는지, 아니면 사전 학습을 하는 동안 본 일부 태스크에만 특화된 것인지 반사실적 태스크를 통해 평가한다.



[그림 2] 두 자리 숫자의 덧셈

파란색은 사실적 태스크, 주황색은 반사실적 태스크에 대한 성능

테스트는 프롬프트 엔지니어링(zero-shot Chain of Thought와 'Let's think step by step, and...')을 적용한 성능과 적용하지 않은 성능, 그리고 CCC(Counterfactual Comprehension Check)를 통해 언어모델이 태스크의 추상적 개념을 이해하고 있는지 확인했다.

논문에서의 결론:

- 11개의 태스크 평가를 통해 반사실적 태스크에 대한 성능이 사실적 태스크에 대한 성능에 비해 많은 차이가 나는 것을 알 수 있다.
- 이를 통해 현재의 언어모델들은 추상적인 태스크에는 잘 대처하지만, 문제를 풀기 위해 종종 정해진 절차들에 의존하고 응용을 잘하지 못한다는 것을 알 수 있다.
- 반사실적 태스크가 언어모델이 기존에 학습한 개념과 얼마나 다르냐가 성능에 영향을 줬다. [그림 2]에서

볼 수 있듯 두 자리 숫자를 더하는 태스크에서 10진수는 완벽에 가까운 성능을 보여주지만 16, 8, 11, 9진수 순으로 성능이 점점 떨어지는 모습을 볼 수 있다. 16진수와 8진수는 컴퓨터 과학이나 RGB의 값 등에 사용되어 9진수나 11진수에 비해 친숙하므로 비교적 좋은 성능을 내는 것으로 볼 수 있다.

- 언어모델이 과정 없이 답만 외운 태스크일수록 'Think step by step'이나 Chain of Thought 프롬프트를 사용했을 때 오히려 성능이 낮아진다.

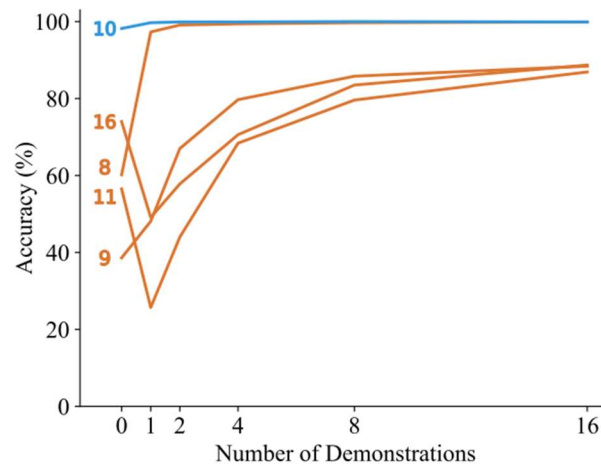


Figure 6: Two-digit addition accuracy when given different numbers of demonstration examples. The default-counterfactual gap reduces, but is not eliminated.

[그림 3] Few-shot learning에 대한 성능

Few-shot을 통한 반사실적 개념의 이해 테스트:

- Few-shot을 이용해 예시를 몇 개 줬을 때 반사실적 태스크의 성능이 향상되긴 했지만, 사실적 태스크와의 차이를 좁히지는 못했다.
- 16개의 예시를 줬을 때 성능이 수렴하는 것으로 보아 Few-shot으로도 응용문제를 푸는 데 한계가 있다.(모든 경우에 대해서는 확실하지 않음)
- 11진수와 16진수의 경우 1-shot에서 오히려 성능이 떨어지는데, 저자의 가설은 숫자와 문자를 같이 써서인 것 같다고 한다.

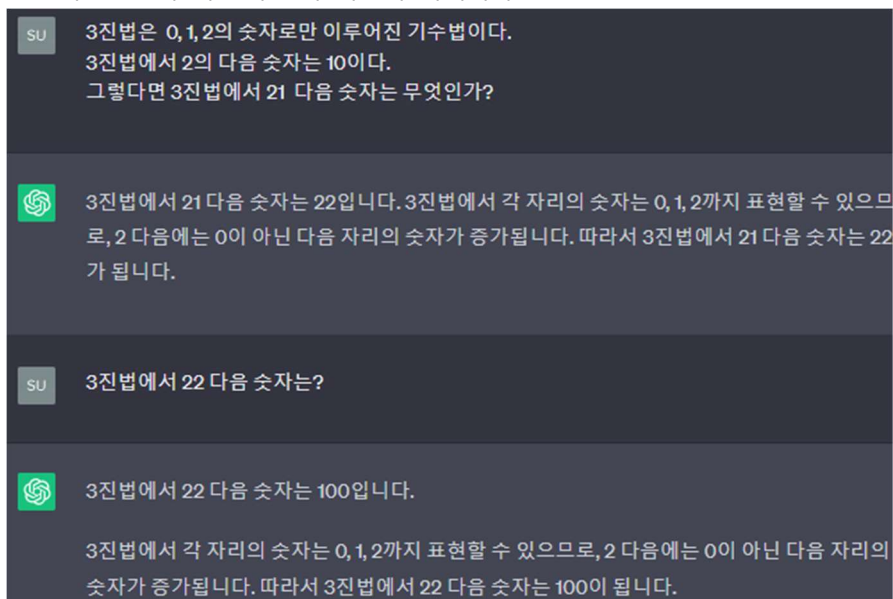
Mode	Prompt
Test	You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is 59+37? {Let's think step by step, and } end the response with the result in "\boxed{result}".
CCC	You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is the next number after 11A? Do this by counting the few preceding numbers and completing the sequence. End the response with the result.
Few-Shot CoT	<p>You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is 25+68? Let's think step by step, and end the response with the result in "\boxed{result}". We add the ones digits first. In base-11, 5+8=12. So the ones digit of the final sum is 2. We need to carry over the 1 to the tens place. Then we add the tens digits. In base-11, 2+6=8. Since we carried over the 1, 8+1=9. So the tens digit of the final sum is 9. Putting the digits of the final sum together, we get \boxed{92}.</p> <p>...[optionally more demonstrations in the same format]...</p> <p>You are a mathematician. Assuming that all numbers are in base-11 where the digits are "0123456789A", what is 59+37? Let's think step by step, and end the response with the result in "\boxed{result}".</p>

Table 1: Prompts for the arithmetic task. **{Let's think step by step, and }** is added only if 0-shot CoT is used (and the following e is capitalized without 0-shot CoT). We use the `\boxed{result}` syntax to wrap results because we found in preliminary experiments that the models tend to use this format even without this specification. The Few-Shot CoT prompt is used for the analysis in §5.5.

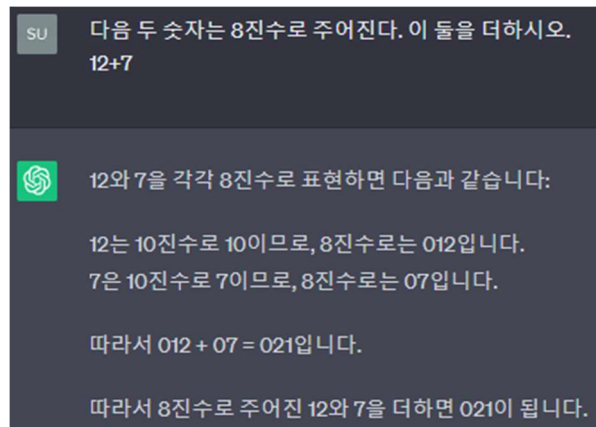
[그림 4] 테스트에 사용된 프롬프트

태스크의 난이도:

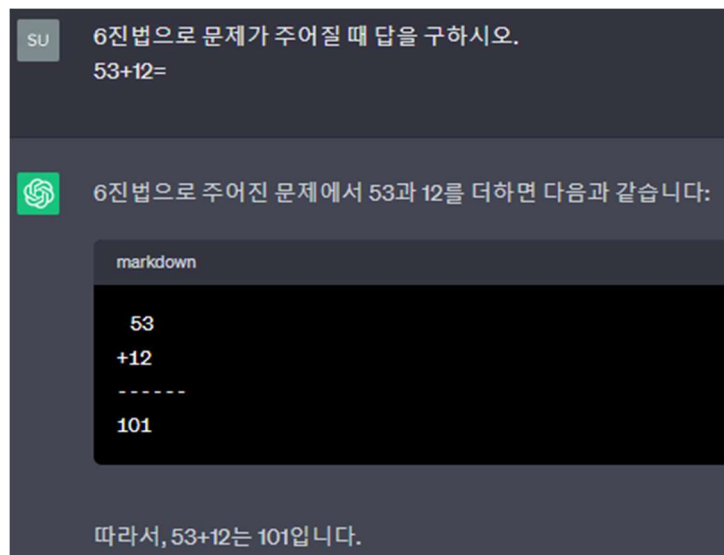
- 추상적 개념의 이해 – 훈련에서 거의 못 본 형태여도 CCC 프롬프트를 통해 쉽게 이해시킬 수 있다. 다음 그림은 ChatGPT의 3진법에 대한 추상적 개념의 이해이다.



- 문제 풀이 – 보통 훈련에서 학습한 개념 Real World에서 많이 보는 개념에 속한다. 문제의 숫자가 바뀌어도 풀 수 있는 정도로, 0-shot으로 풀 수 있는 난이도이다. 다음 그림은 8진법에서 문제 풀이에 대한 답변이다.

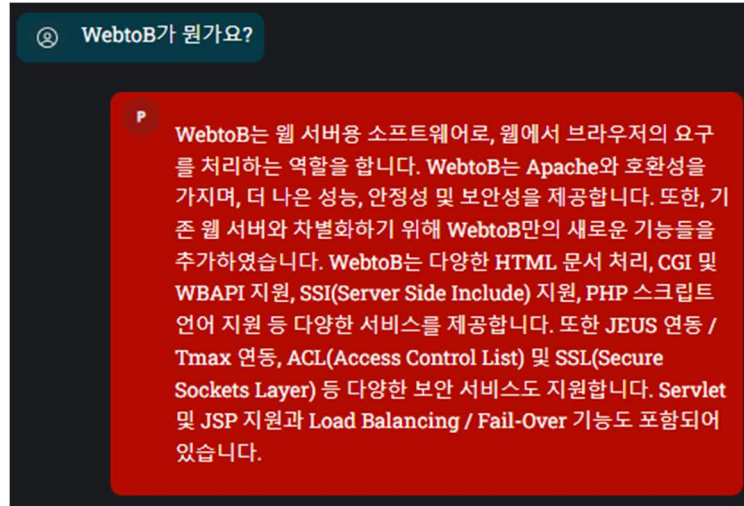


- 응용문제 풀이 – 훈련에서 학습한 개념을 응용해야 하는 난이도. Few-Shot을 통해 성능을 높일 수 있지만, 알고 있는 문제 풀이와 성능 차이가 난다. 다음 그림에서 6진법의 추상적 개념을 이해시킨 뒤에 질문해도 틀린 답변을 하는 모습을 볼 수 있다.



Pensive와 SoftGPT의 비교:

- 테스트 환경: Pensive는 WebtoB 5fix4 Installation Guide를 기준으로 실행됐다.
- 추상적 개념의 이해부터 응용문제 풀이까지 난이도 별로 진행했다.
- ▶ 질문1: WebtoB가 뭔가요?[추상적 개념]
- Pensieve



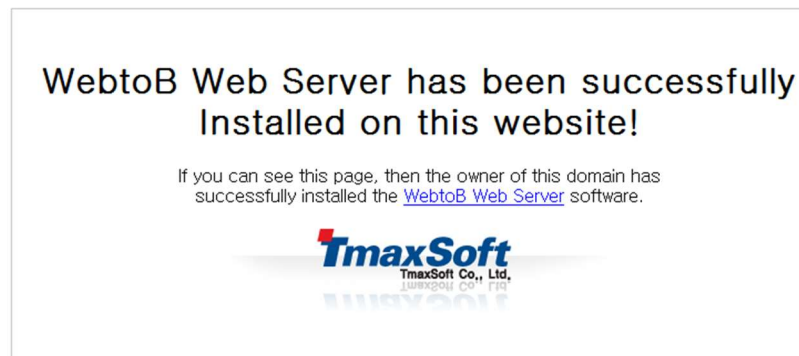
- SoftGPT

WebtoB는 'Web to Business'의 약자로, 웹에서 브라우저의 요구를 처리하여 주는 서버용 소프트웨어입니다. WebtoB는 Apache와의 호환성을 갖추며, 더 나은 성능, 안정성, 그리고 보안성을 제공합니다. 또한, WebtoB는 기존 웹 서버와의 차별을 위해 새로운 기능들을 추가하였습니다. WebtoB에서는 다양한 서비스를 제공합니다. HTML 문서들의 처리, CGI 및 WBAPI 지원, SSI(Server Side Include) 지원, PHP 스크립트 언어 지원, JEUS 연동 / Tmax 연동, ACL(Access Control List) 및 SSL(Secure Sockets Layer) 등 다양한 보안 서비스 지원, Servlet 및 JSP 지원(Standard Edition), Load Balancing / Fail-Over(Standard Edition) 등의 서비스가 있습니다.

WebtoB는 사용자에게 자유도가 높고 각자의 환경에 맞는 최적의 미세조정을 제공하여 웹 서버 운영자가 서버를 튜닝하여 효율과 성능을 높일 수 있도록 합니다. 또한, HTH 프로세스와 작업 스레드의 설정을 통해 CPU를 효율적으로 사용할 수 있습니다.

WebtoB의 특징과 설정에 대한 자세한 내용은 매뉴얼 제7장 'WebtoB 튜닝'에서 확인할 수 있습니다. 이를 참고하여 웹 서버 운영에 필요한 튜닝을 수행할 수 있습니다.

[그림 3.1] WebtoB 기동 화면



[그림 3.1]은 WebtoB가 기동되었을 때의 화면을 보여줍니다. 이 화면에서는 WebtoB 서비스에 접속할 수 있으며, 해당 화면을 통해 원하는 작업을 수행할 수 있습니다.

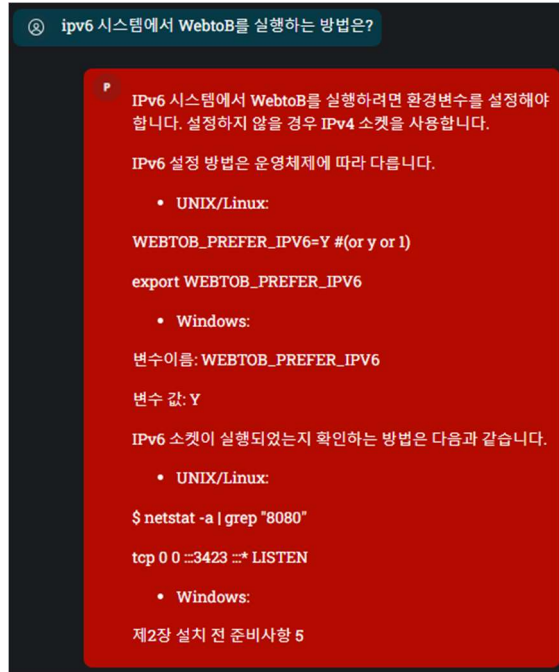
WebtoB는 대규모의 전문적인 웹 서비스 운영을 위해 개발되었으며, 다양한 기능과 설정을 제공하여 웹 서버 운영자가 효율적인 서비스를 제공할 수 있도록 도와줍니다.

전체복사

WebtoB라는 개념이 컨텍스트로 주어지고 이에 대한 질문을 했을 때, Pensieve와 SoftGPT에서 잘 답변하는 모습을 볼 수 있다. 이는 둘다 추상적인 개념 난이도의 태스크에서 좋은 성능을 보여준다고 볼 수 있다.

▶ 질문2: ipv6 시스템에서 WebtoB를 실행하는 방법은?[문제풀이]

- Pensieve



맨 마지막 줄에 질문과 연관이 없는 'Windows: 제2장 설치 전 준비사항 5'가 그대로 출력된 것으로 보아 질문을 이해하고 답을 했다고보다는 질문과의 유사도가 가장 높은 부분을 문서에서 그대로 가져왔다고 볼 수 있다.

- SoftGPT

```
IPv6 시스템에서 WebtoB를 실행하는 방법은 다음과 같습니다:
1. 운영체제에 따라 IPv6 환경 변수를 설정해야 합니다. 설정 방법은 다음과 같습니다:
- UNIX/Linux:
  - 터미널을 열고 다음 명령어를 입력합니다:
  ..
  export WEBTOB_PREFER_IPV6=Y
  ..
코드 복사

- 설정한 환경 변수를 적용하기 위해 다음 명령어를 입력합니다:
  ..
  source ~/.bashrc
  ..
코드 복사

- Windows:
  - 제어판을 열고 '시스템'을 클릭합니다.
  - '고급 시스템 설정'을 선택하고 '환경 변수' 버튼을 클릭합니다.
  - '새로 만들기'를 클릭하여 변수 이름을 'WEBTOB_PREFER_IPV6'로, 변수 값을 'Y'로 입력합니다.
2. WebtoB를 기동하기 위해 다음 명령어를 사용합니다:
  ..
  wsboot
  ..
코드 복사

3. WebtoB가 정상적으로 가동되면, 브라우저에서 다음 주소로 접속할 수 있습니다:
  ..
  http://[IPv6 주소]:[포트번호]/
  ..
코드 복사

여기서 [IPv6 주소]는 시스템의 IPv6 주소이고, [포트번호]는 WebtoB가 가동될 때 설정한 포트 번호입니다.
4. WebtoB를 종료하려면 다음 명령어를 사용합니다:
  ..
  wsdown
  ..
코드 복사

위의 단계를 따라하면 IPv6 시스템에서 WebtoB를 실행할 수 있습니다. 추가로 궁금한 사항이 있으신가요?
```

- Pensieve

답변에서 WebtoB를 IPv6로 설정하기 위한 환경변수 설정과 WebtoB를 실행하기 위한 방법 또한 설명하는 것으로 보아 질문을 이해하고 필요한 내용을 가져왔다.

또한 환경변수 설정을 그대로 가져오는 것이 아니라 이를 이해하고 두 스텝을 하나로 줄여준 것을 볼 수 있다. 윈도우 환경 변수 설정 또한 매뉴얼에 없는 내용을 알려준다.

'source ~/.bashrc' 명령어는 '.bashrc' 파일에 환경변수를 작성하여 자동으로 적용되도록 할 때 필요한 명령어이다. 이는 답변에 약간의 Hallucination이 포함되어 있는 것으로 볼 수 있다.

▶ 질문3: WebtoB의 포트를 8321로 변경해서 기동하는 방법 알려줘. [응용문제풀이]

⑧ WebtoB의 포트를 8321로 변경해서 기동하는 방법 알려줘.

P 문서에서는 WebtoB의 포트를 변경하는 방법에 대한 정보가 제공되지 않습니다. 따라서, 포트를 변경하는 방법에 대해서는 해당 제품의 공식 문서나 관련 지원팀에 문의하시는 것이 좋습니다.

- SoftGPT

WebtoB의 포트를 8321로 변경하여 기동하는 방법은 다음과 같습니다:

1. WebtoB를 설치한 디렉토리로 이동합니다.
2. 설치된 디렉토리에서 'conf' 폴더로 이동합니다.
3. 'conf' 폴더 안에 있는 'server.xml' 파일을 텍스트 편집기로 엽니다.
4. 'server.xml' 파일에서 다음과 같은 부분을 찾습니다:

```
-----  
<Connector port="8080" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443" />  
-----
```

코드복사

5. 위 부분에서 'port' 속성 값을 8321로 변경합니다:

```
-----  
<Connector port="8321" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443" />  
-----
```

코드복사

6. 'server.xml' 파일을 저장하고 닫습니다.

7. WebtoB를 기동합니다.

이제 WebtoB는 포트 번호 8321로 기동될 것입니다. 이 포트 번호를 사용하여 브라우저에서 WebtoB에 접속할 수 있습니다. 예를 들어, 'http://localhost:8321'과 같이 접속할 수 있습니다.

이렇게 WebtoB의 포트를 8321로 변경하여 기동하는 방법입니다. 추가로 궁금한 사항이 있으시면 말씀해주세요.

전체복사

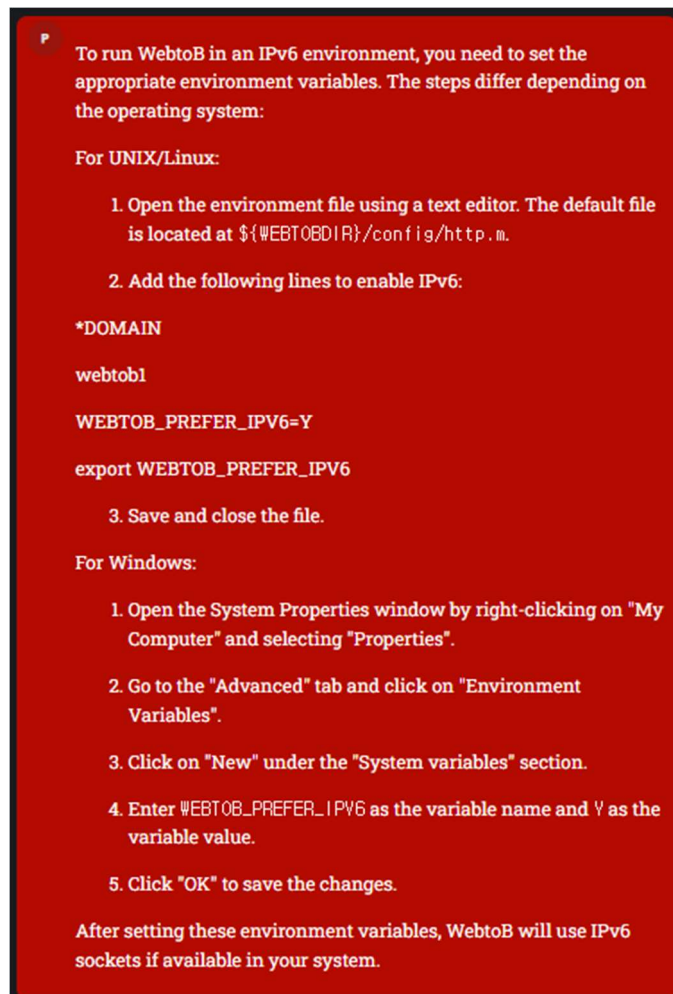
이 질문은 WebtoB의 환경 파일 <http.m>에 포트에 대한 설정이 있고, 포트 번호를 변경하고 컴파일을 해야 wsconfig로 적용된다는 여러 개념을 알고 응용해야 답변이 가능하다. Pensieve는 매뉴얼에 제공되지 않았다는 답변을 했다. SoftGPT의 경우 환경파일이 웹투비가 설치된 디렉토리에 존재한다는 개념과 환경파일을 변경해야한다는 개념까지 접근했지만, 환경 파일의 이름이나 내용을 Tomcat의 설정으로 답변했다.

- Pensieve 영어 매뉴얼 추가 테스트:

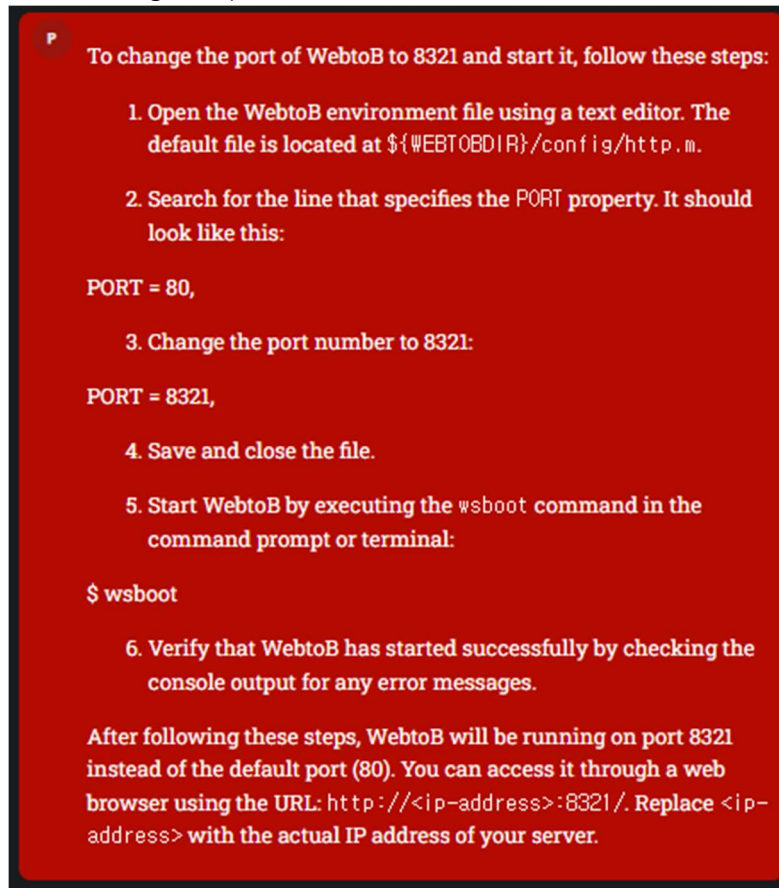
▶ 질문1: What is WebtoB?[추상적 개념]



▶ 질문2: How to run WebtoB in IPv6 environment?[문제 풀이]



▶ 질문3: Tell me how to change the port of WebtoB to 8321 and start it.[응용 문제풀이]



질문1은 이전과 비슷하게 잘 대답했다.

질문2의 경우 Unix에서의 설정이 틀렸다. 환경변수를 설정해야 하는데 WebtoB의 설정 파일을 말하고 있다.

질문3의 경우(8321포트가 실제로 쓰일 일은 없을것) 'http.m' 환경 파일을 제대로 말하고 있지만, 컴파일 절차는 스킵했다. OpenAI의 언어 모델들은 한국어 보다 영어를 사용할 때 성능이 더 좋다는 것을 알 수 있다. 또한 SoftGPT에서 응용 문제에 대한 답변을 개선할 수 있는 여지를 보여준다.

5. Text-to-SQL

1. text-to-sql과 중요한 점

Text-to-SQL이란 데이터베이스와 자연어 질문이 주어지면 질문에 맞는 SQL 쿼리를 출력해주는 Task이다. text-to-sql 태스크를 위한 모델을 만들 때 풀어야 하는 두가지 문제가 있다. 첫번째는 데이터베이스에는 보통 테이블이 여러개가 있고 테이블간의 관계가 존재하는데, 정확한 쿼리를 출력하기 위해서는 테이블간의 복잡한 관계를 모델이 이해해서 질문을 답하는 데 필요한 관계를 추론해야 한다는 점이다. 두번째는 사용자의 데이터베이스 구조가 다양하기 때문에 일반화 성능이 좋아야 한다.

2. Spider 데이터셋

Spider 데이터셋에 Multi-table queries 관련 데이터가 있다. 단순한 Select, Where부터 group by, order by, nested query 등 여러 복잡한 쿼리에 대한 데이터를 가지고 있고 또한, 정규화 성능을 측정할 수 있도록 train과 test set으로 나뉘어서 학습 데이터에 없는 새로운 데이터베이스와 쿼리 데이터가 있다. 관계 이해와 일반화 성능을 측정하기 좋은 데이터셋이다.

Spider와 비슷한 데이터셋으로 SParC가 있다. 한개의 테이블로 여러 질문을 하는 WikiSQL과 Spider를 대화형으로 만든 CoSQL 또한 존재한다.

2. 성능을 높이는 방법

모델의 구조 및 미세튜닝

Encoder와 Decoder 각각에 다른 목적을 가지고 연구가 진행되었다. Encoder의 경우 데이터베이스의 테이블들 간의 관계를 모델에 더 잘 이해시키기 위한 접근이 많았다. Encoder에 Graph-Aware Layers를 적용시키는 Graphix-T5 모델이 있는데, 2020년도 모델인 T5를 베이스로 사용했음에도 뒤에서 설명된 PICARD와 함께 적용되어 Spider1.0에서 준수한 성능을 보여준다.

Decoder의 경우 syntax에 맞는 SQL을 출력하기 위한 접근이 있었다. 2017년에는 SQL Query문의 구조에 대한 사전 지식을 기반으로 SELECT / WHERE에 어떤 값을 넣을 지 판단하는 Subtask 형식을 사용했는데, LLM이 발전하면서 추론 능력이 향상이 되었고, decoding 전략 또한 바뀌었다. 언어 모델에 자연어를 주고 SQL을 출력하라고 하면 많은 토큰들 중에 고르기 때문에 틀린 문법이나 잘못된 코드를 뱉을 때가 많았다. 코드를 돌리기 전에 컴파일러로 Syntax를 확인하는 것과 비슷한 방식을 Decoding 단계에 적용한 것이 PICARD(Parsing Incrementally for Constrained Auto-Regressive Decoding) 이다. Decoding을 진행하면서 후보들 중 Lexical, Grammatical하게 SQL의 Syntax를 따르는 후보들로만 parsing을 해서 최종 결과를 출력하는 방식인 것 같다.

7	Graphix-3B+PICARD (DB content used)	77.6
Sep 13, 2022	Alibaba DAMO & HKU STAR & SIAT	
	(Li et al., AAAI'2023) code	

학습 데이터 및 프롬프트 엔지니어링

프롬프트 엔지니어링은 일반적으로 inference에서 적용되지만, 학습에서도 필요하다. 학습 파이프라인을 구성할 때 언어모델이 데이터베이스 스키마를 보다 쉽게 이해할 수 있는 스키마 데이터 설명 템플릿을 만들어야 하기 때문이다. 이렇게 만들어진 템플릿은 학습이 완료된 후 inference에서도 적용이 될 수 있다.

템플릿 방식은 아니지만, Spider 스코어보드를 보면 상용화된 LLM(gpt 3.5, 4; codex 등)에 In-Context-Learning과 Self-Correction을 할 수 있는 프롬프트의 체인 그리고, 복잡한 태스크를 여러개의 작은 태스크로 분해함으로써 Spider 데이터셋에서의 성능을 높였다(하지만 해당모델들은 Spider 데이터셋을 학습과정에서 봤을 가능성이 존재한다).

Rank	Model	Test
1 Apr 21, 2023	DIN-SQL + GPT-4 <i>University of Alberta</i> (Pourreza et al., '2023) code	85.3
2 Jun 1, 2023	C3 + ChatGPT <i>Anonymous</i>	82.3
5 Apr 21, 2023	DIN-SQL + CodeX <i>University of Alberta</i> (Pourreza et al., '2023) code	78.2

3. 출시된 제품

OpenAI의 codex와 gpt3.5, 4(Copilot, Copilot X)

codex는 천만개의 공개 레포지토리 데이터를 사용해 학습된 모델이다. 해당 모델은 Copilot에서 사용된 모델로 다양한 프로그래밍 언어와 SQL부터 셸 스크립트등의 생성 또는 완성 태스크가 가능한 모델이다. 현재 codex는 OpenAI에서 deprecated 되었으며, 마이크로소프트에서 사용이 가능한 것으로 보인다. gpt3.5와 4모델 또한 이러한 태스크에 좋은 성능을 보인다.

USER

```
# PostgreSQL
# Table albums, columns = [AlbumId, Title, ArtistId]
# Table artists, columns = [ArtistId, Name]
# Table media_types, columns = [MediaTypeId, Name]
# Table playlists, columns = [PlaylistId, Name]
# Table playlist_track, columns = [PlaylistId, TrackId]
# Table tracks, columns = [TrackId, Name, AlbumId,
MediaTypeId, GenreId, Composer, Milliseconds, Bytes,
UnitPrice]

# Create a query for all albums by Adele
```

ASSISTANT

To retrieve all albums by Adele, you can use a JOIN operation between the `albums` and `artists` tables. Here's the query:

```
```sql
SELECT albums.AlbumId, albums.Title, artists.Name
FROM albums
JOIN artists ON albums.ArtistId = artists.ArtistId
WHERE artists.Name = 'Adele';
```
```

This query will return the `AlbumId`, `Title`, and `Name` of the albums by Adele.

Chat

Model

gpt-3.5-turbo-0613

Temperature 0.2

Maximum length 256

Stop sequences
Enter sequence and press Tab

Top P 1

Frequency penalty 0

Presence penalty 0

[TEXT2SQL.AI](#)

다양한 데이터베이스의 text-to-sql 태스크와 Regex 및 엑셀 수식 질문 답변 태스크가 가능한 서비스이다. OpenAI의 Codex 모델을 사용하고 있다. OpenAI 모델을 사용하기 때문에 다양한 언어지원과 생성된 답변을 설명하는 기능 또한 가지고 있다.

4. Back-bone 모델의 성능

모델의 reasoning 능력과 성능이 좋을 수록 주어진 데이터베이스의 스키마와 질문과의 관계를 더욱 잘 이해할 수 있다. 그래서 성능이 높은 모델을 back-bone 모델로 사용해 미세튜닝을 진행하면 같은 데이터셋과 학습방식을 사용하더라도 Text-to-SQL 태스크에서도 보다 나은 성능을 보일 수 있다.

좋은 성능의 공개 모델들은 대부분 영어에 유리한 설정과 성능을 가지고 있고, 한국어보다 영어로 된 공개 데이터셋과 자료들이 많기 때문에 영어 환경에서 진행하는 것도 고려해볼만한 사항이다.

5. 결론

연구에 사용된 back-bone 모델이 최근 언어모델과 성능 차이가 있기 때문에 다른 언어모델을 미세튜닝하여 성능을 향상해볼 수 있다. 또한 템플릿의 경우 비슷한 경험을 가지고 있기 때문에 다양한 실험을 통해 성능을 향상해볼 여지가 많다.