**~~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~~**

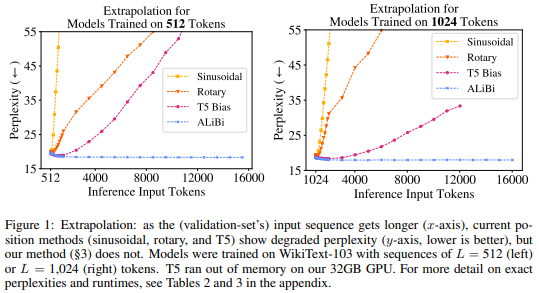
~~Fine-tuning이 가능한 모델을 위해서는 두가지 문제를 해결해야 한다.~~

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

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

**~~a. Length Extrapolation 문제:~~** ~~기존 트랜스포머의 sinusoidal postitional 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.~~

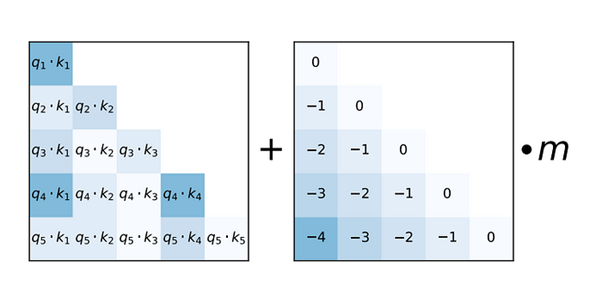
**~~1) ALiBi(Attention with Linear Bias): ​~~**~~sin positional encodings을 토큰 pair의 거리에 비례해 attention socre에 패널티를 부여하는, 학습이 필요 없는 bias를 추가해준다. 해당 방법은 토큰끼리의 거리가 멀어질 수록 서로 간의 관련성 혹은 중요성이 떨어질 거라는 직관에 기반한 방법이다.~~



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

~~~~

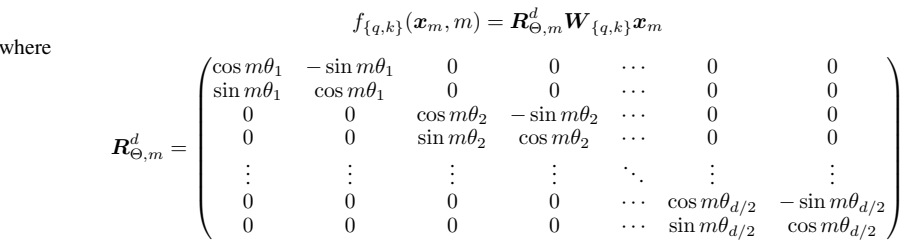
**~~[Figure 2]~~**



**~~[Figure 3]~~**

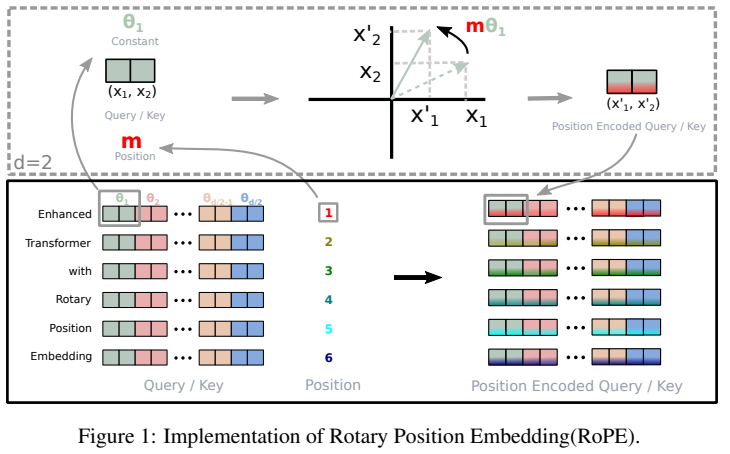
~~모델이 n개의 head를 가지고 있으면 2-8/n으로 시작한다. 예를 들어 8개의 head가 있는 경우 1/2로 시작해 1/22, 1/23…., 1/28이런 식으로 적용된다.  
이렇게 구한 slope는 여러 도메인, 여러 모델사이즈에서 다양하게 적용할 수 있으며, 해당 논문의 저자들은 새로운 모델에 따라 slope를 구할 때 튜닝이 필요하지 않다고 믿는다.~~

**~~2) RoPE(Rotary Embedding)[[참고]](https://blog.eleuther.ai/rotary-embeddings/" \o "이해를 돕기 위해 Visual Intuition의 그래픽 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.~~

~~~~

~~~~

~~RoPE를 적용한 self-attention 수식은 다음과 같다.  
~~

~~~~

~~RoPE와 Sinusoidal embedding과의 차이점은 두가지가 있다.~~

~~1. Sinusoidal 임베딩은 각 좌표별로 따로 적용되지만, Rotary 임베딩은 좌표 쌍을 혼합하여 적용한다.~~

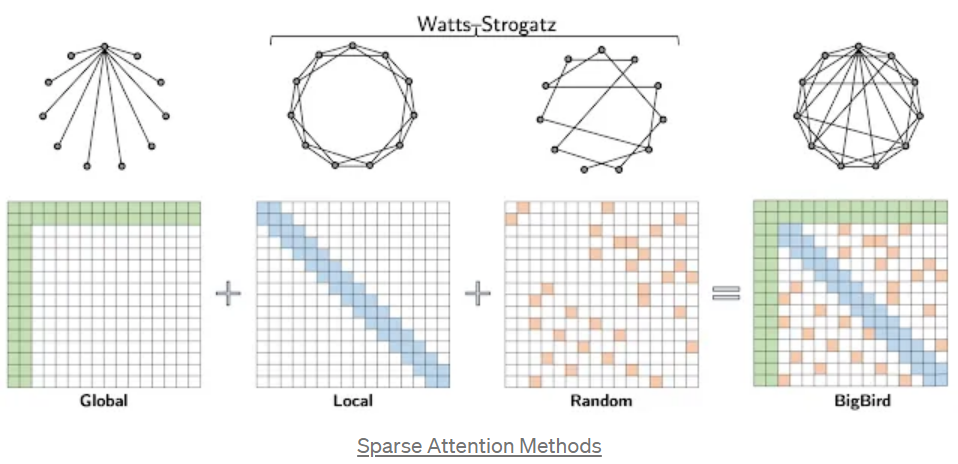
~~2. Sinusoidal 임베딩은 코사인과 사인을 더하여 사용하지만 Rotary 임베딩은 이를 곱해 사용한다.  
단점은 RoPE를 계산하고 적용하는 시간이 Sinusoidal보다 2~4배정도 느리다는 점이다.~~

**~~3) “방해토큰” 완화를 위한 방법[[참고]](https://arxiv.org/abs/2305.16300" \o "Landmark Attention):~~** ~~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](https://arxiv.org/pdf/2305.19370.pdf)):~~** ~~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 희소성를 추가해 줌으로써 Attention의 계산 복잡도를 n의 제곱이 아닌 n에 선형적으로 비례할 수 있게.~~

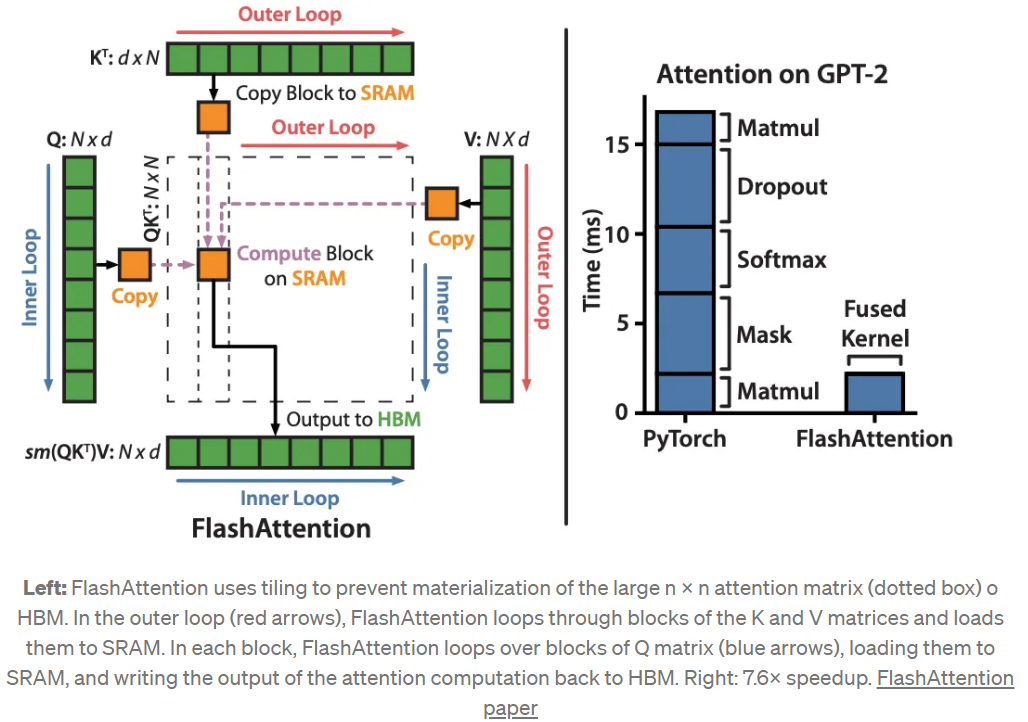
~~~~

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

~~Big Bird Attention은 세가지 sparse attention method를 모두 합친 방법이다.~~

**~~2) FlashAttention~~**~~: attention layer에서 계산하기 위한 중간 결과의 크기는 (n x 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.~~

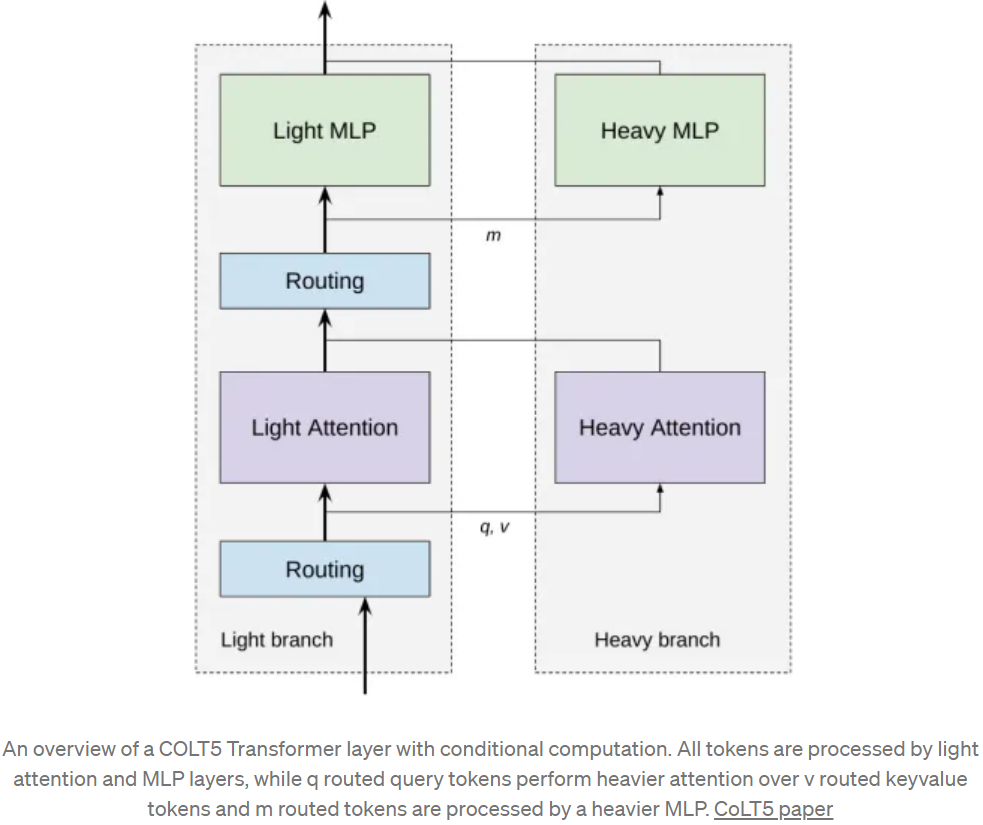
~~FlashAttention 알고리즘은 Q, K, V 행렬들을 블록으로나눠서 SRAM으로 로딩한 뒤, 해당 블록에 대한 어텐션 결과를 계산해 최종 결과 값을 SRAM에 저장한다. 이러한 프로시져를~~ **~~tiling~~** ~~이라고 부른다. FlashAttention은 학습과 추론의 속도를 향상시켜준다. (Pytorch 2.0에는 flash-attention이 내장되어 있다.)~~



**~~3) Multi-Query attention(MQA)[[참고]](https://arxiv.org/abs/2211.05102" \o "Efficiently Scaling Transformer Inference)~~**~~: 추론 속도를향상시켜준다. 기존의 Multi-Head Attention은 각 K와 V 행렬마다 다른 선형 레이어를 가진다. MQA는 K와 V를 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) Conditonal Computation~~**~~: 입력 길이가 임베딩크기보다 적으면feedforward와 선영투영 시 bottleneck이있다. 이를 해결하기 위한 일반적인 방법은 모델 모든 파라미터를 중요한 토큰에만 전부 적용하고, 모든 토큰은 일부 파라미터만 적용하는 방법이다. 해당 방법은 64K 와 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.~~

~~~~

**~~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](https://github.com/kocohub/korean-hate-speech) 데이터셋과 같이 목적이 toxic speech detection인 데이터를 사전학습된 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) 라이센스~~**

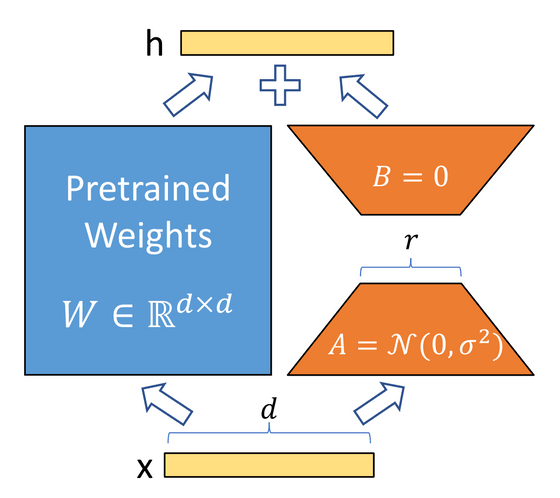
~~상업적 이용이 가능한 라이센스인지 확인이 필요하다.(e.g. apache 2.0 등)~~

~~[후보: KULLM, XGen-base, Polyglot-ko]~~

**~~4. 파인튜닝 기법 및 태스크 설정~~**

**~~1) [QLoRA](https://github.com/artidoro/qlora" \o "QLoRA 데모 깃허브) 기법을 사용한 Finetuning[[참고]](https://colab.research.google.com/gist/Beomi/a3032e4eaa33b86fdf8de1f47f15a647/2023_05_26_bnb_4bit_koalpaca_v1_1a_on_polyglot_ko_12_8b.ipynb" \o "koalpaca QLoRA 학습 코랩) [[다른기법]](https://arxiv.org/pdf/2304.01933.pdf)~~**

~~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]×[r]과 [r]×[d]라는 두 개의 행렬이 된다. 이렇게 하면 트레이닝 대상의 파라미터 수를 d의 제곱개에서 2×d×r개 까지 줄이는게 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](https://devocean.sk.com/blog/techBoardDetail.do?ID=164779&amp;boardType=techBlog" \o "peft의 lora) 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](https://arxiv.org/pdf/2109.01652.pdf)이 있다. Finetune 시점에서 보지 못한 생소한 task도 해결할 수 있다는 것을 보여줬다.~~

~~다르지만 비슷한 케이스로 [phi-1](https://arxiv.org/pdf/2306.11644v1.pdf)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]](https://arxiv.org/abs/2212.10560" \o "Self-Instruct: Alligning LM with Self Generated Instructions)~~**

~~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.~~