boostcamp ai tech

부스트캠프 AI Tech

Generation for NLP

수능형 문제 풀이 모델 생성

팀 NLP크민

권지수A_T7311, 김성은_T7318, 김태원_T7329
이다현_T7408, 이한서_T7425, 정주현_T7436

1 프로젝트 개요

주제	수능형 문제 풀이 모델 생성: 작은 AI 모델을 활용해 한국어 수능 시험(국어 및 사회 과목)을 풀이하는 Task. 한국어와 수능 시험의 특성에 최적화된 모델을 설계하여, 대형 모델(GPT, Claude, Gemini 등)과 경쟁할 수 있는 성능을 목표로 한다.
구현 내용	주어진 수능형 문제 형식의 데이터셋을 전처리 및 증강하여 Train Dataset을 구성하고, 사전 학습된 소형 모델을 Train Dataset에 맞게 파인튜닝 후 학습을 진행한다. 이후 학습된 모델로 Test Dataset에 대해 추론을 수행한다. 이 과정에서 최적의 데이터셋, 모델, 파라미터 및 프롬프트를 탐구한다.
개발 환경	- GPU: NVIDIA Tesla V100 32GB 서버 4개 - Tool: VS Code, Jupyter Notebook
협업 환경	- Zoom: 실시간 비대면 회의 - Github: 코드, 데이터 공유 및 버전 관리 - Notion: 역할 분담, 실험 가설 설정 및 경과 공유
구조도	code ├ requirements.txt ├ config.json ├ make_train_dataset.ipynb ├ data_processing.py ├ utils.py ├ train.py └ test.py data ├ aug_gemini.csv ├ aug_gpt.csv ├ classified_train_raw.csv ├ classified_train_processed.csv ├ classified_train_processed_filtered.csv ├ train_augmented.csv └ train_final.csv prompts ├ prompt_no_question_plus.txt └ prompt_question_plus.txt

2 팀 구성 및 역할

이번 프로젝트에서 텍스트 노이즈 분류는 권지수, 김성은 캠퍼가 맡아 데이터의 품질을 개선했으며, 텍스트 클리닝과 모델 경량화는 권지수와 이한서 캠퍼가 담당했다. EDA(탐색적 데이터 분석)와 Re-Labeling은 김태원과 이다현 캠퍼가 담당하며 데이터의 구조와 정확도를 개선했으며, Evol-Instruct LLM을 활용한 데이터 증강은 이한서 캠퍼까지 총 세 사람이 주도했다. 모델 탐색은 권지수, 김성은, 정주현이 수행하여 최적의 모델을 탐색했다. Prompt-Based 데이터 생성은 김성은, 이한서 캠퍼가 기여했다. 마지막으로, PM 역할은 김태원과 이다현이 맡아 프로젝트의 전반적인 진행을 관리했다.

이름	역할
권지수	Text/Label Noise Split, Text Cleaning, Model Searching, Model Compression
김성은	Text/Label Noise Split, Model Searching, Prompt-Based Generation
김태원	PM, EDA, Re-Labeling, Evol-Instruct LLM for Augmentation
이다현	PM, EDA, Re-Labeling, Evol-Instruct LLM for Augmentation
이한서	Model Searching, Data Augmentation, Prompt Engineering, Fine-tuning
정주현	Re-Labeling, ML Model Searching

3 프로젝트 수행 절차 및 방법

효율적인 협업을 위해 매일 아침 ZOOM 화상 회의를 통해 각자의 진행 상황과 당일 실험 계획을 공유하였으며, 오후 4시부터 7시까지 다시 ZOOM 회의를 통해 실험 결과를 실시간으로 공유하고 피드백을 주고받았다.

3.1 협업

a. Server	모든 서버를 Git으로 연동해 버전 관리, 유휴 상태의 서버에서 실험 진행
b. Git	개인별 branch에 작업한 코드 및 데이터 공유
c. Notion	가설, 구현 및 실험 결과를 기록해 진행 상황 공유
d. Submission	리더보드 마감 5일 전까지 자유롭게 제출, 이후 인당 2회씩 부여

3.2 프로젝트 타임 라인: 2024. 11. 11.(월)~11. 28.(목) (17일간)

- **(1~5 일차)**: 강의 수강, Baseline Code 분석 및 데이터 EDA
- **(6~12 일차)**: 모델 탐색 및 실험, 데이터 정제
- **(12~17 일차)** : 데이터 증강, 파라미터 및 프롬프트 실험

4 프로젝트 수행 과정 및 결과

4.1 데이터 프로세싱

데이터 프로세싱 과정에서는 수능형 문제를 포함한 다양한 데이터셋(KMMLU, MMMLU, KLUE MRC)을 결합해 Train 데이터 2031개와 Test 데이터 869개를 구성하였으며, 선택지 수를 5개로 통일하고 정답 분포를 균등화하는 등 데이터 정제를 통해 수능 형식에 맞는 일관성을 확보하려 했다. 또한, 데이터 증강 단계에서는 선택지 위치 무작위 변경, GPT API를 활용한 새로운 문제 생성, 그리고 KorQuAD와 같은 외부 데이터를 추가하여 데이터의 양과 다양성을 증대시켰다. 특히, 새로운 문제 생성은 Gemini API와 GPT-3.5-turbo를 사용하여 Train 데이터의 paragraph 기반 문제를 생성하고, Few-shot 학습으로 문제 생성 방식을 개선하는 등 다양한 접근 방식을 활용했다.

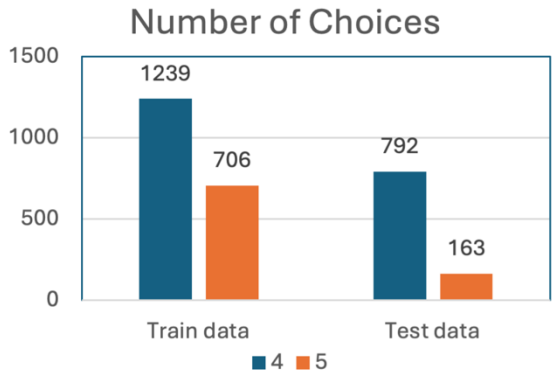
4.1.1 EDA

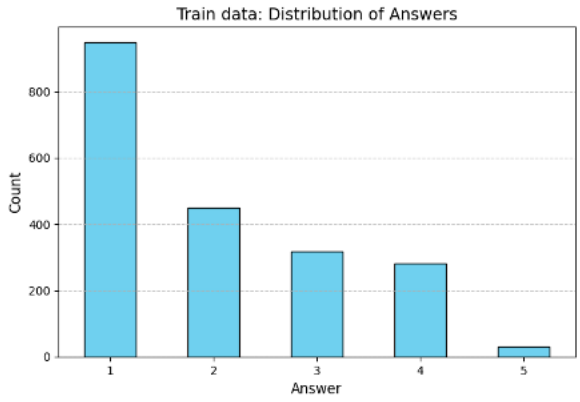
데이터는 수능형 문제로 Train data 2031개, Test data 869개로 이루어져 있다. 데이터는 수능의 국어 및 사회 영역(윤리, 정치, 사회)과 유사한 문제를 포함하고 있으며, KMMLU (Korean History), MMMLU (HighSchool 데이터 중 역사, 경제, 정치, 지리, 심리), KLUE MRC (경제, 교육산업, 국제, 부동산, 사회, 생활, 책마을) 데이터를 결합하여 구성되었다.

Trina data는 id, paragraph, problems, question_plus로 구성되어 있으며problems안에는 'question', 'choices', 'answer' key를 가진 dictionary가 들어있다.

header / key	paragraph	question	choices	answer	question_plus
내용	지문	질문	선지	정답	보기

수능 문제의 경우 모든 선지가 5개이지만, 본 데이터셋에서는 선지가 4개인 경우도 존재하였다. Train 데이터에서는 선지가 4개인 데이터가 792개, 5개인 데이터가 1239개로 나타났으며, Test 데이터에서는 선지가 4개인 데이터가 163개, 5개인 데이터가 706개로 나타났다. 또한, Train 데이터에서 정답 선택지의 비율은 불균등했으며, 1번이 정답인 경우가 가장 많았고 5번이 정답인 경우는 매우 드물었다.





4.1.2 데이터 정제

EDA를 바탕으로 선택지 수를 5개로 통일하였다. 수능 형식과 일치하도록 선택지 개수를 5개로 맞추어, 모델이 일관성 있는 학습을 할 수 있도록 했다. 선택지가 4개인 데이터에 5번째 선택지로 " "과 "정답 없음" 등 여러가지 버전으로 선택지를 추가하여 실험하였다. 그러나 선택지 개수의 통일은 모델 성능에 큰 영향을 미치지 않았다.

또한, 1번으로 편향되어 있던 정답의 개수를 균등하게 통일하였다. 불균등한 정답 분포를 조정하여 데이터셋의 균형을 맞춰 데이터의 편향을 줄이기 위한 시도였다. 하지만 이 또한 모델의 성능 향상에는 큰 기여를 하지 않았다.

4.1.3 데이터 증강

데이터 증강 과정에서는 다양한 접근 방식을 활용하였다. 선택지 위치를 무작위로 바꾸는 방법, GPT API를 활용한 새로운 문제 생성, 그리고 외부 데이터 활용의 세 가지 주요 전략이 사용되었다.

A. 선택지 위치 바꾸기

첫 번째로, 데이터 증강을 위해 선택지의 위치를 무작위로 바꾸는 작업을 수행하였다. 수능형 문제에서 모델이 특정 위치의 선택지가 정답일 확률이 높다는 편향을 학습하지 않도록 하기 위해서이다. 이를 통해 모델이 모든 선택지를 동등하게 학습하고, 데이터를 2배로 늘려 문제 해결 능력을 강화할 수 있도록 했다. 하지만 결과적으로 모델의 성능 향상에는 큰 기여를 하지 않았다.

B. 새로운 문제 생성

Test data는 Train data와 달리 동일한 paragraph에 대해 여러 문제가 존재했기 때문에 Train data에 대한 paragraph에 대해 새로운 문제를을 생성하고 이를 통해 데이터셋을 확장했다.

먼저 Gemini API를 활용하여 새로운 질문과 선택지를 생성하였다. 각 데이터 행에 대해 주어진 단락과 기존의 질문, 선택지, 정답을 기반으로 새로운 문제를 생성하였다. 이 과정에서 Chain of Thought(CoT) 접근법을 사용하여 단락의 주요 내용을 분석하고, 기존 질문과는 다른 유형의 질문을 만들었다.

또한 GPT 유료 API(gpt-3.5-turbo)를 사용하여 단락의 내용을 바탕으로 새로운 문제, 5개의 선택지와 그 중 정답을 한국어로 생성하도록 설정하였으며, Few-shot 학습 예시를 통해 모델이 문제 생성 방식을 학습할 수 있도록 했다. 각 데이터 행에 대한 질문 생성을 위해 API를 호출하고, 그 결과를 수동으로 파싱하여 필요한 정보(질문, 선택지, 정답)를 추출하였다.

이러한 방법들을 통해 데이터셋의 다양성을 증가시키고, 모델이 다각적인 문제 해결 능력을 갖출 수 있게 하는 데 기여했다.

C. 외부데이터

마지막으로, KorQuAD와 같은 외부 데이터를 추가적으로 활용하였다. KorQuAD는 한국어 질의응답 데이터셋으로, 본 프로젝트의 데이터셋과 성격이 유사한 점을 활용하였다. 이를 통해 데이터셋의 양과 질을 동시에 향상시키고, 모델이 다양한 주제와 문장 유형에 대한 이해를 증진하도록 하였다. 이러한 외부 데이터의 사용은 모델의 전반적인 성능을 높이고, 새로운 질문에 대한 대응 능력을 강화하는 데 도움을 주었다.

4.2 모델 탐색 및 경량화

4.2.1 모델 탐색

Tesla V100의 32GB 메모리 제약으로 인해 최대 8B 모델까지 로드 가능하다고 판단하였다. 이를 기반으로 upstage/open-ko-llm-leaderboard, open-llm-leaderboard/open_llm_leaderboard, upstage 게시판 등을 참고하여 적합한 모델을 탐색했다. 보다 구체적인 모델 서치는 아래 링크에서 확인할 수 있다. 노션: 모델 탐색

upstage/open-ko-llm-leaderboard를 참고하던 중 blueapple8259/TinyKo-v5-a 모델을 발견하였다. 해당 모델은 korean_textbooks 데이터셋의 tiny-textbooks를 활용해 학습되었으며, 우리가 목표로 하는 수능형 모델 구축에 적합해 보였다. 그러나 실험 결과, 할루시네이션이 매우 심해 성능이 기대에 미치지 못했다.

이후 upstage 게시판을 통해 Qwen 모델을 알게 되었으나, 초기에는 grad_norm이 지속적으로 Nan 값이 발생하며 제대로 동작하지 않았다. 문제를 해결한 뒤 Qwen/Qwen2.5-3B를 사용한 결과, 성능이 0.5576로 크게 상승하였다. 이를 기반으로 경량화 작업을 진행한 끝에 Qwen/Qwen2.5-7B-Instruct 모델로 최종 선정하였다.

이후에도 팀원들과 함께 LLaMA 계열 모델 및 더 큰 규모의 모델들을 탐색하고 실험에 사용하였으나, Qwen/Qwen2.5-7B-Instruct만큼의 성능 향상을 얻지는 못했다. 따라서 해당 모델을 픽스하고, 다른 방법론을 적용해 추가적인 성능 개선을 시도하기로 결정하였다.

4.2.2 경량화 : 양자화(Quantization)

양자화는 신경망의 가중치와 활성화 함수 출력을 더 작은 비트 수로 표현하여 모델의 메모리 사용량과 계산량을 줄이는 기술이다. 이번 프로젝트에서는 **Qwen/Qwen2.5-7B-Instruct** 모델의 메모리 효율성을 개선하기 위해 **4비트 양자화**를 적용했다.

양자화 과정에서 **NF4 양자화 방식**과 **bfloat16 데이터 타입**을 함께 사용하여, 정확도와 메모리 효율성 간의 균형을 맞추고자 하였다. 이를 통해 모델의 메모리 요구량을 대폭 줄이면서도 양자화로 인한 성능 저하를 최소화하는 데 성공했다.

양자화는 모델의 메모리 효율성과 처리 속도를 높이는 데 효과적인 기술이지만, 일부 정확도 손실이 발생할 수 있다는 점을 알게 되어, 성능 저하를 보완하기 위해 다음과 같은 방법들을 추가로 시도했다.

1. **Gradient Accumulation Steps**: 큰 배치를 작은 단위로 나누어 메모리 효율을 개선
2. **Gradient Checkpointing**: 필요한 활성화(Activation)만 저장하여 메모리 사용량 절약
3. **Low CPU Memory Usage**: low_cpu_mem_usage=True 설정을 통해 대규모 모델 로드 시 CPU 메모리를 최적화

실험 결과, 실행 시간이 단축되고 메모리 효율성 측면에서 개선 효과를 확인할 수 있었으나, 모델 성능은 4비트 양자화만 적용했을 때와 비교해 여전히 기대에 미치지 못했다. Gradient Accumulation과 Checkpointing은 주로 메모리 절약에 초점을 둔 기술이기 때문에 성능 향상에는 한계가 있음을 확인할 수 있었던 실험이었다.

	양자화	Gradient Accumulation & Checkpointing
Leaderboard [mid]	0.6521	0.6382
Leaderboard [final]	0.6000	0.5816

4.3 방법론

4.3.1 SOTA

데이터셋 내 모든 지문의 길이를 분석한 결과, **상대적으로 짧은 지문은 정보량**이 부족하거나 데이터 품질이 낮은 경우가 많았다. 이러한 **데이터를 삭제**함으로써, 모델 학습 과정에서 불필요하거나 부정확한 정보로 인해 발생할 수 있는 성능 저하를 방지하고자 했다. 데이터 필터링 후에는 상대적으로 긴 지문 위주로 구성된 데이터셋이 유지되어 모델 학습에 더 적합한 데이터를 확보할 수 있도록 했다. 특히, 지문의 길이가 길더라도 일부 품질이 낮은 데이터가 존재할 수 있다는 점을 고려하여 선택지에 "정답 없음"을 추가하였다. 이를 통해 데이터의 불완전성을 보완하고, 모델이 잘못된 정답을 학습하는 상황을 방지하고자 했다.

```
problems['choices'].append('정답 없음') # 추가
```

또한, 기존 프롬프트는 "1, 2, 3, 4, 5 중에 하나를 정답으로 고르세요."라는 표현을 사용하였다. 하지만 이는 선택지가 4개 로 구성된 경우 사용자에게 혼란을 줄 가능성이 있었다. 이에 따라 프롬프트를 "위 선택지 번호 중 하나를 정답으로 고르세요."로 변경하여, 보다 유연하고 직관적인 질문 형식으로 변경하였다.

4.3.2 사전지식 O / 사전지식 X 분류

A. 사전지식 필요 여부

사전지식의 필요 여부에 관해서, 우선 훈련 데이터 셋을 KMMLU, MMMLU 등 원본 데이터 셋을 사용하여 주제 별로 분류를 진행했다. 또한 LLM 을 사용한 분류 작업을 실행하여, 사전지식을 필요한 부류와 필요하지 않은 부분을 나누었다. 크게 KLUE-MRC 지문은 기계독해, 즉 MRC 와 관련된 부분이었기에, 독해라고 판단하여 사전지식이 필요없다고 생각했다. 이외의 나머지 훈련 데이터 셋은 사전지식이 필요하다고 판단했다.

B. 힌트 제작

문제 풀이의 효율성을 높이기 위해, MRC문제가 아닌 경우 각 문제에 대한 추가적인 도움을 제공하는 힌트를 제작하는 과정을 진행하였다. 이 과정은 Train 데이터와 Test 데이터로 구분하여 진행되었으며, 각 데이터의 특성에 맞는 모델과 프롬프트를 사용하였다.

Train data의 힌트 제작 과정에서는 데이터의 지문과 질문을 입력으로 하여, OpenAI의 GPT-4 기반 모델을 사용해 300자 이내의 간결하고 명확한 힌트를 생성하였다. 각 데이터의 주제별 특성에 맞게 세부적으로 나눈 후, 각각 다른 프롬프트를 사용하여 생성형 AI를 통해 힌트를 제작하였다. 예를 들어, 지리 관련 데이터를 처리할 때는 GPT 모델에 "지리 전문가"라는 역할을 부여하고, 지문과 질문을 바탕으로 중요한 지리적 맥락이나 추가적인 배경 지식을 제공하도록 하였다. 프롬프트에는 지형, 기후, 인구, 문화, 경제 활동, 환경 문제 등과 관련된 내용을 강조하여, 질문에 대한 이해를 돕기 위한 정보를 추가적으로 생성하도록 설정하였다.

Test data의 힌트 제작 과정에서는 유료 API를 사용할 수 없다는 규칙이 있었기 때문에 생성형 AI 모델인 MLP-KTLim의 llama-3-Korean-Bllossom-8B 모델을 활용하였다. 각 문제의 지문, 질문, 선택지를 입력으로 하여 모델에게 적절한 프롬프트를 생성하고, 이를 통해 간결한 200자 이내의 힌트를 생성하였다. 생성된 힌트는 수동으로 파싱하여 유효한 정보를 추출한 후, 데이터셋에 반영하였다.

힌트 제작 과정은 문제 풀이에 필요한 추가적인 배경 지식과 핵심 정보를 학습자에게 제공함으로써,

학습자가 문제를 더욱 쉽게 이해하고 해결할 수 있도록 돕는 역할을 하였다. 이를 통해 문제 풀이의 정확도를 높이고, 학습 과정에서의 도움을 극대화하였다.

paragraph	나는 윤리학이란 도덕 이론에 근거하여 우리가 당면한 실질적인 도덕 문제를 해결하는 것을 목표로 삼아야 한다고 생각한다. 그런데 어떤 사람은 사회에서 통용되고 있는 도덕 현상을 과학적으로 설명하는 것을 윤리학의 목표로 삼아야 한다고 주장한다. 나는 이러한 주장이 ㉠ 고 생각한다.
question	㉠에 들어갈 진술로 가장 적절한 것은?
choices	['도덕적 담론의 논증 구조에 대한 논리적 분석을 강조한다', '도덕 판단의 표준에 대한 체계적인 이론의 정립을 강조한다', '도덕적으로 바람직한 삶의 이상에 대한 규범적 탐구를 간과한다', '도덕적 딜레마 해결을 위해 타 학문과의 학제적 연구를 강조한다', '도덕규범이 형성된 인과 관계에 대한 경험적인 탐구를 간과한다']
hint	윤리학의 목표에 대한 논의에서 도덕적 현상을 과학적으로 설명하는 것과 실질적인 도덕 문제 해결을 중시하는 두 가지 관점이 존재한다. 윤리학이 도덕적 판단의 표준을 정립하거나 도덕적 딜레마를 해결하는 데 중점을 둔다는 주장도 있다. 도덕적 이슈를 분석하고 논리적으로 설명하는 접근법도 윤리학의 중요한 역할로 간주된다. 이러한 다양한 관점들을 이해하고 비교하는 것이 중요하다.

4.3.3 RAG

Train 데이터셋에 대해 사전지식이 필요한 데이터의 경우 문제의 유형별로 RAG를 시도해보기로 하였다.

A. Corpus 데이터셋

먼저 허깅페이스의 maywell/korean_textbooks을 corpus 데이터셋으로 정했다. 이는 Gemini-1.5 flash를 사용하여 생성된 교과서 품질 데이터셋으로 과목별로 데이터셋이 나뉘어져있다. 따라서 문제의 유형별로 적합한 문서를 찾아올 수 있을 것이라 생각하였다.

B. Retriever 단계

아래 세가지 방법으로 query를 주어 corpus 데이터셋에서 관련된 문서를 추출하고자 하였다.

1. Paragraph
2. Questions
3. Paragraph + Questions

C. Generation 및 결과

Retriever 단계에서도 제대로 관련된 문서를 가져오지 못하여 결과도 좋지 못한 모습을 보여주었다. 이는 corpus 데이터셋도 AI로 생성된 데이터셋이기 때문에 수능 문제의 특성과 복잡성을 정확히 반영하지 못할 수 있다고 판단하였다. 또한 수능 문제는 교과서 내용을 참고하되, 이를 응용하거나 새로운 문맥에서 출제되는 경우가 많아 관련된 문서를 찾기 어려울 수 있다고 판단하였다. 따라서 corpus 데이터셋을 바꾸거나, retriever 성능을 올리는 것에 집중하였다면, 좋은 성능을 가져올 수 있을 것이라 생각하였다.

4.4 결과

마지막으로 시행했던 사전지식이 필요한 모델과, 사전지식이 불필요한 모델의 경우, 팀이 스스로 평가하기에 테스트 데이터 셋에 사전지식을 부착하는데 문제가 있다고 판단했다. 또한 테스트 데이터 셋과, 훈련 데이터 셋의 양식이 조금은 상이한 부분이 존재하여, 훈련 데이터 셋의 검증 정확도와, 테스트 데이터 셋의 검증 정확도의 차이가 존재했다. 이러한 결과로 미루어 보아, 훈련 데이터를 한번에 학습시킨 모델을 사용하게 되었다. 마지막으로 제출 전, 사용했던 방법론중 효과가 있었던 방법들을 모두 앙상블 하여 사용했다.

5 자체 평가 의견

5.1 잘한 점

- 리더보드 상에서 하위권에 있었지만 최종 결과에서 2위한 팀과 비교해보았을 때, 같은 모델에서는 더 좋은 성능을 보이며 방법론적인 측면에서는 더 나은 결과를 가져온 것에 대해 뿌듯했다.
- 지난 프로젝트보다 각 팀원들의 역할 분배가 뚜렷하게 되어 각자의 파트를 성실히 수행했다.

5.2 실험에서 실패했으나 유의미한 실험

- **사전 지식 필요성 유무에 따른 유형별 모델 학습:**
- **RAG:** 적절한 corpus 데이터셋 선정과 retriever 방법에 대해 집중적으로 파악했다면 최종 성능 개선에 기여했을 것이라 판단한다.
- **Unsloth:** 모델 경량화 진행 시에, unsloth라는 방식을 조금 늦게 발견했다. unsloth을 도입해보지 못한 게 아쉬웠고, 다음부터는 좀 더 조사를 충실히 하면 좋을 것 같다.

5.3 프로젝트 협업 프로세스에서 아쉬웠던 점

- 초반에 노션정리를 열심히 했는데 후반으로 진행될수록 코드가 고도화돼어서 공유할때 아쉬웠다.
- 협업 tool로 Notion을 사용했으나 전반적인 진행 과정을 한 눈에 파악하기가 어려웠다. 발표한 팀들처럼 Jira, Confluence 등의 tool을 사용해보면 좋을 것 같다.

5.4 프로젝트를 통해 배운점 또는 시사점

- 첫번째 잘한 점에서 다른 조들은 모두 우리 조보다 더 큰 모델을 사용했다는 것을 알았는데, 이를 통해 제한된 GPU 서버 자원내에서 큰 모델 적용에 더 신경 썼다면 훨씬 좋은 성능을 낼 수 있었을 것이라 생각이 든다. 이를 통해 다음 프로젝트에서는 리더보드 상에서도 좋은 결과를 낼 수 있으리라 생각한다.

boostcamp ai tech

부스트캠프 AI Tech

개인 회고

boostcamp ai tech

권지수 캠퍼

이번 프로젝트에서의 역할

이번 Generation for NLP 프로젝트에서, 본인은 데이터를 사전지식이 필요한지 여부에 따라서 분류하고, 노이즈가 있는 텍스트를 정제하고, 적합한 모델을 탐색하고, 이를 경량화하고, 프롬프트를 바꾸면서 더 좋은 성능을 낼 수 있도록 실험하는 역할을 수행했다.

사전지식이 필요한 데이터들은 대부분 paragraph의 길이가 짧거나, problems 안에 <보기>가 주어지는 형식이었다. 또한 사전지식이 필요없는 데이터들은 대부분 paragraph 안에 선지의 내용이 그대로 들어가 있었다. 따라서 1. problems 안에 <보기>가 있는지, 2. paragraph 안에 선지의 내용이 있는지, 3. paragraph의 길이가 50자 이하인지를 기준으로 데이터를 분류했다. 이를 통해서 ODQA task인 데이터들과, 사전지식이 추가로 필요한 데이터를 대부분 잘 분류해낼 수 있었다.

그리고 주어진 baseline 코드를 통해 여러 가지 모델들을 탐색하고 실험하며, 우리의 이번 task에 적합한 모델을 탐색했다. 이번 task에서는 수능 데이터로 fine-tuning 된 모델을 쓰는 것이 금지되었었기 때문에, 그렇지 않은 모델 중 성능이 가장 잘 나오는 모델을 김성은 캠퍼와 함께 탐색하였다.

또한 baseline 코드는 3B 이하 크기만 올릴 수 있었기 때문에, 이보다 조금 더 큰 7B까지 올리는 것을 목표로 경량화 방법을 탐색했다. 결과적으로 train.py 안에서 모델을 로드할 때 FP16을 설정하고, load_in_4bit=True을 설정함으로써 7B 뿐만 아니라 13B 모델까지 올리는 것에 성공했다.

지난 프로젝트보다 더 발전한 점

지난 프로젝트보다 협업 측면에서는 훨씬 더 발전했던 것 같다. 나 뿐만 아니라 많은 팀원들이 git과 notion에 익숙해지면서 서로의 진행상황을 좀 더 한눈에 빠르게 확인할 수 있었던 것이 좋은 발전이었다.

또한 지난 프로젝트보다 다양한 시도를 해봤다. 프로젝트의 목표가 '작은 모델로' GPT 등의 거대 언어 모델을 뛰어넘어보는 것이었기 때문에, 7B 모델을 기반으로 다양한 방법으로의 데이터 증강, 경량화, RAG 등의 실험을 도전했다. 이번 프로젝트에서는 비록 좋은 결과를 내지는 못했지만, 향후 많은 프로젝트에서 쓰일 기술들을 미리 도전하고 실험해봤다는 측면에서 이번 프로젝트가 큰 의미가 있었다고 생각한다.

아쉬웠던 점과 더 나은 개발자가 되기 위한 회고

확실히 아쉬운 점은 있다. 우리 팀은 늘 상승세를 타서 저번 프로젝트는 3위까지 차지를 했는데, 이번 프로젝트에서는 계속 하위권에 머물렀다. 1, 2위 팀의 발표를 들어보니까 우리보다 큰 32B 모델까지 올려서 성능이 더 잘 나왔던 것 같다. 다만 모든 팀에게 주어진 데이터와 서버, 시간은 동일하고, '큰 모델'이라고 정의하는 것 또한 상대적이다. 그래서 나도 모델 경량화를 맡아서 진행한 사람으로써, unsloth 등 조금 더 좋은 경량화 방법론을 더 찾아냈더라면 하는 아쉬움이 있다.

또한, 데이터 분류 등에서 LLM에 너무 의존을 하지 않았나 싶다. 성적이 좋은 팀들의 발표를 들어보니, LLM으로 사전지식이 필요한 데이터 분류는 많이 하지 않았던 것으로 보인다. 그러나 우리 팀은 프롬프트를 조금만 바꿔도 성능이 훅훅 변하는 LLM을 보면서 프롬프트 엔지니어링에 많은 시간을 쏟았다. 차라리 그 시간에 어떻게 하면 데이터를 좀 더 정확하게 분류할 지를 고민하는 것이 나았을 것 같다.

앞으로 진행할 프로젝트에서는 내가 맡은 역할에서 찾아낸 방법론에서 그치지 않고, 프로젝트가 끝날 때까지 계속 탐구하고 방법을 적용해보는 자세를 가져야겠다. 또한, LLM에 대한 의존성을 줄이고 우리가 직접 탐구하고 진행하는 방식으로 나아가야겠다.

boostcamp ai tech

김성은 캠퍼

데이터 이해 및 준비

- 대회에서 제공된 KMMLU, MMMLU 데이터셋과 MRC 데이터셋의 형식과 구성을 분석한 결과, 데이터 원본에 따라 4지선다형 문제와 5지선다형 문제가 혼합되어 있다는 점을 파악했다.
- 다른 팀의 발표를 통해 약 2000여 개의 데이터를 구글 스프레드시트로 하나씩 검토하며 분리했다는 이야기를 듣고, 데이터 분석과 정리에 있어 좀 더 체계적으로 접근하지 못한 점이 아쉬웠다.

데이터 증강

1. **가오카오 데이터셋 활용 시도**
   - Deepl API를 활용하여 '중국어 -> 영어 -> 한국어'로 번역하는 방식을 시도했으나, 번역 품질이 온전하지 않아 중단했다.
   - 마스터클래스에서 다른 팀이 Deepl API 키를 무한 생성하며 데이터를 성공적으로 증강했다는 소식을 들으며 좀 더 적극적으로 도전하지 못한 점이 아쉬웠다.
   - 번역 품질을 개선할 방법을 연구하거나, 데이터 증강의 대안을 체계적으로 탐구하는 것이 중요하다는 점을 깨달았다.

2. **maywell/ko_wikidata_QA**
   - 데이터 생성 과정에서 디버깅 부족으로 어려움을 겪어, 데이터의 활용 가능성을 충분히 검증하지 못했다.

모델 탐색

Qwen 7B instruct 모델 활용

- 적절한 모델을 찾은 점은 좋았지만, Unsloth 등의 대안을 떠올리지 못한 점이 아쉬웠다.

프롬프트 엔지니어링

1. **사전지식 유무에 따른 프롬프트 설정**
   - 사전지식이 필요하지 않은 경우에는 "국어를 잘하는 똑똑한 학생"이라는 역할을 부여한 프롬프트를 통해 성능을 0.77까지 끌어올릴 수 있었다.
   - 하지만 사전지식이 필요한 경우, 성능 향상이 어려워 프롬프트 설정의 한계를 경험했다.

2. **AI 수능 국어 만점 프로젝트 참고**
   - "AI 자극 프롬프트", "감정적 호소 프롬프트" 등 다양한 방식의 프롬프트를 실험했으나, 가장 간결한 프롬프트가 성능 면에서 가장 우수하다는 결론을 얻었다.
   - 프롬프트 설계에서 복잡한 시도보다, 간결하고 핵심적인 접근이 더 효과적일 수 있다는 점을 확인했다.

협업

- 이번에 프로젝트 시작 전에 노션을 새롭게 정비하며 팀원 간 협업을 강화하려 했으나, 체계적인 실행과 준비가 부족했던 점이 아쉽다.
- 다른 팀의 발표를 통해 자극을 받았으며, 앞으로는 더 능동적이고 조직적인 태도로 프로젝트를 진행해야겠다는 다짐을 하게 되었다.
- 더 나은 성과를 위해 시간 관리, 데이터 활용 능력, 디버깅 스킬, 협업 역량을 전반적으로 강화할 필요성을 느꼈다.

boostcamp ai tech

👨‍🏫 김태원 캠퍼

팀내에서 나의 역할

Project Manager 의 역할을 담당했다. 먼저 Project Manager 로서 팀이 하나의 방향성으로 진행하기 위해서 실험해보고 싶은것, 또는 실험 해야할 것에 대한 분석을 하여 팀원들에게 배포했고, 노션 템플릿을 골라 페이지를 만들어, 분업할 수 있는 환경을 조성했다.

AI 개발자로 성장을 하기위해 시도했던 모험들(어떤 것을 도전했는가? 어떤 것들을 익혔는가?)

우선 첫 주차가 지나고 나서, 팀원들과 내가 많은 것을 놓치고 있다는 것을 깨달았다. 데이터의 수를 증강을 사용해서 늘린다고 해서 사용되는 데이터의 수가 급격하게 늘어나지 않는 다는 것을 깨달았다. 그래서 생각을 달리 하여, 본인이 직접 수능을 푼다고 생각해봤다. 이때 우선 국어와 사회 문제가 혼재된 문제 셋을 푼다고 가정 했을 때, 가장 먼저 해야할 일은 어떤 것이 국어 문제이고, 어떤 것이 사회문제인지 파악해서 스스로 labeling 하는 것이다.

이후, 국어 문제의 경우 독해 문제이면, 독해할 때 지문 안에 답이 있는 것을 파악하고, 사회 문제인 경우, 사전지식이 필요함을 알고, 스스로 사전 지식을 떠올리는 작업이 필요하다. 이러한 생각의 흐름, CoT 를 LLM 에게 주입 시키기 위해서 도전했었다.

아쉽게도, test dataset 이 train dataset 의 양식들과는 사뭇 달라서, 이런 식으로 모델을 분리하여 학습 시켰을 때 좋은 효과를 보진 못했지만, 뜻깊은 도전이었던 것 같다.

부딪힌 한계, 개선해야할 부분

우선, 데이터셋 형식 차이가 큰 문제였다. 테스트 데이터와 훈련 데이터의 양식이 달라서 모델을 분리하여 학습시킬 때 성과가 예상만큼 좋지 않았다. 이 부분을 개선하려면, 데이터 전처리와 파라미터 튜닝을 좀 더 세밀하게 조정하고, 테스트와 훈련 데이터 간의 일관성을 더 신경 써야겠다는 생각을 했다.

또한, LLM 의 미숙한 사용이 발목을 잡았다. 문제 유형을 구분하거나, 필요한 사전 지식을 떠올리는 과정에서 모델이 종종 논리적 오류를 범하거나 잘못된 답을 내놓는 경우가 있었다. 모델이 맥락을 충분히 이해하지 못하고 훈련 데이터만 따라가다 보니 이런 실수가 생겼던 것 같다. 이를 해결하기 위해서는 모델의 피드백 루프를 더 잘 다듬고, 다양한 시나리오를 통해 학습시켜서 더 정확한 결과를 낼 수 있도록 해야겠다는 교훈을 얻었다.

그리고 노션 사용에 있어서도 미숙함을 느꼈다. 처음엔 작업 관리나 팀원들과의 협업을 효율적으로 하는 데 생각보다 잘 사용했었다. 하지만 프로젝트 후반부로 진행 할 수록, 실험해야 하는 경우의 수가 많아지고, 또 코드가 계속 바뀌다 보니, 노션의 다양한 기능을 제대로 활용하지 못해서 생산성이 떨어지고, 팀원들 간의 정보 흐름에 차질이 생기기도 했다.

이러한 점을 해결하기 위해서는, 맨 처음부터 코드의 모듈화를 정확히 시행하고, 모두 같은 양식으로 코드를 짜고 코드를 수정해야 함을 깨달았다. 나중에는 노션을 잘 활용할 수 있도록 더 공부하고, 팀원들과의 소통을 원활하게 하기 위한 방법을 찾아야겠다고 다짐했다.

boostcamp ai tech

이다현 캠퍼

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

점수에 상관없이 생각해본 방법들을 모두 시도하려고 노력했다. 비록 대부분의 시도가 성공적이진 않았지만, 시도해보고 싶었던 것들은 모두 진행했다. 이를 통해 단순히 결과보다는 시도 자체의 가치를 느낄 수 있었다.

2. 나는 어떤 방식으로 모델을 개선했는가?

데이터 처리 방법이나 문제 접근 방식에 대해 여러 아이디어를 내고 시도했지만, 그 방법들이 큰 효과를 내지는 못했다. 하지만 다양한 방법론을 실제로 시도해본 것 자체가 의미 있는 경험이라고 생각했다. 시행착오를 통해 여러 가능성을 확인할 수 있었다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

내가 한 행동의 결과로 잘못된 접근 방식을 판별할 수 있었다. 이를 통해 내 지식이 아직 많이 부족하다는 점을 깨달았고, 앞으로 더 많은 공부가 필요하다는 것을 느꼈다. 시행착오가 나의 약점을 분명하게 보여준 계기가 되었다.

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

이번에는 깃을 좀 더 활발히 사용하려고 했다. 이전보다 깃에 대해 더 깊이 이해하게 되었고, 팀원들과의 협업에서 조금 더 수월하게 작업할 수 있었다. 하지만 여전히 개선할 부분이 남아 있음을 느꼈다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

PM 역할을 맡았지만 프로젝트가 잘 굴러가는 느낌을 받지 못했다. 내가 제대로 PM 역할을 해내지 못한 것 같아서 아쉬웠다. 그리고 상위권에 있는 사람들이 한 접근법들을 나는 모르고있던 점도 아쉬움으로 남았다.

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

다음 프로젝트에서는 시작 전에 구체적인 계획을 세우고 진행해보려고 했다. 또한 다양한 자료를 적극적으로 찾아보고, 이를 바탕으로 더 나은 방법론을 시도해볼 계획이다. 다음엔 부족함 없이 프로젝트를 시도해볼 것이다.

boostcamp ai tech

👨‍🏫 이한서 캠퍼 (Model Searching, Data Augmentation, Prompt Engineering, Fine-tuning)

총평

과정과 결과가 매우 아쉬웠지만, 배운 점이 가장 많았던 프로젝트였다. 문제에 대한 정의, 즉 사전 지식이 필요한 문제('국어 문제')와 그렇지 않은 문제('사회 문제')로 나누고, 사회 문제에 대해서는 RAG를 적용하자는 생각은 정말 좋았다고 생각한다. 그러나 프로젝트 초반에 몸이 아파 휴가를 써서 팀원들을 따라가는데 급급했고, 이후 다양한 시도를 해보았으나 성능이 전혀 개선되지 않았고, 다른 팀원들 역시 드라마틱한 성능 향상을 이끌어내지 못했다. 마지막 주차에는 번아웃이 와서 이전 프로젝트들처럼 몰두하지 못했던 것 같아 아쉬웠다. 지난 프로젝트까지 리더보드 최종 순위가 9위→7위→3위로 상승해 이번에는 1등할 수 있겠다고 생각했는데, 꼴찌에 가까운 순위로 마쳤다.

프로젝트를 마친 뒤 '리더보드 상위권 팀들은 32B 모델을 돌렸다'는 이야기를 듣고, 처음에는 소형 모델로 대형 모델을 능가하는 것이 이 프로젝트의 목표인데, 이는 프로젝트의 목적에 맞지 않는다고 생각했다. 그러나 다른 팀들의 발표와 마스터님의 코멘트를 듣고 그 생각이 바뀌었는데, 우선 32도 대형 모델들에 비하면 작은 편이고, 32B 모델을 경량화하여 V100 GPU에 올리는 것도 굉장히 어렵기 때문이다. 실제로 발표한 리더보드 1위 '나야 자 연어'팀도 Unsloth 설치를 위해 서버를 9번 터뜨렸고, 10번만에 성공했다. 포기하지 않고 성공할 때까지 해낸 의지가 정말 멋졌다. 그 외에도 다양한 출처의 데이터로 증강하거나 구현하기 어려운 RAG도 시도해보고, 결과가 좋게 나오지 않아도 좌절하기보단 그 원인을 고민해보는 등 본받을 점이 너무나 많은 발표를 들을 수 있어 기뻤고, 이 팀을 본받아서 후회가 남지 않는 최종 프로젝트를 해보겠다.

실패했지만 의미있는 시도

1. **데이터셋 정제 및 증강**
   Train Dataset에 노이즈가 많아서 정제를 해보고자 하였다. 이에 Streamlit으로 Tool을 만들어 직접 정제를 시도하였는데, 양이 방대하고 Test Dataset에도 노이즈가 많아 투자한 시간에 걸맞는 성능 향상이 이루어질 지 확신이 가지 않아 중단했다. 또한 KorQuAD를 Train Dataset의 형식으로 수정해 데이터를 증강해보았는데, 오히려 성능이 감소하였다. 막연히 한국어 질의응답 데이터셋이라서 증강하였는데, 수능형 문제와 많이 달라서 그런 것 같다. 1위 팀의 발표를 보면 8개의 데이터를 정제하니 0.018의 성능 향상이 이루어지던데, 역시나 데이터의 양보다는 질이 훨씬 더 중요하다는 것을 뼈저리게 느꼈다.
2. **힌트 추가**
   사전 지식이 필요한 문제들을 풀기 위해 힌트를 추가하고자 했다. 이에 LLM을 이용한 지문 요약, 질문 분석, 지문 해설, 어휘 해설, 혹은 사전 지식이 필요하다는 정보 등을 힌트로 주었는데 성능 향상으로 이어지지는 않았다. LLM을 이용한 방식들은 직접 사전 지식을 제공해주는 방법은 아니므로, RAG를 통해 사전 지식을 채우는 방법이 더 근본적인 해결책이 되었을 것 같다.
3. **프롬프트 수정**
   LLM으로 수능 국어 1등급에 도전하는 NomaDamas/KICE_slayer_AI_Korean Repo를 보고, 해당 레포에서 제시한 형식 지정 프롬프트, zero-shot-CoT 영어 프롬프트, 감정적 호소 등 다양한 프롬프트를 실험해보았다. 놀랍게도 동일한 환경에서 프롬프트만 바꿨을 때 0.1 이상의 성능 차이를 보이기도 했다. 이전까지 프롬프트 엔지니어링이라는 단어가 과하다고 생각했는데, 프롬프트가 데이터 만큼이나 중요하다는 것을 몸소 깨달았다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

데이터 검증 및 정제 과정에서 LLM에 과하게 의존한 것이 아쉽다. 지난 Data-Centric 프로젝트에서 데이터 정제에 Gemini API를 적극적으로 활용하였고, 그 결과가 매우 좋았다. 그러나 박장원 멘토님께서 '너무 외부 API에 의존한 것 같아 아쉽다'는 평가를 남겨주셨는데, 처음에는 왜 결과가 좋은데 외부 API를 사용하면 안되는지 이해가 가지 않았다. 그런데 이번 프로젝트에서 그 말씀이 이해가 갔다. 1위를 한 팀의 발표를 들어보니, 데이터를 정제 시 2,031건의 학습 데이터셋에 대해 수동으로 레이블링했다고 한다. 나도 수동 레이블링 및 정제를 시도하다가 막막해서 그만 두고 또 다시 LLM을 사용했는데, 안락함에 취해 편협하고 성능 개선이 불확실한 방법을 선택한 점이 아쉽다. 번거롭고 오래 걸리지만 끝까지 해내어 성능 향상으로 이끈 점이 존경스러웠다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

데이터를 분석하고 정제할 때, 몸이 편한 방법보다는 힘들지만 더 근본적이고 정확한 방법으로 접근해야겠다. 데이터를 증강하거나 모델을 선택하고 튜닝할 때, 최대한 많은 실험을 하고, 그 과정과 결과를 보기 좋게 기록해 팀원들에게 공유하여 한 마음 한 뜻으로 움직여야겠다. 실험 후 성능 개선에 실패하더라도 거기서 멈추지 말고 실패한 원인을 고민해보고, 이를 멘토님께 질문하여 지식으로 저장해야겠다. 마지막으로 절망스러운 상황이 생기더라도 '나야 자연어'팀처럼 포기하지 않고 성공할 때까지 시도해야겠다.

boostcamp ai tech

👨‍🏫 정주현 캠퍼

베이스라인 모듈화

기존 베이스라인 **ipynb 파일**을 토대로 팀원들이 보다 쉽게 실험할 수 있도록 코드를 모듈화하였다.

- **중앙 관리 체계 구현**:
  실험에 사용되는 데이터, 주요 설정 값, 출력 파일 경로 등을 중앙에서 관리할 수 있도록 **config.json 파일**을 작성하였다. 이를 통해 실험 관리와 설정 변경이 용이해졌다.
- **코드 구조 개선**:
  학습과 추론 과정을 분리하여 **Train.py**, **Test.py** 파일로 나누었으며, 데이터 처리를 위한 **DataProcessing.py** 파일과 유틸리티 기능을 위한 **Utils.py** 파일을 별도로 구성하였다. 이를 통해 다양한 방법론을 적용하여 실험할 수 있도록 구현하였다.
- **프롬프트 관리 개선**:
  프롬프트 엔지니어링을 효율적으로 관리하기 위해 **Prompt 파일**을 따로 분리하여, 실험마다 필요한 프롬프트를 체계적으로 다룰 수 있도록 하였다.

이러한 구조적 개선을 통해 팀원들이 모듈화된 코드를 활용하여 실험을 진행할 수 있었으며, 다양한 실험을 보다 효율적이고 체계적으로 관리할 수 있었다.

데이터 EDA

이번 Generation 프로젝트에서는 데이터를 면밀히 분석하며, **Paragraph**와 **Question**의 길이 분포를 중점적으로 살펴보았다. 분석 결과, 지문의 길이가 짧을수록 질문에 대한 정보량이 부족하거나 데이터 품질이 낮다는 점을 확인했다. 데이터 품질이 낮다는 것은 질문에 대한 답변이 부정확하거나, 모델이 학습 과정에서 잘못된 정보를 학습하게 되는 경우를 의미한다. 이를 바탕으로, 지문의 길이에 대한 통계 분석을 실시하였으며, **최소값에서 제1사분위수**에 해당하는 해당 데이터에서 **중앙값**에 해당하는 지문까지의 데이터를 검토했다. 검토 과정에서 해당 데이터들이 대체로 품질이 떨어짐을 확인한 후, 해당 데이터를 삭제하기로 결정하였다. 이러한 조치를 통해 데이터의 신뢰성을 높이고, 모델 학습의 정확도를 개선하고자 하였다.

데이터 처리에 기반한 프롬프트 엔지니어링

이번 프로젝트의 훈련 데이터는 4개의 선택지 또는 5개의 선택지를 포함한 객관식 데이터로 구성되어 있었다. 그러나 베이스라인 프롬프트가 "1, 2, 3, 4, 5 중 하나를 정답으로 고르세요."로 설정되어 있어, 모델에 혼동을 줄 가능성이 있다고 판단하였다. 이에 따라 프롬프트 수정을 검토하였으며, CoT, Few-shot 등 다양한 방법을 시도한 결과, **간결하고 명확한 프롬프트**가 가장 효과적임을 확인했다. 최종적으로, 모델의 혼동을 줄이고 명확성을 높이기 위해 프롬프트를 "위 선택지 번호 중 하나를 정답으로 고르세요."로 수정하여 모델 성능 개선에 효과를 보았다.

실패했지만 의미 있는 시도

- **RAG**

  SOTA의 성능을 개선하기 위해, **train**과 **test 데이터셋**을 사전지식이 필요한 데이터와 그렇지 않은 데이터로 나누어 학습과 추론을 각각 진행하였다. 특히, 사전지식이 필요한 데이터에 대해 RAG (Retrieval-Augmented Generation)를 적용하면, 모델이 지문에 부족한 정보를 보완하여 문제를 풀 수 있을 것이라 생각하여 성능 향상을 기대했다. 이를 위해, 수능 문제 풀이에 적합한 **corpus 데이터셋**을 조사하였고, 교과서 품질 데이터를 포함하며 과목별로 나뉘어 있는 허깅페이스의 **maywell/korean_textbooks** 데이터셋을 선정하였다. 다음으로, **train 데이터셋**을 Gemini를 이용해 각 문제의 토픽(과목)을 분류한 뒤, 해당 과목에 맞는 corpus 데이터셋에서 **BM25**를 활용해 관련 문서를 검색하도록 구현하였다. 이때, **paragraph + question**을 query로 사용하였는데, 그 이유는 다음과 같다:

  - **Paragraph 단독**으로 query를 사용할 경우, question이 요구하는 정보를 반영할 수 없어 적합한 문서를 추출하기 어려울 것이라 생각하였다.
  - **Question 단독**으로 query를 사용할 경우, "위 지문에서 옳은 것을 고르시오"와 같은 질문에서, 지문 정보 없이 문제와 관련된 문서를 추출하기 어려울 것이라 판단하였다.

  그러나 retriever 실험 결과, 적절한 문서를 가져오지 못하는 한계가 드러났다. 원인으로는 다음과 같은 점이 있다고 판단하였다:

  1. **Corpus 데이터셋의 품질**: 데이터가 AI에 의해 생성된 것으로 문서가 자연스럽지 않을 가능성이 있음.
  2. **수능 문제의 특성**: 교과서 내용을 참고하되 새로운 문맥에서 출제되는 경우가 많아 관련 문서를 찾기 어려움.
  3. **Query의 길이 문제**: 대부분의 query가 500자를 초과하여, 중요한 정보를 retriever가 효과적으로 판단하지 못할 수 있음.

  이러한 한계를 보완하기 위해, 더 적합한 **corpus 데이터셋**을 선정하고, **Dense Retriever**와 BM25를 결합한 **Hybrid Retriever**를 사용했더라면 성능 개선이 가능했을 것이라는 아쉬움이 남았다.

- **프롬프트 엔지니어링**

문제 풀이 모델의 프롬프트에 **CoT (Chain-of-Thought)** 방식을 적용하여 모델이 보다 논리적으로 문제를 풀 수 있도록 유도하였고, 이를 통해 성능 향상을 기대하였다. 그러나 성능이 향상되지 않았으며, 이는 사용한 모델이 **7B 파라미터**로, 거대 언어 모델에 비해 상대적으로 작기 때문에 프롬프트가 길어질수록 복잡한 요구사항을 제대로 처리하지 못한 것이 원인이라고 판단하였다. 프롬프트에는 지문, 질문, 선택지가 함께 포함되는데, 여기에 CoT를 적용하면 이미 길다고 느껴지는 프롬프트가 더욱 길어져 결과적으로, **간결하고 명확한 프롬프트 지시**가 오히려 더 좋은 성능을 낼 수 있었다고 판단하였다.

협업

이번 프로젝트에서는 전 프로젝트에 비해 협업이 굉장히 잘 진행되었다. 깃 사용이 전 프로젝트에선 거의 되지 않았지만 이번 프로젝트에서는 팀원들의 여러가지 방법론을 깃에 올려 코드를 공유할 수 있었다. 또한 노션을 이용하여 실험 결과와 코드 이외의 아이디어적인 측면을 적어 팀원들과 공유하였다. 초반 모델과 방법론 최적화에 큰 역할을 하였으며 팀원들이 어떤 모델로 어떤 방법론을 적용하여 실험했는지 확인할 수 있었다. 하지만 대회 후반으로 갈수록 노션 활용도가 떨어지는 느낌을 받아 아쉬운 느낌도 있었다. 개인적으로도 후반으로 갈수록 노션을 잘 보게되지 않아 팀원들의 방법론을 줌 회의에서 알게 되는 경우도 있어 나의 부족함을 느꼈다. 따라서 프로젝트 때마다 성실히 팀원들의 노션을 성실히 체크해야겠다고 마음을 다시 잡았다. 하지만 전반적으로 우리 팀에서 협업은 대체로 잘 진행되는 느낌이었다. 서로 잘 모르는 부분에 대해 개인적인 연락도 원할히 하였고, 전체가 모델 성능의 개선을 위해 한 방향으로 나아갔다.