Generation for NLP Wrap-Up Report

프로젝트 회고

1. 프로젝트 개요

이 프로젝트는 LLM을 활용하여 AI 모델이 어떻게 하면 얼마나 다양한 분야의 시험에서 우수한 성적을 받을 수 있을 지에 대해 연구해보고, 한국어와 수능형 문제에 특화된 모델을 개발하여 소규모 모델로 대형 모델 수준의 성능을 달성하는 것을 목표로 하는 프로젝트이다.

학습 데이터의 경우에는 KMMLU / MMMLU(Ko) / KLUE MRC 데이터를 합해 총 2031개의 데이터를 기반으로 학습을 진행하였고, 이를 기반으로 학습한 LLM 모델을 평가하는 데이터는 수능형 문제 + KMMLU / MMMLU(Ko) / KLUE MRC로 총 869개의 데이터로 평가를 진행했다.

서버 환경

- NVIDIA Tesla V100 GPU Server (* 4EA) - VRAM 32GB

평가 방법

- 주어진 선택지 중 정답 예측
- 지정된 submission 형식에 따른 csv 파일 제출 및 평가
- Metric: Accuracy

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

데이터 구성

- 수능형 문제
  - 수능의 국어, 사회 영역(윤리, 정치, 사회)과 비슷한 문제
- KMMLU (Korean History), MMMLU (HighSchool 데이터 중 역사, 경제, 정치, 지리, 심리)
- KLUE MRC (경제, 교육산업, 국제, 부동산, 사회, 생활, 책마을)

Column	설명	예시
id	데이터를 구분하기 위한 식별자	generation-for-nlp-2708
paragraph	문제의 지문	*(전략)* … 시장조사회사 딜로직에 따르면 지난 19일 페이스북이 모바일 메신저서비스 왓츠앱을 190억달러에 인수한다고 발표하면서 올 들어 현재까지 … *(후략)*
problems	질문, 선택지, 정답이 포함된 json	{”question”: “…”, “choices” : […], “answer”: “…”}
- question	풀어야 하는 질문	페이스북이 190억 달러에 인수한 모바일 메신저 서비스의 이름은 무엇인가?
- choices	질문의 선택지 (4개 혹은 5개)	['인스타그램', '왓츠앱', '스냅챗', '네스트랩', '보스턴 다이내믹스']
- answer	문제의 정답. 1, 2, 3, 4, 5 중 하나. *(Test 데이터셋에는 주어지지 않는다.)*	2
question_plus	지문 외에 질문을 위해 추가적인 정보를 담은 보기 (Optional)	*(null)*

분류	샘플 수	비고
Train	2,031	모든 정보가 공개되어 있으나 question_plus를 활용하는 데이터는 없음
Test	869	이 중 434개 데이터는 public, 435개 데이터는 private 점수 계산에 사용

2. 프로젝트 팀 구성 및 역할

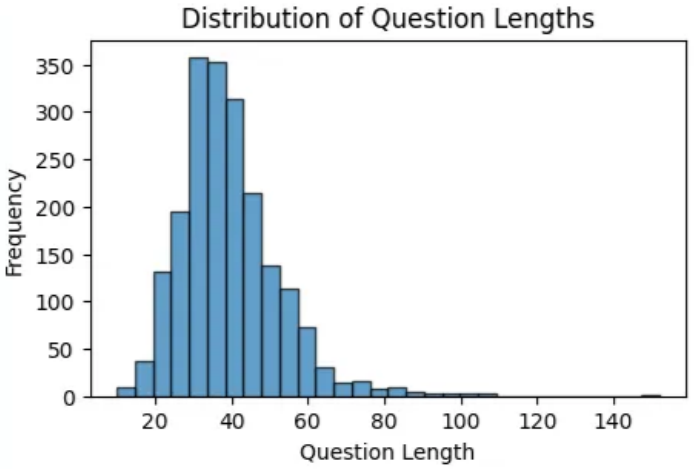
팀원	역할
강전휘	Generation 기반 CoT, prompt engineering
박상준	Model, KFold Cross Validation, Hyper-Parameter Tuning
박준성	OpenAI API를 사용한 Data Enhancing, CoT 실험 진행, 베이스라인 코드 모듈화, 앙상블
백승우	RAG (Sparse/Dense/Hybrid Retriever)
서태영	EDA, Model
이재백	EDA, Data Augmentation, Data Preprocessing(Label balancing), Zero-shot-CoT 구현

3. 프로젝트 수행 절차

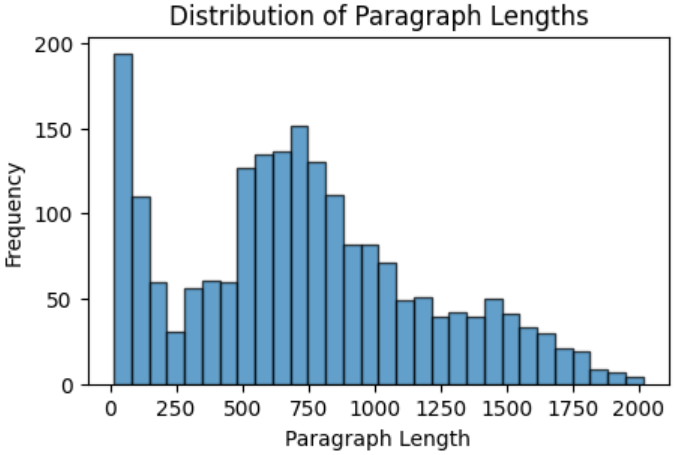
1. 프로젝트 개발 환경 구축(Github, Server, VScode) 및 Baseline code template 생성
2. Data EDA 및 Augmentation 진행
3. RAG와 Generation 파트로 나누어 실험 및 분석
    a. RAG
        i. Retrieve를 위한 Documents 탐색 및 parsing
        ii. Retriever 성능 개선
        iii. LoRA기반 모델에 적용
    b. Generation
        i. Open source LLM 모델 탐색
        ii. CoT를 활용한 generation 기반 결과물 생성
        iii. LoRA를 이용해 dataset의 특성에 맞게 fine-tuning
4. Ensemble을 진행하여 최종 결과물 도출 (Soft Voting 및 Hard Voting)

4. 프로젝트 수행 방법 및 결과

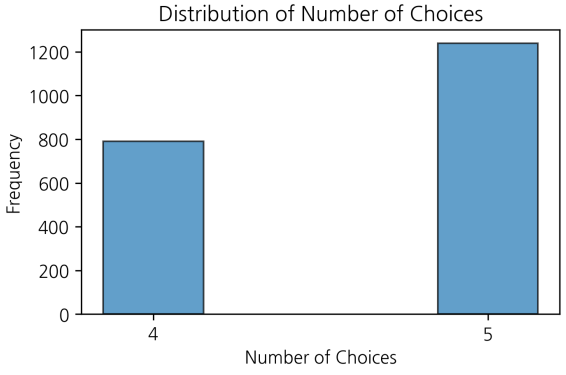
1) EDA



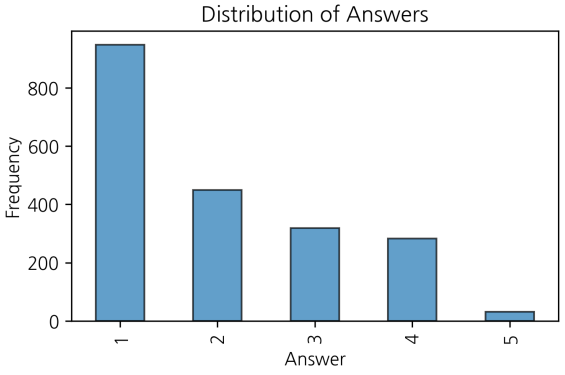
[Figure 1] 문제의 길이가 어느 정도 되는 지 파악할 수 있었다. 30~40의 구간에 가장 많은 분포가 이루어졌으며, 양은 적지만 20 이하의 문제나, 60 이상의 긴 문제도 존재했다.



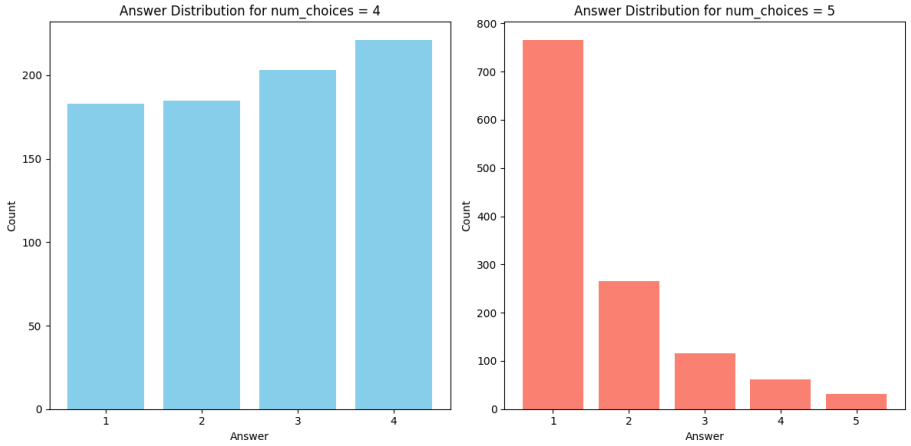
[Figure 2] 전체적인 paragraph의 길이를 보기위해 그래프화 시켜보았다.



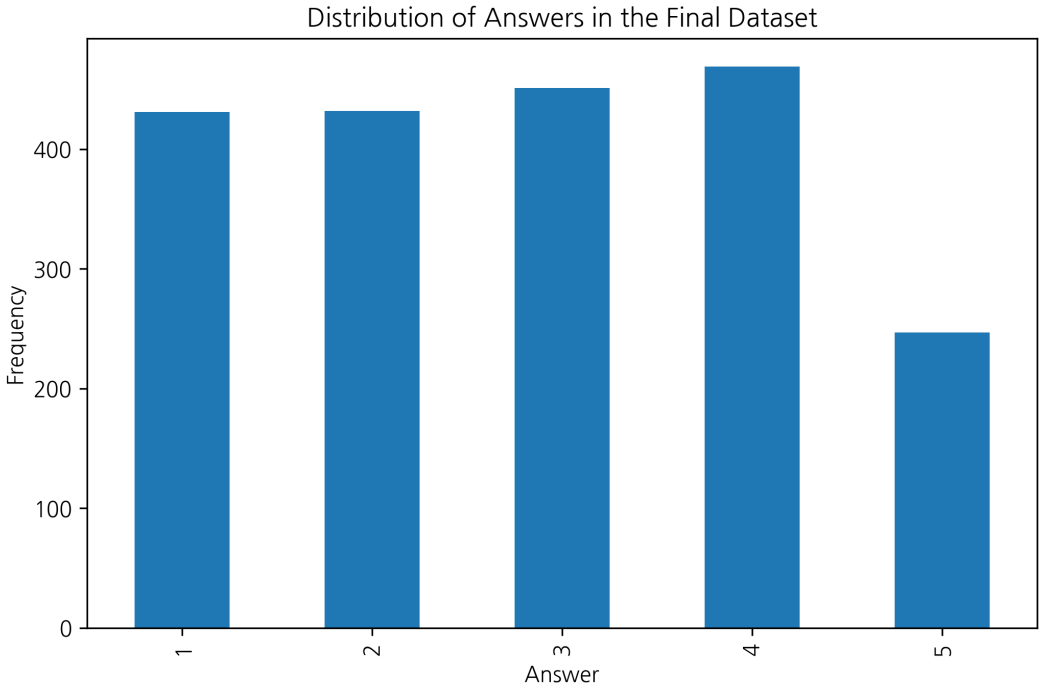
[Figure 3] 사지선다 문제는 791개, 오지선다 문제는 1239개의 분포를 보인다.



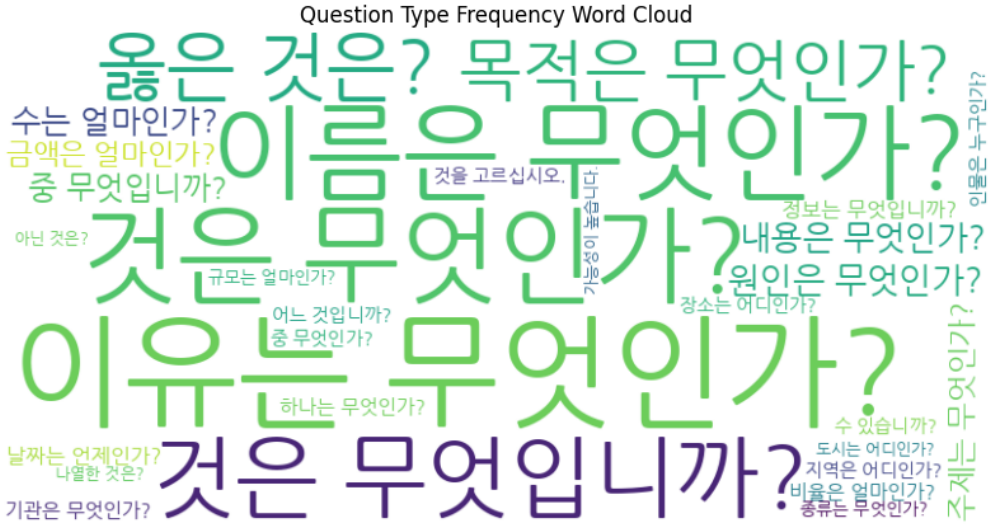
[Figure 4] 정답의 분포는 1번 948개, 2번 449개, 3번 319, 4번 283개, 5번 31개로 편향된 분포를 보였다.



[Figure 5] 정답의 불균형한 분포는 대부분 오지선다 문제에서 비롯한 것으로, 사지선다 문제들에선 균일한 정답의 분포를 확인할 수 있었다.



[Figure 6] 데이터 불균형을 확인하고 전처리를 통해 균형잡힌 라벨 데이터셋을 제작하였다.



[Figure 7] question 컬럼의 마지막 단어 두개의 구성 분포를 통해 문제 유형의 분포를 시각화하였다.

2) Augmentation

- 기존 2031개의 train 데이터셋에서 9031개까지 증강을 하였다. 아래는 증강에 사용한 데이터셋들이다.
  - PSAT 언어논리(2017~2024)
  - CLIcK 데이터셋
  - HAE-RAE 벤치마크 데이터셋
  - RACE 데이터셋 번역본
- public 리더보드에서는 차이가 없었지만 프로젝트가 끝난 후 공개된 private 리더보드에서는 성능 향상을 확인할 수 있었다.

기법	public	private
base_t3q	0.6705	0.6161
base_t3q + augmentation	0.6705	0.6461

3) RAG(Retrieval-Augmented Generation)

- RAG는 먼저 데이터베이스나 문서 집합에서 관련 정보를 검색하고, 검색한 정보를 바탕으로 텍스트를 생성하는 것을 말한다.
- RAG를 도입하고자 한 이유
  - 기존 모델은 수능 문제 풀이에 필요한 도메인 지식을 충분히 포함하지 못했으며, 할루시네이션 발생 가능성이 높아 별도의 Retriever를 활용하여 외부 정보를 제공할 필요가 있었다.
  - 주어진 GPU환경에서 돌릴 수 있는 최대 크기의 모델이 약 12B 크기
    →
    `GPT4o-mini`로 생성한 답변도 할루시네이션이 발생했다.
    - 수능 문제 풀이에 필요한 지식을 잘못 생성한 사례

      `ex)제너럴 셔먼호 사건은 1871년에 발생한 대원군과 청을 상대로 한 사건입니다.`

RAG 진행 단계

1. **Indexing**: 문서들을 Chunk로 분리하고 벡터로 인코딩하여 벡터 데이터베이스에 저장한다.

2. **Retrieval**: Query와 관련된 Top K개의 Chunk를 가져온다.
3. **Generation**: 원본 Query와 Retriever로 가져온 Chunk들을 LLM의 Input으로 넣어 최종 답변을 생성한다.

Indexing Data 선정은?

시험 문제 도메인이 역사, 경제, 정치, 지리, 심리, 윤리, 사회 등 다양한 분야를 포함하기에 이에 대한 정보를 담은 데이터를 활용하고자 하였다.

- 고등학교 교육 과정 내의 각 과목 교과서
  - 수능 도메인과 직접 관련된 고등학교 교과서를 사용하면 데이터의 품질과 관련성이 높았으나 **저작권 문제로 사용할 수 없었다.**
- **Wikipedia**
  - 다양한 주제와 방대한 정보를 포함하여, 수능 과목을 커버 할 수 있는 **관련 도메인 데이터가 다수 포함**되어있다.
  - 그리고 **저작권 문제가 없고, 지속적인 업데이트로 최신 정보가 포함**되어 있다.
  - 구조화 되어있어 **문단 단위 검색에 용이**하기 때문에 **Wikipedia 데이터 사용하기로 결정**하였다.

Indexing Data Preprocessing

- HTML, Fomula 등 **불필요한 부분**과 **공백을 제거** 후 **문단 단위로 Chunk**를 나누어 Indexing을 진행했다.
  - **Why?**
    - 문단 단위는 일반적으로 **한 주제에 대해 간결하고 명확한 정보를 제공**하고, 의미적으로 독립적인 내용을 담고 있다.
    - 긴 문서를 그대로 쓰면 모델의 **입력 텍스트 길이 제한**으로 정보가 잘려서 들어갔다.
    - 그래서 **문단 단위로 사용하기로 결정**하였다. (문단 개수: 3,520,854)
- **정보를 제공하기 어려운 문단도 관련 문서로 제공되는 문제를 발견**해서 이를 해결하고자 아래와 같은 작업을 진행하였다.
  - Paragraph 길이가 50자 미만인 문단 제거
  - `"단어."` 형태의 짧은 소제목 문단 제거
  - 2단어 이하이면서 마침표로 끝나는 문단 제거
  - **Why?**
    - `“클럽 경력.”` `“어학분야 공헌.”`

      → 이러한 문단은 의미 있는 정보를 제공하지 못한다.
    - 정보 부족의 예시
      - 50자 미만

        `“고구려는 삼국 시대 국가이다.”`
      - 50자 이상

        `“고구려는 삼국 시대의 주요 국가로, 장수왕 때 수도를 국내성에서 평양성으로 옮겨 남진정책을 강화했다.”`
  - **최종 문단 개수: 2,038,774**

Indexing

1. **Sparse Retriever (BM25)**
   - **BM25 알고리즘을 기반으로 한 Sparse Retrieval 방식**

    - `rank-bm25`보다 빠르게 동작하도록 구현된 `bm25s` 사용하였다.
    - 한국어 형태소 분석기를 활용하여 **문서 토큰화**를 진행하였다.
        - `mecab` → `C++`기반이라 속도가 빠르며, 실제 구현에서 `mecab`**을 사용해** 처리 속도를 개선

2. **Dense Retriever (FAISS,** Facebook AI Similarity Search**)**
    - **Dense Embedding 기반의 Retrieval 방식**
        - `faiss`를 활용하여 벡터 검색 구현하였다.
        - **IndexFlatL2**
            - 문서 간의 유클리디안 거리(L2 거리)를 기반으로 유사도를 계산하였다.
            - **Why?**
                - 데이터 크기가 커질수록 검색 속도가 느려지지만, 다른 압축 기법(N-List, IVF 등)과 비교해 **검색 결과가 손실 되지 않기 때문에 사용**하였다.
    - **임베딩 모델**
        - 한국어로 학습된 Sentence-BERT 모델인 `jhgan/ko-sbert-nli` 모델을 사용하였다.
        - **Why?**
            - NLI(자연어 추론) 데이터를 기반으로 학습되어 지문과 질문의 논리적 관계를 추론하는 데 유리할거라 생각했다.
    - **문서 및 쿼리 임베딩 생성**
        - 문서를 `jhgan/ko-sbert-nli` 모델을 활용해 임베딩 벡터로 변환한 뒤 `faiss`인덱스에 저장하였다.
        - 쿼리도 동일한 방식으로 임베딩한 뒤 `faiss`에서 관련 문서를 검색하였다.
3. **Hybrid Retriever (Sparse + Dense)**
    - **Sparse와 Dense Retrieval의 결합**
        - 지문뿐만 아니라 질문과 선지에 관련된 문서를 불러오기 위해 도입하였다.
    - **구현 방식**
        - Sparse로 초기 후보 검색
            - Sparse 방식(BM25)으로 전체 문서에서 지문을 Query로 하여 **Top-20 문서**를 검색하여 후보 문서군을 생성하였다.
        - Dense 방식으로 문서 유사도 계산
            - 문서군 중 질문과 선지 각각의 유사도에서 **가장 높은 점수를 받은 문서**를 최종 선택하였다.
        - 최종 문서 선정
            - 각각 **가장 관련성이 높은 문서**를 5~6개 선정하여 결과를 반환하였다.
              (질문, 선지 4~5개 각각)

적용 결과

기법	public	private
baseline	0.4332	0.4046
baseline + Sparse top5	0.4194	0.3632
exaone	**0.6705**	**0.6092**
exaone + Sparse top5	0.6452	0.5908

exaone + Dense top5	0.6475	0.6023
exaone + Hybrid	0.6429	0.5885

- baseline 모델에 Sparse Retriever 적용하였을 때, RAG적용 전 정확도보다 **감소**하였다.
  - **Why?**
    - 2B 크기의 모델이라 Retriever를 통해 주어진 문서를 프롬프트에 추가하면 프롬프트 길이가 길어져 모델이 제대로 추론하지 못했을거라 예상하였다.
- exaone모델(7.8B 크기)을 사용해 모델 크기를 키워서 적용하였으나 RAG적용 전보다 **감소**하였다.
  - **Why?**
    - 여전히 긴 프롬프트 처리하기 아쉬운 모델 크기
      - 2B 모델과 7.8B 모델 모두 긴 프롬프트에서 중요 정보를 놓칠 가능성이 있어, 프롬프트 최적화를 통해 성능을 개선할 여지가 있을 것 같다고 생각하였다.
    - Retriever가 제대로 문서를 못 가져왔을 가능성
      - 일부 소수 데이터에서 필요한 정보를 가져오는 것을 눈으로 체크하였지만
      - Sparse와 Dense Retriever 각각의 검색 정확도를 평가하고 분석할 수 있는 별도의 데이터 셋이나 평가 지표를 구성해서 체크해봤으면 좋았을 것 같다.
    - Retriever가 가져온 문서가 Generator가 답을 생성하는 데 적합하지 않은 경우
      - 지문과는 관련있으나 실제 풀이에 도움 되는 문서가 아닐 수 있어 hybrid 방식을 고안하였다.
        - Hybrid 방식도 여전히 RAG를 사용하기 전보다는 낮은 성능을 보였다.
      - 문제 유형마다 배경 지식이 필요하지 않을 수 있어서 문제를 유형별로 나누는 방식을 시도해봤었으면 좋았을 것 같다.
        - 추가된 문서가 문제 풀이에 오히려 방해가 될 수 있는 가능성이 있었다.
        - 원래 지문으로도 잘 푸는 문제와 그렇지 않은 문제를 분리했으면 더 좋은 성능을 얻을 수 있었을 것 같다.
- **최종적으로 RAG는 적용되지 않았음**

4) CoT (Chain-of-Thoughts)

일부 훈련용 데이터셋에는 모델이 가지고 있는 사전 지식에 의존해야 하는 case들이 존재했다.

지문	(가)은/는 의병계열과 애국계몽 운동 계열의 비밀결사가 모여 결성된 조직으로, 총사령 박상진을 중심으로 독립군 양성을 목적으로 하였다.
문제	(가)에 대한 설명으로 옳지 않은 것은?
선택지	1) 고려 문종 때에 남경(南京)으로 승격되었다. 2) 종루(鐘樓), 이현, 칠패 등에서 상업활동이 이루어졌다. 3) 정도전은 궁궐 전각(殿閣)과도성성문 등의 이름을 지었다. 4) 성곽은 거중기 등을 이용하여 약 2년 만에 완성되었다.
정답	1

- 모델이 이러한 사전 지식들을 충분히 보유하고 있는지 직접적으로 판단하기 힘들었기 때문에 충분히 배경지식을 가지고 있다고 가정하고 CoT 프롬프팅을 통해 모델이 학습한 지식을 최대한 사용할 수 있도록 유도하고자 했다.

Prompt Engineering

- 사용한 프롬프트의 예시

```
Prompt = """
1. 주어진 paragraph를 주의 깊게 읽고 핵심 정보를 파악하세요.
2. question을 정확히 이해하고, paragraph와 additional_information에서 관련된 정보를 찾으
3. 제시된 choices를 하나씩 검토하며 paragraph의 내용과 일치하는지 확인하세요.
4. 가장 적절한 답변을 선택하고, 왜 그 답변이 정답인지 간단히 설명하세요.
5. 최종 답변을 choices의 순서에 따라 1부터 5까지의 번호 중 하나로 제시하세요.
주어진 정보:
paragraph: {paragraph}
question: {question}
additional_information:{question_plus}
choices: {choices}


위 정보를 바탕으로 다음 형식에 맞춰 답변해주세요:

추론 과정:
1. 핵심 정보: [paragraph와 additional_information에서 파악한 핵심 정보]
2. 질문 분석: [question의 요점]
3. 선택지 검토:
   - 선택지 1: [적절성 여부 및 이유]
   - 선택지 2: [적절성 여부 및 이유]
   - ...
4. 결론: [가장 적절한 선택지 및 선택 이유]

최종 답변: [선택한 답변의 번호]

이 형식에 따라 주어진 질문에 대한 정답을 추론하고 설명해주세요.
"""
```

- Zero-shot-CoT

  Large Language Models are Zero-Shot Reasoners(Takeshi Kojima et al., 2022)에 따르면 프롬프트에 `“Let’s think step by step”`이라는 프롬프트를 추가하는 것 만으로(Zero-shot-CoT) 모델의 성능 향상을 확인할 수 있었다.

  이 방법을 베이스라인에 구현하기 위해 프롬프트에 `“단계별로 생각하여 답을 구하세요”`라는 프롬프트를 추가하여 CoT를 자체적으로 생산하고, 해당 CoT를 통해 추론을 진행하도록 하였다.
  하지만 논문에서 제시한 Zero-shot-CoT 방법론의 한계로는 모델의 크기가 작을 때는 큰 효과를 보지 못했다는 것이었고, 해당 프로젝트 환경에서 사용할 수 있었던 10B 내외 크기의 모델들에선 유의미한 결과를 내지 못했다는 것을 확인할 수 있었다.
  실험의 결과는 Zero-shot-CoT를 적용했을 때의 점수(public:
  0.6429, private: **0.5908**)가 적용 하지 않은 기존 모델의 점수(public: **0.6590**, private: **0.6322**)보다 낮은 수치를 기록해 논문이 제시한 한계점을 확인할 수 있었고, 최종적으로 적용하지 않았다.

  프롬프트 예시)

```
if self.zero_shot_cot:
    user_message.replace("정답:", "")
    tmp_user_message = user_message + "단계별로 생각하여 답을 구하세요."
    response = self.model.generate(self.tokenizer(tmp_user_message, return_tens
```

```
    generated_text = self.tokenizer.decode(response[0], skip_special_tokens=Tru
    user_message += generated_text + " 따라서 정답:"
```

Post-processing

- 모델이 생성한 답변의 품질은 모델마다 상이하고 원하는 형태로 일관된 정답을 생성하지 않았기 때문에 이러한 답변을 후처리 하는 것도 중요했다.

Case	설명	생성된 답변의 예시
Case 1	정답을 고르지 못함	없음 (주어진 선택지 중 적절한 설명이 없음) 없음 (모든 선택지가 적절함)
Case 2	선택지 번호가 아닌 답변 출력	가장 적절한 답변은 "오징은 ㄱ과 ㄹ에 동의하지 않겠군."입니다. [C] '2000만원'
Case 3	중복된 정답 선택	[3, 4] 1 또는 3
Case 4	정답을 출력하지 못함	핵심 정보: - 경마식 보도는 선거와 정치에 무관심한 유권자들의 참여를 독려할 수 있지만, 선거의 공정성을 저해할 수 있다. - 공직선거법과 선거방송심의에 관한 특별규정, 선거 여론조사보도준칙 등이 경마식 보도의 문제점을 줄이기 위한 조치로 제시되어 있다. - 선거 방송 토론회는 후보자 간 정책과 자질을 비교할 수 있는 중요한 매체로 활용될 수 있다. 핵심정보 : .. (이후 반복)
Case 5	추가적인 정보 출력	5 (모든 선택지가 적절함) [5]

- Case 5와 같이 비교적 처리하기 쉬운 예제들은 정규식을 이용하여 처리하고 나머지 Case들에 대하여 프롬프트에 강한 제약조건을 주어 다시 답변을 생성하게 하는 방식을 사용하였다.
- 사용한 모델마다 편차가 존재하긴 하였으나 일반적으로 전체 inference set중 70%정도는 정상적인 답변을 생성했고 남은 30% 중 10%정도는 Case 5에 해당하는 처리가 용이한 답변들이었다.
- 일반적으로 보다 강한 제약조건을 주어 다시 생성하게 되면 대부분의 Case들이 올바른 형식의 답변을 생성하는 모습을 보였고 일부 처리되지 못한 예시들은 여러 모델들이 중복으로 답변한 데이터들을 정답으로 사용했다. 하지만 사용했던 모델들의 크기가 비교적 작아 fine-tuning 없이는 좋은 성능을 보이지 못했다. (public : **0.5876**, private : **0.5517**)

Knowledge Distillation

- 우리가 사용하는 모델 중 가장 큰 모델의 크기는 12B으로, 유료 API로 사용할 수 있는 LLM에 비하면 턱없이 작았기 때문에, 프롬프트 엔지니어링만으로 좋은 품질의 CoT를 생성하기에는 다소어려움이 있을 것으로 생각했다.
  따라서, 프롬프트 엔지니어링 외에도 훈련용 데이터셋의 각 데이터에 적절한 CoT를 만들어 이를 label로 사용하여 supervised learning을 진행하는 방법을 고려하였다.
- 훈련용 데이터셋에 포함된 데이터의 수는 2031개로 라벨링 작업을 손수 진행하는 것은 다소 무리가 있을 것이라 판단했다.
  따라서 OpenAPI를 이용하여 Knowledge distillation으로 인한 성능 향상 또한 기대할 수 있으면서도 효율적으로 CoT를 생성하도록 하였다.
  - `gpt-4o-mini`
    - 가장 저렴하게 사용할 수 있는 모델.
      이 모델로부터 얻은 CoT로부터 할루시네이션 현상을 확인할 수 있었고, 이와 같은 label을 사용하게 된다면 학습한 모델 또한 할루시네이션 현상을 피할 수 없을 것이라고 판단하여 최종적으로는 선택하지 않았다.
    - ex. `제너럴 셔먼호 사건은 1871년에 발생한 대원군과 청을 상대로 한 사건입니다.`

    - `gpt-4o`
        - `gpt-4o-mini` 가 생성한 CoT 중 할루시네이션이 나타났다고 판단한 데이터를 기준으로 비교해보았을때, `gpt-4o` 는 할루시네이션이 나타내는 정도가 매우 적다고 판단하였다.
        - ex. `Pyongyang was the location where the General Sherman Incident occurred in 1866`
    - `gpt-o1-mini`, `gpt-o1-preview`
        - OpenAI에서 제공하는 유료모델로 수능 문제를 풀게 하는 개인연구(링크)에 따르면, `gpt-o1-mini` 는 `gpt-4o` 보다 비싸면서도 더 낮은 성능을 보이는 것으로 나타나 시도해보지 않았다. `gpt-o1-preivew` 가 가장 높은 성능을 보이는 것으로 나타났지만, `gpt-4o` 에 비해 최소 4.8배 비싼 것으로 확인하여 시도해보지 않았다.
    - 따라서 가격과 성능을 종합적으로 판단했을 때 가장 적절한 모델은 `gpt-4o` 라는 판단하에 labeling 작업을 진행했다.
- 훈련용 데이터셋에 CoT label을 학습하여 테스트 데이터셋에 대한 CoT를 추출(모델 A)한 후, 문제와 CoT를 확인하여 정답을 logit 기반으로 추출(모델 B)하는 **2-step 파이프라인**, CoT를 생성하면서 마지막에 정답도 함께 생성하도록 하는 **1-step 파이프라인**, 총 2개의 방법론으로 시도해보았다.
    - **결론 1) 2-step 파이프라인 성능(**public : **0.6452** / private: **0.6092)이 1-step 파이프라인 성능(**public: **0.6336** / private: **0.6046)보다 좋았다.**
        - 분석: 1-step 파이프라인에서 rule-base로 post-processing을 진행한 것보다 2-step으로 모델이 직접 생성한 CoT를 보며 무엇이 정답일지 파싱하여 정답으로 내놓는 것이 더 효율적인 것으로 생각된다.
    - **결론 2) 베이스라인 모델 성능(**public: **0.6590** / private: **0.6322)이 2-step 파이프라인 성능(**public : **0.6452** / private: **0.6092)보다 좋았다.**
        - 분석: CoT가 처음 제안된 논문을 보면 10B 이하의 모델에서는 CoT를 사용하는 것이 대부분 모델 성능을 해친다고 한다. 비록 논문에서 진행한 실험과 우리 팀에서 진행한 실험이 완전히 같지는 않지만, 적은 크기의 모델에서는 비슷한 양상을 보이는 것으로 생각된다.

5) Model

베이스라인

- 모델을 선정하기 전에, 가장 먼저 베이스라인 코드에 들어있는 모델을 돌려보기로 했다. 모델의 이름은 `beomi/ko-gemma-2b` 모델이었으며, 기본적인 learning rate는 2e-5, epoch은 3, batch_size는 1로 설정했으며, LoRA의 경우 rank는 6, 그리고 alpha 값은 8로 설정되어있었다. 뿐만 아니라, VRAM 메모리의 제약으로 인해 input 데이터의 길이가 1024 이상인 경우에는 제거가 되었다. 그 결과, 성능은 Accuracy를 기준으로 `(public: 0.3986, private: 0.3908)` 이라는 점수가 나왔다.

input 데이터의 길이가 1024 이상인 경우에도 포함시키기

- 가장 먼저 시도해 본 방법으로는 input 데이터의 길이가 1024 이상인 경우에도 데이터를 포함시키도록 하는 것이다. 길이 '1024'는 지문을 tokenizer를 거쳐 토크나이징이 된 이후를 기준으로 한다. 그렇게 하면 LLM 모델이 훨씬 더 긴 지문을 추가적으로 학습하기 때문에, 길이가 긴 지문을 보았을 때 더 잘 대응할 수 있을 것이라는 생각이 들었다. 결과적으로 성능이 `(public: 0.4332, private: 0.4046)` 이 나왔으며, 이를 통해 긴 지문을 학습하는 것이 성능을 올릴 수 있는 요인이 되었다는 사실을 알 수 있었다.
- 그러나, 모델의 크기가 커지면서 메모리의 제한으로 인해 1024 이상인 데이터를 포함시키는 것이 점점 힘들어졌고, 결국 모델의 크기와 데이터의 길이는 트레이드 오프 관계가 되었다. 그리고, 이 둘 간의 적절한 균형을 찾는 것이 중요한 과제가 되었다.

모델의 크기를 키워보기

- 그 다음으로 진행한 부분은 바로 단순하게 '모델의 크기를 키워보는 것'이었다. https://arxiv.org/pdf/2001.08361 에도 주장하는 것 처럼, 모델의 크기와 성능은 상관관계가 존재하며 이것이 LLM의 핵심 요소라는 생각이 들었다. 이를

증명하기 가장 쉬운 모델은 바로 `beomi/ko-gemma-7b` 모델이었다. 기존 베이스라인 모델과 동일한 `gemma` 모델을 기반으로 fine-tuned된 이 모델을 사용하니 결과적으로 `(public: 0.4839, private: 0.5241)` 이 나왔다. 이를 통해 모델의 크기를 키우는 것이 성능 향상의 결과로 이루어질 수 있다는 사실을 알 수 있었다.

- 모델의 크기는 성능에 직결된다는 사실을 확인했으니, 이번에는 10B 단위의 모델도 적용해보기로 했다. 세상에는 수많은 모델들이 있었고, 그 모든 모델들을 일일이 시험해보는 것은 불가능하다. 따라서 명확한 지표를 기반으로 한 리더보드(링크)를 참고해야 했다. V100 GPU 서버 특성상 모델 크기에 한계가 있을 거라고 생각했다. 10B 이상의 모델을 올리기 위해서는 여러 가지 과정을 거쳐야만 했다.
  - 가장 먼저, tokenizer.chat_template이 기존 gemma 모델과는 많이 달랐다. 따라서, `DataCollatorForCompletionOnlyLM` 모델이 제대로 동작하지 않았다. 이 문제를 해결하기 위해 `custom_data_collator` 함수를 직접 만들고, 정확도를 측정하는 `compute_metrics` 함수도 따로 만들어야 했다.
  - 10B 모델이 넘어가면서부터는 `OOM(Out Of Memory)` 현상이 자주 발생했기 때문에 양자화를 적용하였고, `bfloat16` 의 형식을 사용했다. fp16과 비교했을 때 bfloat16의 경우 정확도는 더 떨어질 수 있지만, 훨씬 더 안정적이기 때문에 이를 채택했다. 그리고, 계산량을 줄이기 위해 `초기 레이어를 동결`시키는 코드를 작성한 뒤 이를 반영했다.
  - 개인적으로는 병렬 컴퓨팅과 deepspeed를 사용할 수 없는 환경이었다는 점이 상당히 아쉬웠다. 이 두 가지 방법이 모두 가능했다면 LLM의 성능을 크게 끌어올릴 수 있었기 때문이다.

모델	public	private	추가적인 기법
T3Q-LLM/T3Q-LLM1-sft1.0-dpo1.0	0.6705	0.6161	
LGAI-EXAONE/EXAONE-3.0-7.8B-Instruct	0.6682	0.6276	test_valid_split : 0.2
LGAI-EXAONE/EXAONE-3.0-7.8B-Instruct	0.6590	0.6322	
yanolja/EEVE-Korean-10.8B-v1.0	0.6544	0.6253	
upstage/SOLAR-10.7B-v1.0	0.6175	0.5517	

하이퍼파라미터 튜닝

- 하이퍼파라미터를 튜닝하는 작업은 생각보다 훨씬 중요했다. 최적의 성능을 찾기 위해 epoch을 3으로 고정시킨 다음, learning rate를 조절하는 방식으로 최적의 성능을 찾아보고자 했다. 가장 먼저, 베이스라인 코드의 learning rate인 2e-5에서 이를 1/10로 줄인 2e-6으로 해보고, 이 둘 중에서 성능이 더 좋은 쪽에 좀 더 가까운 learning rate를 테스트해보는 방식으로 진행했다.
- LLM을 사용하면서 효율적으로 파인튜닝을 진행하기 위해 LoRA를 사용했다. 이 때, `LoRAConfig`을 통해 조정할 수 있는 파라미터는 두 가지가 존재한다. 첫 번째는 바로 `r` 로, Low-Rank 행렬의 차원을 의미한다. r이 작을 수록 학습할 수 있는 파라미터의 개수가 줄어들며, r이 커질수록 반대로 표현력이 증가할 수 있다. 보통 r의 값은 8에서 64사이이며, OOM Issue가 발생하지 않는 선에서 최대한 r의 개수를 키우는 방식으로 진행했다. 두 번째는 `alpha` 인데, 이 값은 LoRA 가중치의 스케일링에 사용되는 파라미터로 학습된 가중치를 축소하거나 확장하는 데에 기여할 수 있다. 이 값은 보통 r과 동일한 값을 사용하는 편이지만 실험적으로 최적의 가중치를 찾았어야 했다. 이외에도 dropout 도 성능에 영향을 미치긴 하지만 0.05로 이를 고정했다.
- `gradient_accumulation_steps`를 늘리면 배치의 크기가 증가하는 역할을 하기 때문에 더욱더 안정적이다. 그리고 steps를 늘릴 수록 노이즈에 더 강하기 때문에 learning rate도 동시에 증가시켜야 한다는 특징을 가지고 있다. 그러나 지나치게 큰 값을 가지게 되면 오히려 성능이 감소하기 때문에 최적의 지점을 찾는 것 또한 중요한 과제였다.
- `train_valid_split` 함수를 사용해서 학습 데이터와 검증 데이터를 랜덤하게 나눌 수 있는데, 이 비율을 어느 정도로 할 것인지도 성능에 중요한 영향을 발휘했다. 검증 데이터의 비율이 너무 적으면 모델의 일반화 성능을 측정하기 어렵고, 반대로 검증 데이터의 비율이 너무 클 경우에는 학습 데이터가 너무 적어 학습 자체가 잘 안될 수도 있기 때문에, 이 사이의 최적점을 찾는 것이 중요했다. 총 3가지의 비율(0.1, 0.2, 0.3)을 가지고 실험하였으며, 결과적으로 0.2가 가장 좋은 성능을 낼 수 있었다.

K-Fold Cross Validation

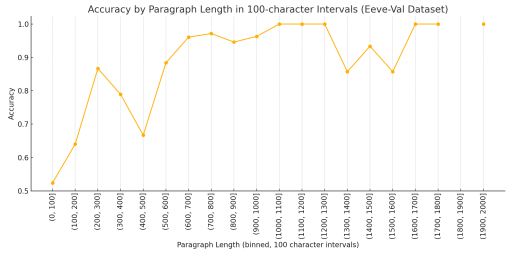
- K-Fold Cross Validation은 단일 검증 세트를 기반으로 한 기존 방식의 단점을 극복하기 위해 만들어진 방식으로, 만약 검증 세트가 특정 데이터셋을 대표하지 못한다면 진행할 수 있는 방식이기도 하다. 모든 데이터가 검증 방식에 사용

되기 때문에 샘플링의 편향이 줄어들며, 보다 신뢰성이 있는 측정이 가능하다.

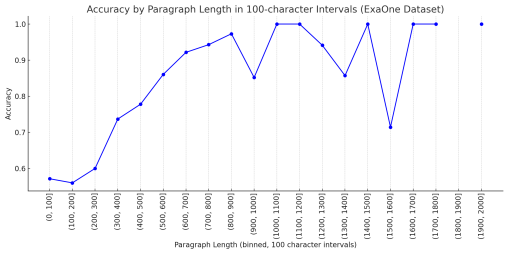
- 1번부터 5번까지의 데이터 분포가 편향되어 있으므로 클래스의 원본 비율을 유지할 수 있는 `StratifiedKFold`를 사용하는 것이 더 나은 판단일 수 있다는 생각이 들었다. (Fig. 4)

지문의 길이와 모델의 성능

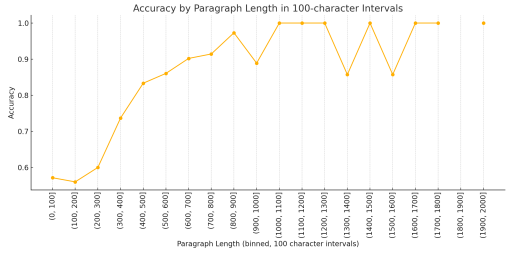
훈련용 데이터셋을 기반으로 valid 데이터셋의 점수를 매겨본 뒤, 해당 모델이 어떤 지문에서 정답을 잘 맞추고, 어떤 지문에서 정답을 못 맞추는 지에 대해 분석을 해보고 싶었다. 그리고 한 가지 유의미한 결과를 얻었었는데, 바로 '지문의 길이가 짧을 수록 정답률이 낮아지는 경향'이었다. 그리고 이는, 지금까지 시도해본 모델 4가지를 기준으로 실험을 해본 결과 모두 동일한 경향성을 드러내는 것으로 확인되었다.



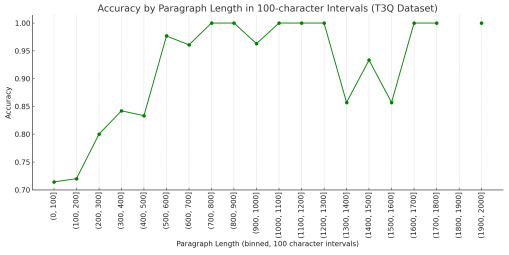
[Figure 8.1] 지문의 길이에 따른 정답률 (EEVE)



[Figure 8.2] 지문의 길이에 따른 정답률 (EXAONE)



[Figure 8.3] 지문의 길이에 따른 정답률 (Mistral)



[Figure 8.4] 지문의 길이에 따른 정답률 (T3Q)

- 위의 그래프는 Mistral, EEVE, T3Q, EXAONE의 지문 길이에 따른 정답률을 나타낸 것이다. 100단위로 문단을 쪼갠 것인데, 지문의 길이가 1000 이하인 경우에는 지문의 길이가 짧을 수록 정답률이 낮아지는 경향을 보였다. 그러나 1000이상의 길이를 가진 경우에는 정확도가 꽤나 높은 것으로 확인이 되었다. 그리고 그 이유로 '정보력의 부족'을 생각했다.
- 특정 모델의 경우 특정 길이의 지문 하에서 성능이 가장 좋게 나올 것이라는 가정을 하고, 100단위로 지문의 길이를 잘라서 그 구간 내에서 가장 성능이 좋은 모델을 추린 뒤 hard voting 앙상블을 진행했다. 그 결과 성능이 하락했기에 해당 가설은 폐기하기로 했다.

6) Ensemble

- 기본적으로 주어진 훈련용 데이터셋을 바탕으로 그 중 20%를 추출하여 validation 데이터셋을 확보하였다.
- 확보한 validation 데이터셋을 이용하여 각 모델이 훈련 종료 후, test 데이터셋 뿐만 아니라 validation 데이터셋에 대하여 추론을 진행하였고, 그 과정에서 logit이 가장 높은 선지의 번호 뿐만 아니라 soft voting을 위한 각 선지에 대한 logit 값을 추출하였다.
- 먼저 가장 다양한 모델을 포함하여 앙상블을 진행하였고, 앙상블한 후의 validation 점수와 public 점수를 종합적으로 고려하여 특정 모델만 제외하거나, soft voting 대신 hard voting을 진행해보는 등의 A/B 테스트를 거쳐 단일 모델 SOTA(public: **0.6705,** private: **0.6184**) 대비 약 0.02% 성능 향상(public: **0.6912**, private: **0.6460**)을 얻을 수 있었다.

자체 평가

잘했던 점

- GitHub issue, Discussion, Projects 등을 이용해 각자의 작업을 공유하고 협업할 수 있었다.
- Notion을 활용해 서버 활용 현황과 실험 기록 관리
- 10시~17시까지 Zoom 회의실 접속 후 캠 켜기

아쉬웠던 점

- PM의 부재
- 작업 분할을 더 세분화 했으면 좋았을 것 같다.
- 가설을 검증하기 위한 validation 데이터셋을 미리 선정하고 보다 효율적으로 실험결과를 검증하지 못한것이 아쉬웠다.
- 데이터셋의 분포나 노이즈 등을 더 자세하게 분석하지 못한 점이 아쉬웠다.
- 프로젝트를 구체적으로 어떻게 진행할지에 대한 파이프라인이 명확하게 설계되지 못한 점이 아쉬웠다.
- 실험 기록 관리가 자세하게 잘 되지 않았던 점이 아쉬웠다.

시도했으나 잘 되지 않았던 것들

- 모델 Quantization을 통해서 보다 큰 모델을 불러오려고 시도했으나 메모리 문제로 잘 되지 않았다.
- 학습 데이터의 라벨 편향성을 제거해 모델의 편향을 없애주어 성능 향상을 시도하였다.**(Fig 6.)**
  하지만 균형잡힌 데이터셋으로 학습하였을 때 오히려 더 낮은 성능을 보여주었다. 원인 분석은 테스트 데이터셋 역시 편향된 분포를 가진게 아니었을까 하는 추측 정도로 남겨두었다.
  (original - public:
  0.6590, private: **0.6322** → balanced - public: **0.6567**, private: **0.6092**)
- vllm을 통해 추론 속도를 향상시키려고 시도했지만 추론의 품질이 많이 떨어지는 현상이 일어났다.
- Zero-shot-CoT를 시도해보았다. 하지만 연구가 제시하는대로 작은 모델에서 큰 효과를 보지 못했다.
- 수능 문제 풀이에 필요한 도메인 지식을 모델이 충분히 포함하지 못하기에 발생하는 할루시네이션을 **RAG**로 해결하고자 하였으나 오히려 약간의 성능 하락을 겪었다.

프로젝트를 통해 배운점 또는 시사점

- 안된다는 생각을 버려라. 하면 된다!
- 가설을 설정하고 검증하는 과정에서 가능한 한 변인을 통제할 수 있는 방법을 찾는 것이 중요하다는 점을 알았다.
- 공유 문화가 활성화되어야 한다.
- PM의 중요성

개인 회고

강전휘

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  EDA를 통해 데이터셋 중 모델이 사전지식을 학습하지 않았다면 풀기 어려운 예제가 있음을 확인하였고, 이러한 문제들을 잘 풀기 위해서는 보다 많은 데이터로 학습 된 큰 LLM을 이용하는 것이 유효할 것이라고 생각했음. 특히 역사, 사회, 문화와 관련된 예제들은 Multilingual 모델들이 아무리 우수한 한국어 성능을 가지고 있다고 하더라도 한국에 특화된 기반지식을 충분히 학습하지 못했을 것이라고 생각했고 Model들을 선택할 때 한국어 corpus로 fine- tuning 된 model들을 중심으로 탐색했다.

제한된 gpu 내에서 최대한 많은 파라미터를 가지는 모델을 로드하고 학습하기 위해서 quantization과 같은 방법들을 시도해보고자 했다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  기존에 Data Augmentation을 위하여 open source llm을 많이 이용했으나 예제들을 생성하거나 text를 정제하는 데에 많은 시간이 걸렸고, 각각의 모델들에 특화된 chat template을 이용하지 못했다는 아쉬움이 있었다.

  이런 문제에 대한 해결방안으로 생성된 텍스트의 품질을 향상시키기 위해 instruct tuning된 모델의 chat template 형식에 맞게 prompt를 설정하였고 여러가지 prompt engineering을 적용하여 원하는 작업에 알맞은 답변을 생성하도록 하였다. 실험한 모델들이 대부분 10b 이하의 작은 모델들이었음에도 불구하고 주어지는 prompt에 따라 생성되는 결과값이 크게 달라지는 모습을 관찰할 수 있었고 추론속도를 증가시키기 위하여 vllm과 같은 라이브러리를 사용하였다. 결과적으로 추론속도를 3배 이상 향상시킬 수 있었으나 생성된 답변의 품질이 vllm 없이 생성했을 때보다 하락한 점은 아쉬운 점으로 남았다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  주어진 DATA를 충분히 분석하지 못한점이 아쉬웠다. 자연어 데이터들은 수치적인 분석으로 얻을 수 있는 유효한 정보가 많지 않았고, 추가적인 분석을 시도해보지 않은 채 문장길이의 분포, 언어의 분포, 질문의 종류 등 단순한 분석만을 수행한 채 실험을 진행했고, 실험 결과가 예상대로 도출되지 않을 때마다 데이터셋을 직접 관찰하여 원인을 분석하는 경우가 잦았다. 미리 주어진 데이터셋이 가지고 있는 정보들을 충분히 관찰하고 분석한 후 실험에 대한 가설을 설정했다면 보다 효율적으로 실험들을 진행할 수 있었을텐데 귀찮고 어렵다는 이유로 간과한 점이 아쉽다.

  모델에 추가적인 정보를 주입하기 위해서 fine-tuning하는 방법론들을 사용해보고 싶었으나 DPO와 같은 RLHF방법론들은 큰 비용이 들고 사용가능한 데이터셋을 확보하는 것이 매우 힘들었고 이에 따라 GPT와 같은 상용모델을 이용해 데이터를 생성하려고 시도했으나 비용적인 한계로 시도해보지 못한 점이 아쉬웠다. 어째서 많은 기업들이 학습을 위한 질 높은 데이터를 확보하는 것에 초점을 맞추고 많은 비용을 투자하는지 느낄 수 있었다.

  모델 quantization를 시도하던 중 라이브러리 버전이나 cuda설정과 같은 시스템 환경 문제로 잘 동작하지 않아 몇번 시도하지 않고 포기했던 점이 가장 아쉽다. 특히 설정값을 여러번 바꿔가면서 계속 시도해 결국 성공한 사례가 있음을 나중에 알게 된 후에는 조금 더 끈기를 가지고 수행했으면 좋았겠다는 아쉬움이 남았다.

  COT와 같은 prompt 기반의 in context learning이 유의미한 성능 향상을 보여줄 것으로 기대했으나 상대적으로 작은 모델을 사용해서인지 극적인 성능향상을 얻지 못한 점이 아쉬웠다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  모델을 학습하거나 새로운 방법론들을 실험하기 전에 학습의 정도나 결과를 미리 확인해볼 수 있도록 dev set을 구성하고 dev set의 결과들을 바탕으로 실험의 진행도를 조절하는 것이 중요하다고 느꼈다. 이러한 dev set을 설정하지 않으면 실험의 결과값이 일관되지 않아 어떤 부분을 개선하여 다시 실험을 수행할지에 대한 명확한 기준을 정하기 어려웠고 전체 데이터셋을 통해 수행한 실험들은 결과값을 얻기 위해 상대적으로 매우 많은 시간이 소요되며 만약 원하지 않은 결과가 나왔을 경우 이렇게 소모한 시간들이 매우 무의미하게 낭비되는 경우가 많았다. 그러므로 다음 프로젝트에서는 미리 팀원들 간에 동일한 dev set을 공유하고 각자 진행한 실험들의 결과값이 일관되어 이러한 결과를 공유하는 것이 프로젝트를 진행하는 데 있어서 조금 더 도움이 될 수 있도록 해보고 싶다.

  llm기반의 데이터 증강을 수행하기 위해 quantization 등의 기술을 통해 충분히 큰 모델을 사용할 수 있도록 확보한 후 prompt learning을 수행해 보고 싶다.

박상준

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  가장 먼저 최적의 모델을 찾기 위해 수능 국어 및 문제 풀이와 관련된 모델들을 취합하여 가장 좋은 모델을 찾아보고, 이 모델이 현재 서버인 Tesla V100 모델 위에서 제대로 돌아가는 지 테스트를 해보았다. 그 과정에서 크기가 큰 모델

들을 성능이 좋지 않은 GPU 위에서 어떻게 하면 돌릴 수 있을 지 고민해보고, Perplexity, ChatGPT, 구글링 등을 활용해서 답을 찾아내기 위해 노력했다.

그 다음으로는 어떻게 하면 한정된 데이터를 기반으로 성능을 올릴 수 있을 지에 대해서 고민을 했다. 비록 대회의 시간 자체가 한정되어 있었기에 시도를 해보지는 못했지만, 지문의 길이가 짧은 데이터와 지문의 길이가 긴 데이터의 성적을 비교해보고, 지문이 짧은 데이터에서 성적이 좋지 않게 나온 것을 발견해서 이를 기반으로 RAG를 이용해서 보완하면 좋을 것 같다는 아이디어가 나왔다.

뿐만 아니라 이번 프로젝트 기간에 배웠던 분야가 '프롬프트 엔지니어링'이었던 만큼, 이 주제를 기반으로 문제를 풀고자 했다. 수능 국어 지문을 잘 풀었던 모델이 어떤 프롬프트 방식을 차용했었고, 어떤 방법이 가장 효과적이었는 지를 중심으로 우선순위를 적용해 문제를 풀기로 했다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  깃허브의 discussion을 활용해서 트러블슈팅 노트를 기록하고, 어떤 부분에서 문제가 생겼는 지, 그리고 이 문제를 어떻게 풀었는 지에 대한 기록을 적어놨다. 결과적으로 다른 사람들이 같은 문제가 생겼을 때 트러블슈팅 노트를 기반으로 문제를 풀 수 있게 되었다.

  github의 discussion을 기반으로 논의가 될 만한 부분이나 정리를 해볼 만한 것들을 정리하는 작업을 진행했었다. 다른 사람들과 정보를 공유할 때 discussion을 기반으로 정보를 공유할 수 있도록 했다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  팀이 새롭게 결성된 지 얼마 되지 않았기 때문에 시스템이 충분하게 갖춰져 있지 않았다는 생각이 들었다. 각자가 하고 싶은 부분들을 하고, 피어세션 때 이를 하나로 모아서 작업을 하는 것을 진행했는데, PM이라고 하는 존재가 없었기 때문에 탑-다운 방식이 제대로 이루어지지 않았다. 결과적으로 팀원 간에 협업보다는 개인적으로 일을 진행하는 경우가 많았다.

  디테일한 부분에 대해서도 아쉬움이 있었다. 한계를 지어버리면 그 이상으로 나아가지 못했는데, 그런 부분이 많이 부족했다. 데이터를 최대한 디테일하게 분석해보고, 이를 기반으로 다음에 나아갈 방향을 결정하는 것이 좋은데, 데이터 분석에 대한 부분이 약간 소홀했다. 무언가를 할 때도 구체적인 계획을 기반으로 하기보다는 그때그때 생각하는 것으로 진행했었는데, 이러한 부분에서 많이 부족했다는 생각이 들었다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  가장 먼저 PM을 한 명 정하고, 이를 통해 서로가 어떤 일을 하면 좋을 지 효율적으로 관리를 하는 작업이 먼저 이루어져야 한다는 생각이 들었다. 안된다는 생각도 버려야 한다고 생각한다. 안된다는 생각을 하는 게 스스로에 대해 한계를 결정짓는 것이라는 생각이 들었기 때문이다. 일을 진행할 때 너무 시급하게 결정하는 부분이 있었는데 이에 대한 계획을 짓고 시작하는 방식으로 진행해보고 싶다.

박준성

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  Chain-of-Thoughts 기법을 통한 성능 향상을 위해, 먼저 gpt-4o를 활용하여 훈련용 데이터셋으로부터 적절한 해설을 생성하고, 이를 기반으로 문제에 대한 해설을 잘 학습하도록 일종의 knowledge distillation을 시도해보았다.
  또한, soft voting으로 앙상블을 진행하기 위해 추론 결과로 logit 값을 저장할 수 있도록 구현하였고, 보다 근거 있는 앙상블을 진행하기 위해 고정된 validation 데이터셋을 구축하여 다양한 모델에 대해 추론을 진행할 수 있도록 주도하였다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

GitHub Discussion의 도입이 아주 좋았다고 생각한다. 기존에는 Notion을 위주로 문서화를 진행했는데 인터페이스가 크게 익숙하지 않아 문서가 좀 정리되지 않았다는 느낌을 받았는데, GitHub Discussion은 한 눈에 잘 들어올 수 있도록 문서를 작성할 수 있어서 좀 더 이용하기 좋았다는 인상을 받았다.
이용하기 더 좋다고 느끼다 보니, 더 자주 지금 진행하는 작업을 보기 좋게 문서화를 잘 할 수 있게 되고 각 팀원이 어떤 작업을 하고 있는지 잘 공유할 수 있어서 좋았던 것 같다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  PM의 부재를 뼈저리게 느낀 것 같다. 하나의 팀이라기 보다는 6명의 개인이 각각 하고 싶은 일을 했던 것 같다. 팀의 공동 목표를 위해서는 그래도 누구 한명이 앞장서서 중심을 잡아줬어야 하지 않았을까 하는 아쉬움이 남았던 것 같다. 데이터 EDA도 훈련용 데이터셋에 대해서만 진행했던 것도 좀 아쉬웠던 것 같다. 고정된 validation 데이터셋을 더 빨리 만들고 inference 결과가 어떻게 되는지 확인하는 데이터 기반의 의사결정을 진행했어야 했을 것 같아 아쉽다는 생각이 든다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  기업 해커톤에서는 누구 한명이 PM을 해야 되지 않을까 생각한다. 일의 마감기한을 정하고 그 전까지 어떤 metric이 어느 정도 수치가 안나오면 깔끔하게 포기하는 식의 방향으로 갔어야 했을 것 같다는 아쉬움이 들어 다음에는 이런 방식으로 협업을 하면 좋을 것 같다.
  또 고정적인 validation 데이터셋이 필요하다는 건 알고 있었지만 자꾸 우선순위가 뒤로 밀려서 결국 마지막에 급하게 진행한 느낌이 되어 아쉬웠다. 이 부분도 초반에 고정적인 validation 데이터셋을 확보해놓고 테스트셋과 경향이 맞도록 수정하는 작업을 진행했으면 어떨까 싶어 다음 프로젝트에는 이 부분을 신경써보고 싶다.

백승우

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  Retrieval-Augmented Generation(RAG) 기반 시스템을 구축하여 수능 문제 풀이에 필요한 도메인 지식을 외부에서 검색하고 이를 활용해 정확도를 향상시키는 것을 목표로 삼았다. 특히, Sparse와 Dense Retrieval 방식을 비교하고 Hybrid 방식으로 결합하며, 주어진 제한된 환경에서 성능을 최대한 높이고자 했다.
  RAG 구현 과정에서는
  Wikipedia 데이터의 전처리 및 효율적 Indexing에 초점을 맞췄으며, **BM25S**와 **FAISS**를 활용하여 Retrieval 시스템을 구축하고, 이를 통해 문제 풀이에 적합한 문서를 효율적으로 검색할 수 있도록 했다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  이번 프로젝트에서는 RAG를 도입하여 **기존의 단일 모델 의존 방식에서 벗어나 외부 정보를 활용**하는 방식을 시도했다. 특히, Sparse Retrieval과 Dense Retrieval 외에 이 두 가지를 결합한 **Hybrid Retrieval**을 통해 질문과 선지 각각에 대해 유사도를 계산하여 더 정교하게 문서를 선택하는 방식을 구현했다.

  그러나 성능 측면에서는 기대만큼의 효과를 얻지 못했으며, 이는 **프롬프트 최적화**와 **Retriever 성능 개선**이 필요하다는 교훈을 얻을 수 있었다.

  또한, 이번 프로젝트를 통해 나 자신에 대한 변화도 경험했다. **'왜'라는 질문을 더 자주 던지며 근거 있는 결정을 내리려고 노력**했다. 단순히 구현하고 결과를 확인하는 단계를 넘어서, **각 구현 선택의 이유를 명확히 하면서 작업을 진행**하니, 프로젝트를 단계적으로 정리하고 체계적으로 발전시키는 데 도움이 되었다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  프로젝트를 진행하며 몇 가지 한계에 부딪혔다. 첫째, Sparse와 Dense Retriever의 성능을 정량적으로 평가할 수 있는 **지표와 데이터 셋**을 구성하지 못해 검색 정확도를 객관적으로 분석하지 못했다. 둘째, **Retriever가 가져온 문서가 실제 문제 풀이에 적합하지 않은 경우**도 있었다. 예를 들어, 지문과 관련은 있지만 풀이에 직접적으로 도움을 주지 않는 문서를 가져오는 문제가 발생했다. 셋째, **문제 유형별로 배경 지식의 필요성을 구분하지 못해** 불필요한 문서가 추가된 문제에서 성능 저하가 나타났다.

이러한 한계들은 구현을 더 빠르게 마쳤다면, 직접 모델의 예측 값과 정답을 비교하며 **더 많은 분석을 진행**할 수 있었을 것이라는 아쉬움으로 이어졌다.

또한, 협업 측면에서는 **작업의 분배와 일정 관리가 더 효율적으로 이루어지지 못한 점**이 아쉬웠다. 새로운 팀으로 협업을 시작하며, 기존보다 더 **개인적인 작업**으로 프로젝트가 진행된 경향이 있었다. **PM 역할을 담당하는 리더가 있었다면 일정 관리와 작업 분배가 더 체계적으로 이루어졌을 것**이라는 아쉬움이 있다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  이번 프로젝트를 통해 단순히 구현을 완성하는 것만으로는 **완성도 높은 프로젝트**를 만들기 어렵다는 점을 깨달았다. 구현 후 **결과를 세밀하게 분석하고, 성능에 대한 정량적 평가**를 통해 문제점을 개선하는 시간이 반드시 필요하다는 것을 느꼈다.

  다음 프로젝트에서는 **Retriever와 Generator 각각의 성능을 정량적으로 평가할 수 있는 지표**를 도입하고, 검색 결과와 문제 풀이 정확도의 관계를 구체적으로 분석하고 싶다. 또한, 프로젝트 초기부터 작업 분배와 일정 관리를 명확히 설정하여 효율적인 협업을 이루고자 한다.

서태영

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  EDA를 통해 데이터의 전체적인 구성, 흐름을 확인을 해봤다. 그러고 난 후 LLM이다 보니 프롬프트를 다르게 주는 것도 의미가 있을 것 같다는 생각에 다르게 줘봤지만 큰 차이가 없어 큰 모델이 필요하다고 생각하여 큰 모델을 GPU 서버에서 돌리기 위해 돌려봤다. 4비트 양자화를 통해 돌리기 성공한 모델이 12B의 모델이고, 14B 모델을 시도해봤지만 메모리 초과 에러가 떴다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  gpu에 큰 모델을 돌리기 위해 양자화라는 것을 해봤고, 그냥 LoRA를 붙이는 것이 아닌 QLoRA를 붙이면 메모리를 아낄 수 있어 QLoRA에 대해서도 알게 된 것 같다. 사실 LoRA에 대해서도 잘 알지는 못했지만 이 과정에서 전에 보다는 알게 된 것 같다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  14B 모델을 양자화해서 돌리지 못했다. 못돌려도 조금은 더 시도를 해보던가, 다른 라이브러리를 찾을 생각을 했으면 좋지 않았을까 싶다. 결과론적으로는 다른 팀들 대부분이 라이브러리를 써서 큰 모델을 돌렸다고 한다. 그래서 조금 더 시도를 해보지 못한게 아쉽다.

  데이터를 충분히 분석하지 않았던 부분이 아쉽다. 모델 돌려보기에 급급해서 데이터는 간단하게 보고 넘어 갔다. 결국 그렇게 대충 보고 넘어간게 나중에 모델을 돌려보면서 데이터에 대한 이해가 부족했다는 생각이 계속 들었던 것 같다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  큰 모델을 한정된 gpu에서 돌리는 것을 시도해보고 싶다. 한정된 gpu에서 큰 모델을 한번 정도는 돌려보는 게 좋은 경험일 것 같다. 또한 RAG를 해보고 싶다. RAG가 궁금했는데 어떻게 해야할지 몰랐다. 그런데 다른 팀원이 하는 것을 보고 어떻게 해야할지 알게 된 게 있어서 RAG를 해보고 싶다.

이재백

- **나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?**

  EDA를 통해 데이터의 분포를 확인하고 데이터의 편향을 확인할 수 있었다. 데이터 전처리를 통해 데이터셋의 라벨 분포 균형을 맞춰주었고, 4개인 문제에 대해 프롬프트를 변경해주었다(5개중 하나를 고르시오 → 4개중 하나를 고르시오).
  Zero-shot-CoT 연구 논문을 읽어보고 우리의 프로젝트에 적용하고자 했고, 더 어려운 문제를 모델이 풀도록 하면 비

교적 쉬운 문제에 대해 성능이 올라갈 것이라는 판단 하에 PSAT 언어논리 영역의 문제들을 파싱해서 데이터 증강에 활용하였다.

- **전과 비교해서, 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?**

  GitHub discussion을 이용해 팀원들과 실험 진행, 문제 해결에 대한 정보 공유가 용이했다. 근거에 기반한 의사결정을 하고자 하였고, 이를 위해 논문을 읽고 논문 리뷰를 GitHub discussion을 통해 공유했다.

- **마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?**

  Public score에 과도한 의존

  진행한 실험과 적용한 방법론의 평가를 제출한 결과물의 Public score에만 과도하게 의존하였다. 실제로 데이터 증강의 효과를 확인하기위해 Public score를 확인하고, 결과적으로 같은 점수를 얻어 데이터 증강을 최종적으로 적용하지 않았지만 대회 종료 후 Private score는 상승한 것으로 확인 하게 되었다.

  PM의 부재

  프로젝트의 진행이 계획 없이 팀원 각자의 흥미와 계획에 따라 프로젝트가 진행되었다. 서로의 진행도를 체크할 수는 있었지만 프로젝트가 제각각으로 진행되다보니 통일성이 부족했고 마감 기한도 설정하지 않아 진행도가 지지부진했던 것 같다.

  데이터 분석의 한계

  데이터 분석을 더 자세히 하지 못해 아쉬웠던 것 같다. 단순히 문제의 길이 분포나 라벨 분포 외에도 문제 유형 분석, 이상 데이터 탐지 등 여러가지를 시도해 볼 수 있었던 것 같다.

- **한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?**

  자체적인 Evaluation dataset 구축

  Public score에 매몰되지 않도록 우리의 모델과 데이터를 평가할 수 있는 자체적인 기준을 설정한다면 좀 더 객관적인 성능평가가 가능할 것 같다.

  PM 정하기 & 철저한 일정 관리

  PM을 정하고 동시에 철저한 일정 관리를 동반해야할 것 같다.

Reference

- Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in neural information processing systems, 35, 22199-22213.