수능형 문제풀이 모델 Wrap-Up Report

1. 프로젝트 개요 및 구성

1-1. 개요

(1) 작성자: nlp-03

(2) 기간: 2025.12.17 10:00 ~ 2026.01.06 19:00

(3) 최종 리더보드 순위: 2위(0.8015)

Rank	Team Name	Team Member	MACRO_F1	Entries	Final
My Rank 2	NLP_03조		0.8015	68	21h

본 프로젝트는 최근 대규모 언어 모델(LLM)이 다양한 시험 환경에서 높은 성과를 보이는 흐름 속에서, 상대적으로 규모가 작은 모델의 가능성을 검증하기 위해 수행되었다. GPT, Claude, Gemini와 같은 대형 모델들은 사법고시, 의사 시험뿐 아니라 한국의 대학수학능력시험(수능)에서도 우수한 성적을 기록하고 있으나, 이러한 성능은 대규모 파라미터와 연산 자원에 크게 의존한다는 특징을 가진다. 이에 본 프로젝트는 대형 모델과 대비되는 소형 언어 모델(sLLM)을 대상으로, 특정 도메인에 집중했을 때 어느 수준까지 성능을 끌어올릴 수 있는지를 살펴보고자 한다.

1-2. 개발 환경

개발환경	Ubuntu / Python 3.10 / PyTorch / Transformers / scikit-learn / Unsloth / llama.cpp
GPU	Tesla V100-SXM2-32GB
관리	VSCode Remote SSH, W&B 실험 로그 관리, VPN 사용
모델 허브	Hugging Face Transformers

1-3. 프로젝트 팀 구성 및 역할

팀원	역할
김윤희	데이터 EDA 및 증강, Unsloth 기반 모델링, gguf 포맷 기반 fewshot inference, Agentic RAG 구성 및 적용
박서진	Baseline Code 분석, Unsloth 기반 모델 학습 및 성능 분석, CoT 프롬프트 엔지니어링, RAG 와 추론 모델 통합
곽나영	Baseline Code 분석 및 코드 모듈화, 데이터 수집 및 증강, 학습 데이터 임베딩 분석, Unsloth 기반 모델링, 앙상블
김이슬	Baseline Code 분석, streamlit 시각화, 모델 학습 및 성능 분석, RAG 구현 및 성능평가, Agentic RAG 구성 및 적용
우혜진	데이터 EDA 및 증강, Few Shot 프롬프팅, gguf inference, 앙상블
최준호	Agentic RAG 구성 및 적용, 프롬프트 엔지니어링, RAG 데이터 구성

2. 데이터 및 베이스라인 코드 분석

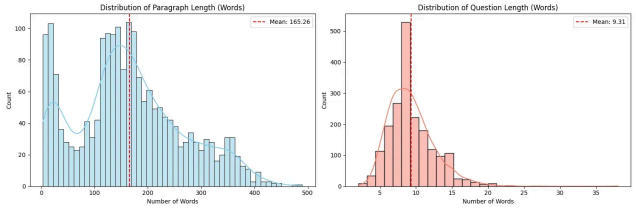
2-1. 데이터 분석

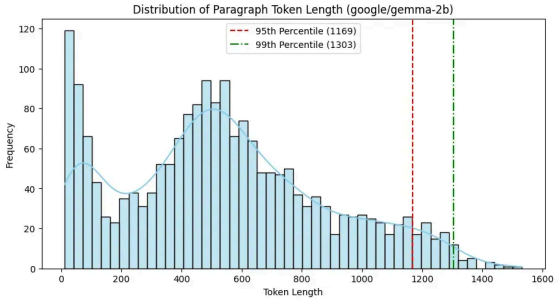
2-1-1. 데이터 구조

분류	개수
train	2,031 개
test	869 개

본 대회 데이터는 수능형 객관식 문제 형태로, `{ id / paragraph / problems / question_plus }`로 구성된다. 핵심 입력은 지문(paragraph), 질문(question), 선지(choices)이며, problems 컬럼 내 JSON을 파싱해 question, choices, answer 를 분리하여 학습용 타깃을 구성했다.

2-1-2. 길이 및 토큰 수 분석



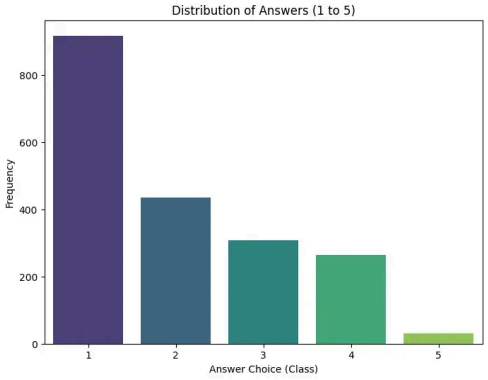


통계치	토큰 수	해석
평균 (Mean)	537 토큰	지문 하나의 평균 길이는 537 토큰
최소 (Min)	11 토큰	매우 짧은 지문도 존재
최대 (Max)	1,532 토큰	가장 긴 지문은 1,532 토큰으로 매우 김
P95 (95% 커버)	1,169 토큰	전체 데이터의 95%를 포함하려면 최소 1,169 토큰의 Context Window가 필요.
P99 (99% 커버)	1,303 토큰	99%의 데이터를 포함하려면 최소 1,303 토큰의 Context Window가 필요.

위 그림은 paragraph와 question의 단어 수와 google/gemma-2b의 tokenizer를 사용한 토큰 수를 나타낸 것이다. 이를통해 데이터 길이

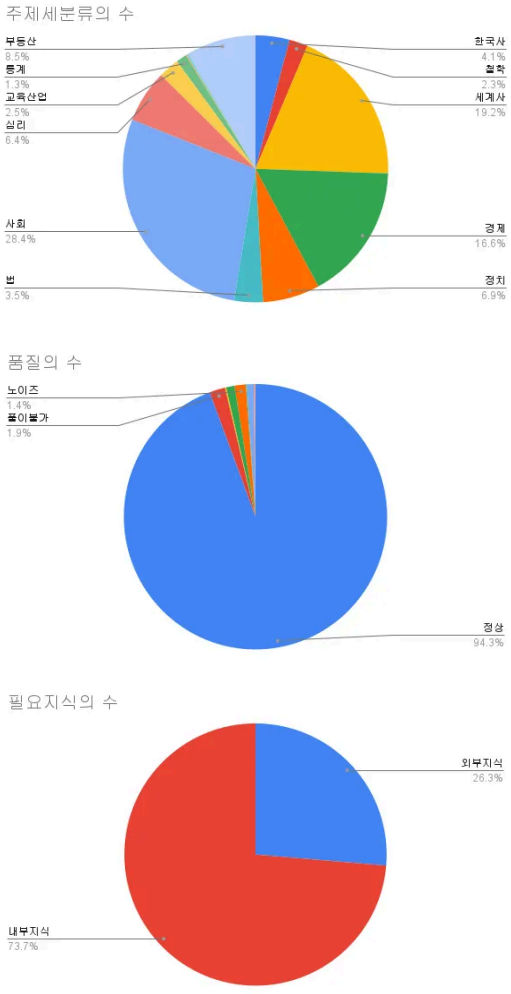
변동성이 매우 큼을 인지했고, 모델의 context window 설정이 중요함을 알았다. 지문의 길이가 긴 데이터는 RAG(검색 증강) 기법을 사용하여 관련 문단만 모델에게 전달하는 전략을 고려해 볼 수 있었다.

2-1-3. 정답 분포 해석



1 번 정답이 5 번 정답의 18 배나 많음을 알 수 있었으며, 5 번 클래스를 단 하나라도 놓치거나(FN) 잘못 예측(FP) 일 경우, 해당 클래스는 F1-score가 0에 가까워져 최종 Macro F1-score를 크게 하락시킬 것을 예상할 수 있었다. 또한 모델이 단순히 1 번을 찍도록 가까워지게 될 가능성이 있어 전략적인 방법이 필요함을 깨달았다.

2-1-4. 학습 데이터 분석



Train 데이터셋 분석을 통해 각 데이터의 세부 주제 분포, 정답 품질, 필요 지식을 파악했다.

① 세부 주제는 사회, 세계사, 경제, 부동산, 정치, 심리, 한국사, 법, 교육산업, 철학, 통계, 문학, 문법으로 나누어 분류 해보았는데, 사회·세계사·경제가 전체의 약 64.2%로 높은 비율을 차지하고 있었다. 반면 문학·문법 문제는 전체의 0.35%로 매우 적은 비율이었다.
② 정답 품질은 대부분이 문제에 알맞은 정답으로 잘 구성되어 있었지만(94.3%), 3.1%의 데이터는 선택지에 `\xa0`와 같은 특수 기호가 포함되거나 복수정답, 정답 오류 등 저품질 데이터가 섞여 있었다.
③ 필요 지식은 지문만으로 정답을 도출할 수 있는 내부지식이 73.7%, 지문 외 배경지식이 추가로 요구되는 외부지식이 26.3%의 비율을 보였다.

2-2. 데이터 증강

2-2-1. 데이터 추가 수집 및 제작

이 태스크가 수능형, 특히 국어·사회탐구 분야의 문제풀이 모델인 만큼, 가장 데이터가 부족한 문학·문법, 그리고 Trainset에는 없는 화법과 작문 데이터의 추가 수집을 우선적으로 진행했다. 뿐만 아니라 전체 비율이 적은 한국사, 정치, 법 주제를 위주로 데이터를 수집했다.

허깅페이스에 업로드된 외국어 데이터셋도 GPT 증강을 통해 학습 데이터셋으로 제작했다. SAT나 LSAT와 영어로 구성되어 있지만, 같은 수능 문제 같이 사고력을 요구하는 데이터셋을 번역 및 수능형 지문으로 문제를 제작하였다.

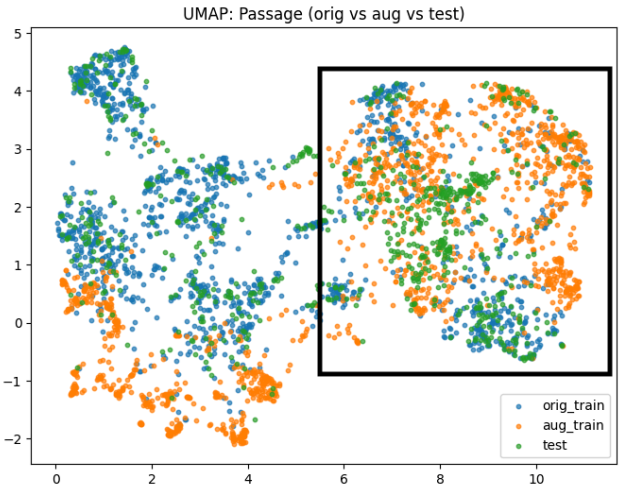
세부 주제	데이터 수집처
문학, 문법, 화작	국가직·지방직 9급 국어, AI-Hub 국어 교과 지문형
비문학	LSAT-RC, PSAT 언어논리, AI-Hub 금융, 법률 문서 기계독해 데이터
한국사, 세계사	국가직·지방직·지역인재 9급, 한국사능력검정시험, SAT-History
정치	국가직 7급, 9급 국제정치학
법	국가직 7급, 9급 국제법

기존 학습 데이터 2,031개를 기준으로 단계적인 데이터 증강을 총 4차에 걸쳐 수행하였으며, 이를 통해 최종적으로 7,065개 규모의 학습 데이터셋을 구축했다.

그러나 4차 증강 데이터(7,565개)를 사용했을 때보다 3차 증강 데이터(4,067개)를 사용하여 학습했을 때 훈련 수렴이 더 안정적이고 성능 지표 또한 우수하게 나타났다. 이에 추가적인 데이터 증강이 오히려 노이즈를 증가시킬 수 있다고 판단하여 이후 증강 단계는 진행하지 않았다.

2-2-2. 데이터 임베딩

증강한 데이터가 주어진 Testset을 잘 커버하는지 시각적으로 확인해보기 위해 `BAAI/bge-m3` 모델로 간단히 임베딩 작업을 진행하였다. Trainset과 중복된 데이터가 63개가 있어 해당 데이터는 분석에서 제외하였다.



UMAP 방식으로 데이터 임베딩 그래프를 그려본 결과 오른쪽 영역에서 augment된 데이터가 Testset 분포를 일부 채워주며, 기존 Trainset 대비 Testset 커버리지가 개선되는 경향을 확인했다.

2-2-3. Validation Dataset 구성

private 리더보드 점수를 안정적으로 유지하고 모델의 일반화 성능을 신뢰성 있게 판단하기 위해, Public 리더보드 점수에만 의존하지 않고 별도의 Validation 데이터셋을 고정적으로 두었다.

① 초기에는 Testset의 문제 유형 분포를 추정하여 Validation의 분포를 이에 맞추는 방안을 시도하였다. 그러나 철학·한국사와 같이 표본 수가 적은 유형이 존재하고, 화법과 작문처럼 Test에는 존재하지만 학습 데이터에는 거의 포함되지 않은 유형도 확인되었다.

② 이후 증강된 train 데이터에서 Validation을 분리하는 방안도 검토하였으나, 증강 데이터가 포함될 경우 Validation 성능이 데이터 증강 전략의 영향을 직접적으로 받게 되어, 모델 자체의 성능 차이를 명확히 비교하기 어렵다고 판단하였다. 이에 Validation 셋은 가능한 한 데이터 생성 과정의 영향을 받지 않는 original train을 기준으로 구성하는 것이 바람직하다고 결론지었다.

③ 멘토님의 조언을 반영하여, 특정 가설이나 분포 추정을 인위적으로 반영하기보다는 편향을 최소화하는 것이 오히려 장기적으로 안정적이라는 결론에 도달하였고, 최종적으로 original train 데이터에서 20%를 무작위로 추출하여 Validation을 구성하고 대회 기간 동안 고정하여 사용하였다.

대회 막바지에는 모델 구조와 학습 전략이 충분히 검증되었다고 판단하여, 최종 성능 극대화를 위해 해당 Validation 데이터까지 포함한 전체 데이터를 사용하여 학습을 진행했다.

2-3. 베이스라인 코드 분석

2-3-1. 데이터 구성 및 입력 설계

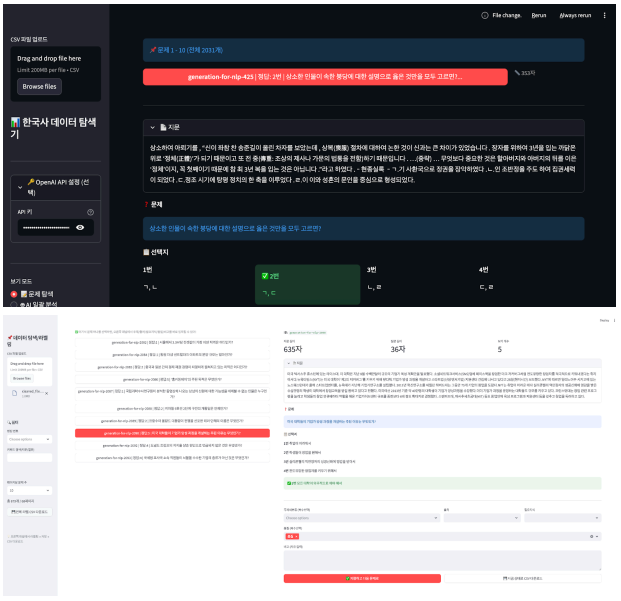
베이스라인 코드는 JSON 형태로 제공된 원본 데이터를 지문, 질문, 선택지, 정답 구조로 평탄화하여 학습에 활용한다. 이때, 데이터에는 <보기> 가 포함되지 않는 경우와 <보기>가 포함된 경우가 존재한다. 이러한 데이터 구성에 따라 서로 다른 프롬프트를 사용하여 문제를 모델에 전달할 때 형식을 충실히 반영하려는 의도를 가지고 있다.

2-3-2. 학습 및 평가 방식

모델 학습에는 `beomi/gemma-ko-2b` 모델을 사용하여 LoRA 기반 PEFT 방식이 적용되었다. 또한 입력 길이를 1024 토큰으로 제한하여 메모리 사용을 제어하고 있으며, 이로인해 긴 지문을 포함한 일부 문제는 학습 대상에서 제외된다. 평가 및 추론 단계에서는 문장 생성을 직접 사용하지 않고, 다음 토큰에 대한 출력 분포를 비교하여 정답 번호를 선택하는 logit 방식이 사용된다.

해당 방식은 제한된 자원 환경에서 효율적으로 동작하지만, 작은 모델 규모와 입력 길이 제한으로 인해 성능 향상에는 한계가 있었다. 이를 보완하기 위해 이후 단계에서는 학습 전략을 개선하여 모델 성능을 추가로 향상시키고자 하였다.

2-3-3. 데이터 관찰용 도구 개발



초기 데이터 라벨링은 구글 시트를 활용해 진행하였으나, 데이터가 많아질수록 문제와 선택지를 함께 비교하기 어렵고 라벨링 기준을 유지하는 데 한계가 있었다. 이를 개선하기 위해 Streamlit 기반의 데이터 탐색·라벨링 도구를 구축하였으며, CSV 업로드 후 직관적으로 데이터를 확인하고 라벨링할 수 있도록 구성하였다. 또한 OpenAI API를 연동해 모델의 분류 결과를 함께 확인함으로써, 수작업 라벨링과 모델 예측을 비교하며 데이터 특성을 점검할 수 있었다.

3. 모델 개선

3-1. Unsloth 기반 모델 학습 및 성능 분석

3-1-1. 대규모 모델 도입 및 베이스라인 성능 비교

베이스라인으로 사용한 `beomi/gemma-ko-2b` 모델은 모델 규모가 작아 수능형 문제를 해결하기에는 추론 능력에 한계가 있다고 판단하였다. 이에 따라 더 큰 모델을 활용하기로 결정하였으며 개발 환경 내에서 큰 모델을 돌리기 위해서 Unsloth을 적용하여 30B 이상의 양자화된 대규모 언어 모델을 사용하는 방향으로 설정하였다.

또한 한국어 이해 및 생성 성능이 우수하다고 판단되는 Qwen 계열 모델을 선택하여, LoRA 기반 파인튜닝을 진행하였다.
베이스라인 모델과 Qwen 계열 모델의 추론 결과는 다음과 같다.

	public score	private score
beomi/gemma-ko-2b	0.2523	0.2788
unsloth/Qwen2.5-32B-Instruct-bnb-4bit	0.6444	0.5759

`beomi/gemma-ko-2b` 모델은 public score와 private score이 전반적으로 낮은 결과를 보인 반면, `unsloth/Qwen2.5-32B-Ins`

truct-bnb-4bit 모델은 public 및 private score 모두 크게 향상된 성능을 기록하였다. 이를 통해 모델 규모에 따라서 수능형 문제 해결에 유의미한 영향을 미친다는 것을 확인할 수 있었다.

3-1-2. 추론 방식 비교

출력 방식에 대한 실험 과정에서 generate 방식으로 추론을 수행한 결과, 추론 중 토큰을 순차적으로 생성하는 과정에서 정답 토큰이 생성되기 전에 출력이 조기에 종료되는 문제가 발생하였다. 이는 모델이 최종 정답에 도달하기 전에 토큰 생성이 끊기는 현상이 빈번하게 관찰되어 정답 번호를 추출하지 못하는 현상이 빈번하게 관찰되었다.

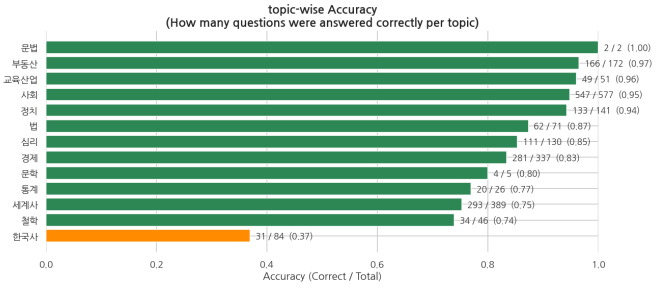
이러한 문제를 보완하기 위해 문장 생성을 직접 수행하는 generate 방식 대신, 모델의 출력 분포를 활용하여 정답을 추론하는 logit 기반 방식으로 추론 방법을 변경하였다.

	public score	private score
generate 방식	0.6444	0.5759
logit 방식	0.7501	0.6910

동일한 모델 unsloth/Qwen2.5-32B-Instruct-bnb-4bit 모델을 LoRA 방식으로 파인튜닝한 후 두 추론 방식을 비교한 결과, logit 방식이 generate 방식 대비 public score와 private score 모두에서 약 0.1 이상 높은 성능을 기록하였다. 이는 logit 방식이 생성 길이에 대한 제약을 받지 않아 정답을 안정적으로 제공하기 때문인 것으로 해석된다.

3-1-3. 주제별 성능 분석

성능을 추가로 향상시키기 위해, 모델이 어떤 유형의 문제에서 정답을 정확하게 추론하지 못하는지를 분석하는 과정을 수행하였다. 이를 위해 파인튜닝이 적용되지 않은 기본 모델을 사용하여, 학습 데이터를 기반으로 추론 결과를 분석하였다. 해당 분석을 통해 모델이 상대적으로 취약한 문제 주제 영역을 식별하고, 이후 성능 개선 방향을 도출하고자 하였다.



모델의 추론 결과를 분석한 결과, 문제 주제별로 성능 편차가 뚜렷하게 나타남을 확인할 수 있었다. 특히, 한국사 주제에서는 정확도가 약 0.37로 현저히 낮게 나타났다.

한국사 주제에서 상대적으로 낮은 성능이 나타난 원인으로, 모델의 지역적 편향의 가능성을 고려할 수 있다. 본 실험에서 사용한 Qwen 계열 모델은 중국 모델이기 때문에 한국사와 같은 특정 국가에 강하게 의존하는 문제 영역에서는 상대적으로 불리하게 작용했을 가능성을 시사한다.

3-1-4. Continued Pre-Training

앞의 분석을 바탕으로, 모델의 취약 영역을 보완하기 위한 방법으로 데이터 증강을 통해 구축한 학습 데이터를 활용하여 Unsloth 환경에서 Continued Pre-Training(CPT)을 수행하였다.

일반적으로 LoRA를 포함한 PEFT 기법은, 상대적으로 매우 적은 수의 추가 파라미터만을 학습하는 방식이다. 이러한 특성은 모델 출력을 특정 형식에 맞게 조정하는 데에는 효과적이지만, 주제 자체에 대한 배경 지식이나 도메인 지식을 모델 내부에 새롭게 학습시키는 데에는 구조적인 한계가 존재한다.

주제별 분석을 통해 확인된 지식 기반 취약 영역을 보완하기 위해, 모델 파라미터를 재학습하는 Continued Pre-Training 방식을 적용하였다. CPT는 증강된 학습 데이터를 통해 기존 모델에 새로운 지식을 재주입함으로써, 특정 주제에 대한 이해 부족으로 인해 발생한 성능 저하를 완화하는 것을 목표로 수행하였다.

	public score	private score
LoRA 없이 추론	0.7488	0.7113
데이터 + LoRA	0.7501	0.6910
증강 데이터 + LoRA	0.7501	0.6910
증강 + CPT 적용	0.7962	0.7616

LoRA 파인튜닝은 데이터 증강 여부와 관계없이 성능 개선 폭이 제한적인 반면, Continued Pre-Training을 적용한 경우 가장 큰 성능 향상을 보였다.

결과적으로, 충분한 학습 데이터가 있는 상황에서 PEFT 방식만으로 한계가 있으며, 모델에 새로운 지식을 반영하는 Continued Pre-Training이 지식 기반 취약 영역을 보완하는 데 효과적인 접근임을 확인할 수 있었다.

3-2. Agentic RAG

베이스라인으로 활용했던 모델은 한국사와 철학과 같은 외부 지식을 요하는 문제에 대해 한계를 보였다. 다음은 이를 Agentic RAG로 발전시키기 위한 아이디어 흐름도이다.

Vanilla RAG는 외부 지식을 Vector Store에서 검색하되, 문제 유형(참조 문제/보기 문제/일반 문제)을 자동 분류하고 각 유형에 맞는 쿼리 전략을 적용하는 솔루션이다.

Agentic RAG는 이를 더 발전시켜 question에 비해 너무 긴 paragraph에 embedding distance가 의존되는 현상을 해결하기 위해 Query 생성, Query Validation, Reasoning 등을 접목한 솔루션이다.

3-2-1. 구성요소

3-2-1-1. 데이터 구성

세부 주제	데이터 수집처	
한국사	우리역사넷	HTML to json
문학	한국민족문화대백과사전	자체 API 활용
경제	경제학개론	PDF Parser
철학	고등학교 철학 교과서	PDF Parser
다방면 논문 데이터	AI Hub 학술 논문 이해 데이터	

3-2-1-2. Embedding

전처리된 chunk들을 BGE-M3 모델을 사용하여 1024차원의 dense vector로 변환하였다. BGE-M3는 다국어 및 한국어 텍스트의 semantic representation에 우수한 성능을 보이는 모델로, CUDA를 활용하여 대량의 문서를 효율적으로 임베딩 할 수 있었다. Langchain의

HuggingFaceEmbeddings 사용 및 L2 normalization을 적용해 cosine similarity 계산을 최적화하였다.

3-2-1-3. Vector DB

대규모 데이터의 검색 효율성을 위해 FAISS를 선택하였다. FAISS는 GPU 가속을 지원하여 실시간 검색이 가능하며, 메모리 효율적인 인덱싱 구조를 제공한다는 장점이 있다. 특히 시간적 제약이 있는 경진대회 환경에서 빠른 실험 iteration을 가능하게 했다.

데이터의 특성을 고려하여 주제별로 독립적인 FAISS index를 구축하였다. 총 7개의 주제별 컬렉션을 생성하였으며, 각 컬렉션의 구성은 다음과 같다.

	dim	vectors
economics	1024	166
humanities	1024	88,490
korean-history	1024	8,820
literature	1024	12,378
philosophy	1024	252
science-tech	1024	39,264
social-science	1024	108,102

주제별 컬렉션 분리는 검색 정확도 향상과 계산 효율성 측면에서 이점이 있었다. 각 문제에 대해 해당 주제의 컬렉션에서만 검색을 수행함으로써 irrelevant한 타 주제 문서가 검색되는 것을 방지하고, 검색 대상 벡터 수를 줄여 검색 속도를 향상시켰다. 또한 문서별 고유 ID(`{file_tag}_{doc_id}_{chunk_id}`)를 부여하여 중복을 방지하고, 메타데이터(subject, title, doc_id, chunk_id)를 함께 저장하여 검색 결과 추적성을 확보하였다.

3-2-2. Vanilla RAG

3-2-2-1. 초기 접근: Naive RAG

가장 기본적인 RAG 방식으로 시작하였다. `paragraph + question + choices`를 쿼리로 vector store에서 top-5 문서를 검색하여 프롬프트에 추가하는 방식이었다.

	RAG 미적용	기본 RAG 적용
MACRO_F1	0.6977	0.5872

Naive RAG를 적용한 결과, 오히려 MACRO_F1 점수가 약 11%p 하락하는 역효과가 발생하였다.

검색 결과를 직접 분석한 결과, 임베딩 기반 검색이 검색 의도를 제대로 파악하지 못한다는 근본적인 문제를 발견하였다. 예를 들어 "수로왕이 건설한 (가) 국가는?"이라는 문제의 경우, vector DB는 "수로왕", "김수로왕" 관련 문서는 검색하지만, 정작 필요한 "김수로왕이 건설한 국가"라는 관계 정보는 가져오지 못하였다. Embedding model이 인물명에만 집중하고 "건설한 국가"라는 관계성을 이해하지 못했기 때문이다. 더욱이 검색된 문서 중 일부는 문제 풀이에 혼란을 주는 잘못된 정보를 포함하고 있어, 모델의 추론을 방해하는 노이즈로 작용하였다.

3-2-2-2. Query Decomposition 기반 다단계 검색

이를 해결하기 위해 문제 유형을 자동 분류하고 각 유형에 맞는 검색 전략을 적용하는 시스템을 구현하였다. 패턴 기반 분류를 통해 참조 문제, 보기 문제, 일반 문제를 구분하였다.

1. 지문+질문으로 1차 검색하여 문제에서 찾아야 하는 대상 검색
2. 각 보기 항목에 대한 증거 자료 개별 검색
3. 각 선택지에 대한 증거 자료 개별 검색
4. 모든 검색 결과를 통합하여 모델에 전달

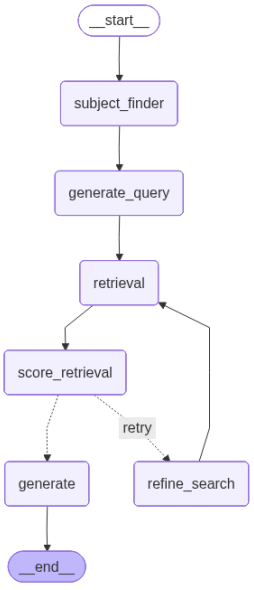
다단계 검색 전략은 일부 문제에서 개선된 결과를 보였다. 예를 들어 "직지심체요절이 간행된 왕대"를 묻는 문제의 경우 키워드가 명확하고 시기 정보가 직접적으로 포함되어 검색이 성공적이었다. 그러나 복잡한 관계 추론이 필요한 문제에서는 여전히 실패하였다. "수로왕이 건설한 국가" 문제에 대해 모든 보기에 대해 동일한 "김수로왕" 문서만 반복적으로 검색되었다. 각 보기의 핵심 개념("해상교역", "경당", "정사암회의")이 아닌 지문의 "수로왕"이라는 키워드에만 반응한 것이다. 검색 결과는 늘어났지만, 실질적으로 도움이 되는 새로운 정보는 추가되지 않았다.

성능 개선이 제한적인 이유를 분석하기 위해 검색 결과를 심층 검토하였다. 주요 한계점은 다음과 같이 정리되었다.

① 가장 근본적인 문제는 임베딩 검색만으로는 "무엇을 찾아야 하는지"를 정확히 표현할 수 없다는 것이었다. 단계별 검색으로 개선되었지만, 위의 예시처럼 각 검색에서 여전히 관계성 파악 실패가 발생하였다.
② 고정된 검색 흐름의 한계가 존재하였다. 직접적인 정보가 있는 문제는 잘 해결했지만, 다단계 추론이 필요한 문제는 여전히 실패하였다.
③ 일회성 검색으로 인해 검색 결과가 부족하거나 잘못되어도 재검색할 방법이 없었다. 검색된 문서가 실제로 문제 해결에 도움이 되는지 평가할 메커니즘이 없어, 유사도는 높지만 무관한 정보가 노이즈로 작용하였다.

앞의 분석을 바탕으로, 고정된 규칙이 아닌 상황에 따라 적응하는 검색 시스템을 구축하고자 Agentic RAG로의 전환을 결정하였다.

3-2-3. Agentic RAG



다음의 구조로 Agentic RAG를 설계하였다.

3-2-3-1. Subject Finder

이후 모든 시퀀스에서 모델이 해당 입력의 과목 도메인을 특정화하여 일관성을 높이기 위한 노드이다.

3-2-3-2. Generate Query

Vector Store에 검색을 하기 위한 질의를 생성하는 노드이다. Hyde 논문을 참고하였으며 Paragraph의 핵심 키워드-질문 쌍에 대한 질의,

개별 선택지-질문 쌍에 대한 질의를 생성하여 인간이 문제를 푸는 형식과 유사하게 구성하고자 하였다.

3-2-3-3. Score Retrieval

score_retrieval 노드는 RAG 검색 결과(Chunk)가 실제 문제 해결에 도움이 되는지를 정량화하여 평가하는 단계이다. 먼저 각 (query, chunk) 쌍에 대해 LLM 기반 유효성 평가를 진행하여, 해당 chunk가 단순히 질의와 유사한 표현을 포함하는지를 넘어 문제에서 요구하는 핵심 정보와 실제로 정합적인지를 점수화한다. 예를 들어 “세종대왕의 업적”이라는 질문에 대해 “세종대학교 설립 연도”와 같은 chunk는 벡터 유사도는 높을 수 있으나 의미적으로 부적절하므로 낮은 유효성 점수를 받도록 설계된다. 이후 LLM이 산출한 유효성 점수를 0~1 범위로 정규화한 값과 벡터 유사도를 가중 결합하여 최종 점수(final)를 계산함으로써, 유사도는 높지만 내용이 부정확한 검색 결과를 효과적으로 필터링한다. 다음으로 여러 query를 사용한 검색 결과를 평가하기 위해 query별 최고 점수만을 취합하고, 이를 평균 내어 전체 검색 품질을 나타내는 avg_score를 산출한다. 이 평균 점수가 사전에 설정된 임계값(threshold)보다 낮을 경우, 현재 검색 결과가 불충분하다고 판단하여 재검색이 필요하다는 신호를 생성하며, 무한 반복을 방지하기 위해 반복 횟수 제한을 함께 적용한다. 이러한 절차를 통해 본 노드는 검색 결과의 신뢰도를 체계적으로 평가하고, 필요 시 다음 단계의 검색 전략 수정으로 자연스럽게 연결되도록 구성된다.

3-2-3-4. Refine Search

refine search 노드는 score retrieval 단계에서 산출된 검색 품질 평가 결과를 바탕으로, 현재의 검색 전략이 충분하지 않다고 판단될 경우 검색을 개선하기 위한 역할을 수행한다. 구체적으로 avg_score가 임계값(threshold)보다 낮고 반복 횟수가 허용 범위 내에 있을 때, 기존 query들이 문제의 핵심을 제대로 포착하지 못했다고 가정하고 LLM을 활용해 query 자체를 재구성한다. 이 과정에서 LLM은 문제의 과목(subject), 지문, 질문, 선택지, 그리고 기존 query들을 종합적으로 고려하여 보다 핵심 개념 중심의 새로운 query 집합을 생성한다. 동시에 검색 범위를 확장하기 위해 top-k 값을 점진적으로 증가시키고, 반복이 진행될수록 threshold를 완화하여 지나치게 엄격한 기준으로 인해 검색이 계속 실패하는 상황을 방지한다. 이러한 점진적 완화 전략을 통해 초기에는 정밀한 검색을 시도하되, 난도가 높은 문제일수록 더 넓은 정보 공간을 탐색할 수 있도록 설계된다. 결과적으로 refine search 노드는 검색 실패를 단순 종료로 처리하지 않고, 검색 전략을 동적으로 조정하여 보다 적절한 근거 정보를 확보하도록 유도하는 역할을 수행한다.

3-2-3-5. Retrieval

Vanilla RAG의 RAG 시스템을 활용하여 개별 질의에 대해 Vector Store에서 검색한 후 답변을 생성하는 노드이다.

3-2-3-6. Generate

이전 Node에서 파악한 지식, 입력에 대한 지식을 모두 활용하여 최종 답변을 생성하는 노드이다. 이 때, 해당 입력과 유사도를 판별하여 예시 데이터 추가로 제공하는 Few-shot, 출력 시 추론 과정을 먼저 자연어로 생성하고 답변을 최종 생성하는 과정을 포함한다.

3-2-3-7. 결과(private score)

	RAG 미적용	agentic RAG 적용
MACRO_F1	0.8008	0.8015

3-2-4. 프롬프트 엔지니어링

단순히 모델에 입력으로 들어가는 프롬프트를 수정하는 것으로 성능을 향상시키는 기법이 프롬프트 엔지니어링이다. 다음은 대회에서 적용했던 주요 프롬프트 엔지니어링 기법이다.

3-2-4-1. Constructed Output

답변의 형식을 교정하는 방법론이다. Sequence to Sequence로 답변을 생성할 때, 매 질문마다 답변 형식을 강제하여 필요한 정보만을 얻기 위한 방법론이다.

3-2-4-2. Few-shot Prompting

LLM은 모델의 크기가 늘어날 수록 예시 데이터를 넣고 안 넣고에 대한 차이가 크다. 이에 착안하여 **BGE-M3 임베딩 기반의 유사도 검색(retrieval)** 을 기준으로, 현재 입력과 가장 유사한 학습 문제를 few-shot 예시로 제공해 추론 성능을 높이는 방법을 적용했다.

① 먼저 `BAAI/bge-m3` 임베딩 모델을 사용해 학습 데이터와 테스트 데이터의 **제시문(paragraph)**, **질문(question)**, **선지(choices)** 를 각각 임베딩하였다. 이때 임베딩 목적에 따라 prefix를 구분하여 적용하고, 코사인 유사도를 효율적으로 계산하기 위해 L2 정규화 후 dot product 방식을 사용하였다.

② 유사 예시의 1차 후보군을 구성하기 위해, 테스트 문제의 질문 임베딩과 학습 데이터의 질문 임베딩 간 유사도를 계산하고 상위 Top-K(50)개의 후보를 선정하였다. 이를 통해 전체 학습 데이터가 아닌, 의미적으로 충분히 가까운 문제들만을 대상으로 Few-shot 선정을 진행하였다.

③ 선정된 후보군 내에서 서로 다른 관점의 정보를 제공하기 위해 3개의 샷을 분리하여 선택했다. 이를 통해 한 문제에 대해 지문 중심, 질문 중심, 선택지 중심의 추론 단서를 동시에 제공할 수 있도록 했다.

- **Shot 1**:Paragraph 임베딩 유사도가 가장 높은 예시
- **Shot 2**: Question 임베딩 유사도가 가장 높은 예시
- **Shot 3**: Choices 임베딩 유사도가 가장 높은 예시

④ 샷 간 중복과 과도한 유사도를 방지하기 위해, 이미 선택된 샷과의 임베딩 유사도가 일정 기준 이상일 경우 해당 후보를 제외하는 diversity threshold를 적용하여 예시 간 다양성을 확보했다.

3-2-4-3. Reasoning Inference

답변만 단순하게 출력하는 대신, 모델의 추론 과정을 함께 생성하게 함으로써 복잡한 문제들에 대한 사고흐름을 순차적으로 이끌어내는 방법론이다.

3-3. GGUF

3-3-1. GGUF 기반 추론 실험

대형 언어 모델의 추론 성능을 최대한 활용하기 위해, unsloth에서 제공하는 GGUF 포맷의 `Qwen3-Next-80B-A3B-Thinking` 모델을 활용하여 llama.cpp 기반 서버 환경에서 추론 실험을 수행하였다. 본 실험은 RAG나 agent 구조를 결합하지 않고, 대형 모델 자체의 추론 능력과 few-shot 일반화 성능의 한계를 확인하기 위한 기준선(baseline) 설정을 목적으로 진행되었다. 이후 agent 기반 RAG 구조와의 성능 비교를 위한 중요한 실험으로 활용되었다.

3-3-2. 사전 토큰 예산 검증 및 입력 관리 전략

실제 추론에 앞서, 입력 길이 초과로 인한 추론 실패를 방지하기 위해 사전 토큰 예산 검증(Token Budget Sweep)을 수행하였다. 컨텍스트 길이, 출력 토큰 수, few-shot 비율, few-shot 상한(cap) 등의 주요 설정값을 다양한 조합으로 변경하며, 각 설정에서 입력 초과 및 truncation 발생 비율을 시뮬레이션 방식으로 점검하였다.

그 결과, 일부 설정에서는 few-shot 예시가 입력의 과도한 비중을 차지하거나 제시문이 빈번하게 축약되는 문제가 확인되었다. 이를 바탕으로 few-shot 비율을 0.6 수준으로 제한하고, 출력 토큰과 안전 여유를 고려한 입력 예산 관리 전략이 가장 안정적이라는 결론에 도달했다.

실제 추론 단계에서는 전체 입력 토큰 중 일정 비율만을 few-shot 예시에 할당하고, 토큰 초과가 발생할 경우 few-shot 제시문을 선택적으로 축약하였다. 이를 통해 긴 지문을 포함한 문제에서도 입력 길이 초과로 인한 추론 실패를 최소화 할 수 있었다.

3-3-3. Prompt 구성 및 설계 의도

추론에는 few-shot 예시를 포함한 Answer-only prompt 구조를 사용하였다. 출력 단계에서는 사고 과정을 노출하지 않고 정답만 JSON 형식으로 반환하도록 유도함으로써, 출력 형식 오류를 줄이고 후처리 안정성을 높이고자 하였다.

또한 여러 지시 방식이 모델의 응답 안정성과 선택 다양성에 미치는 영향을 확인하기 위해, **서로 다른 두 종류의 system prompt를 사용**하였다. 이를 통해 단일 프롬프트에 과도하게 의존하는 문제를 완화하고, 앙상블 단계에서 보다 다양한 추론 결과를 확보할 수 있도록 설계하였다.

3-3-4. 추론 안정성 확보 및 앙상블 전략

대규모 배치 추론 과정에서 발생할 수 있는 **llama 서버의 일시적인 500 오류 또는 무응답 상황**을 고려하여, 단일 추론 실패 시 즉시 종료하지 않고 최대 2회까지 재시도하는 복구 로직을 적용하였다. 재시도 시에는 동일한 입력을 유지하되 seed를 변경하여 추론을 다시 수행함으로써, 서버 불안정이나 일시적인 출력 오류가 전체 결과에 미치는 영향을 최소화하였다. 또한 정답 형식이 정상적으로 추출되지 않는 경우에도 동일한 방식으로 한 차례 추가 추론을 수행하였다.

추론 결과의 안정성을 높이기 위해 **다중 루프 앙상블 전략**을 적용하였다. 서로 다른 system prompt와 seed 조합을 사용해 각 문제를 여러 차례 추론하고, 각 결과에서 정답만을 추출한 뒤 **다수결 방식으로 최종 답안을 결정**하였다. 이와 함께 문제별 응답 다양성을 기록하여, 앙상블이 실제로 다양한 추론 경로를 생성하고 있는지도 최종적으로 점검했다.

	public score	private score
GGUF-Qwen3-Next-80B-A3B-Thinking	0.8327	0.7963
GGUF-Nemotron-3-Nano-30B-A3B	0.7504	0.6814
Qwen2.5-32B-logit-epoch:3_CPT-all_train	0.7616	0.7962

GGUF 포맷으로 추론한 대형 모델 계열이 전반적으로 더 안정적인 성능을 보였다. 반면 `Nemotron-3-Nano-30B-A3B`는 상대적으로 경량화된 구조로 인해 추론 깊이나 장문 맥락 유지 측면에서 한계가 나타났고, 이 차이가 리더보드에서의 성능 격차로 이어진 것으로 판단했다.

3-4. 앙상블

3-4-1. Hard Voting

Hard Voting의 경우, 지금까지 생성하여 제출했던 submission.csv 파일들을 수집한 뒤, 리더보드 macro F1성능이 0.7 이상인 결과만 앙상블 후보로 선별하여 사용하였다. 각 모델(혹은 체크포인트)의 예측 결과에서 id 별로 예측한 정답(1~5)을 하나의 투표 집합으로 모으고, 가장 많이 선택된 정답을 최종 답으로 채택하는 방식으로 Hard Voting을 수행하였다. 동률(tie)이 발생하는 경우도 존재했는데, 이 경우에는 가장 높은 성능을 기록한 모델의 예측을 우선하여 최종 정답으로 결정하였다.

3-4-2. Soft Voting

Soft Voting의 경우, Hard Voting과 동일하게 각 모델의 예측 정답(1~5)을 집계하되, 모델별 성능 차이를 반영하기 위해 리더보드 점수를 가중치로 사용한 가중 다수결(Weighted Voting) 방식으로 수행하였다. 각 문제에서 선택지별로 예측한 모델들의 가중치를 합산해 최종 답을 결정했으며, 동점이 발생할 경우에는 가장 성능이 높은 모델의 예측을 우선하도록 tie-break 규칙을 적용하였다. 가중치 조합을 여러 버전으로 조정·검증한 뒤, 검증 성능이 가장 높았던 설정을 최종 제출 결과로 채택하였다.

	public score	private score
Hard Voting	0.7936	0.7477
Soft Voting	0.8438	0.8015

4. 개인 회고

4-1. 최준호

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

1-1. 문제 정의

지난 대회에서 개인적으로 다음 n개의 아쉬웠던 부분이 있었다.
1. 팀적으로 활동 및 공유 미비
2. 해결 방법에 대해 위험도와 발전 가능성 사이에서 마지막 즈음에 너무 큰 아이디어를 소화하느라 다른 곳에 집중을 잘 못함

1-2. 학습 목표

이전 피드백을 바탕으로 이번 대회에선 다음과 같은 학습 목표를 세웠다.
1. 팀 전체와 아이디어를 공유하며 그 아이디어를 함께 디벨롭할 팀원들을 모으고 같이 고민해볼 것
2. 역할 분담을 더욱 확실히 할 것
3. 이전의 실패 경험을 바탕으로 스케줄링에 위험도 반영에 더욱 가중치를 둘 것

1-3. 활동

처음부터 모든 스케줄을 맞추는 것보단 유동적으로 큰 틀만 잡아둔 뒤, 세부 스케줄은 틀 안에서 새로운 아이디어에 대해 관심있는 소수의 팀원들을 모집하여 디벨롭하는 유연한 방향으로 설정하였다. 이러한 방법을 활용하다보니 모두가 프로젝트에 일정량 이상 기여하면서 흥미를 잃지 않을 수 있었다.

2. 나는 어떤 방식으로 모델을 개선했는가?

2-1. Agentic RAG

AI Agent 기술과 RAG 기술을 접목하여 단순히 필요한 정보를 Retrieval하는 것이 아닌 Query 형태의 입력에 대한 전처리, Retrieval Validation, 프롬프트 엔지니어링 등의 기술을 접목하여 모델 성능에 의존하는 것이 아니라 결과를 처리하기까지를 하나의 파이프라인화를 하여 성능을 향상시켰다.

2-1-1. 문제 정의

단순히 RAG 기술만을 사용하였을 때 현재 입력 중 대부분인 paragraph에 의존하여 질문에 필요한 정보를 가져오지 못하였다. 이를

좀 더 Agentic하게 솔루션을 만들 수 있지 않을까라는 가정에서 해당 아이디어가 비롯되었다.

2-1-2. 해결 과정

Agent라는 기술을 활용하니 단순히 모델의 추론에만 의존하는 것보다 적용할 수 있는 방법들이 많았다. 대표적으로 이번 기술에서 적용했듯 Recursive하게 Query의 유효성을 검증하는 프로세스, LLM이 잘 답변할 수 있도록 큰 문제를 계층적으로 작게 분할하는 프로세스(과목 식별 -> RAG 생성 -> 근거 생성 -> 최종 답변) 등을 적극적으로 활용해보며 AI에서도 모델을 잘 만드는 것 이외에도 여러가지의 엔지니어링적인 가능성을 느꼈다.

2-1-3. 피드백

결과적으로 "한국사"에 한정해서는 Ensemble 솔루션보다 잘 동작하였다. 시간 관계 상(모두 추론하는데 이틀 이상의 시간이 소요됨) 다른 과목에서 적용하지 못했던 것이 아쉬우며, 좀 더 빡빡하게 스케줄링했다면이라는 아쉬움이 들었다.

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

모든 것이 당장에 해결된다고 생각하지 않는다. 아직까지는 기술과 시스템에 대해 잘 이해하지 못한다는 생각이 들었고, 여러 경험을 쌓아가며 해결될 것이라고 생각한다. "프로는 생각한 대로 되고, 아마추어는 걱정한 대로 된다"라는 말이 있듯, 나의 생각이 좀 더 정답에 가까워지도록 경사하강을 열심히 해봐야할 것 같다.

4-2. 우혜진

1. 나는 어떤 방식으로 모델을 개선했는가?

이번 프로젝트에서 세운 가장 큰 목표는, **모델을 바로 바꾸거나 성능 수치를 쫓기 전에 데이터와 실험 결과를 더 집요하게 검증해보자**는 것이었다. 단기적인 리더보드 점수에만 반응하기보다는, 하나의 방법론이 실제로 가능성이 있는지 일정 기간 끝까지 밀어보는 실험 전략을 가져가고 싶었다.

이를 위해 데이터 EDA를 바탕으로 수동 증강을 반복적으로 진행했고, 성능이 유독 낮게 나오는 영역에 대해서는 원인을 확인한 뒤 추가 증강을 이어갔다. 또한 임베딩 기반 Few-shot 프롬프팅을 설계하여 문제 유형에 맞는 예시를 자동으로 구성했고, GGUF 포맷 기반 대형 모델 추론 환경을 구축해 few-shot inference와 앙상블 실험을 병행하였다. 단순히 "모델을 바꾸는 개선"이 아니라, 입력·추론·결과 검증 전반을 조정하는 방향으로 접근했다.

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

데이터 수동 증강에만 약 1주일 반이 걸렸고, 이후에도 학습 성능이 떨어지는 영역을 중심으로 계속해서 증강을 반복했다. 솔직히 말하면 중간중간 '이게 맞나?' 싶은 순간도 많았고, 이른바 현타가 크게 왔던 시점도 있었다. 그럼에도 불구하고, "하나의 문제에 끝까지 몰입해보고 싶다"는 목표만큼은 이번에 확실히 지켰다고 생각한다.

지난 대회 랩업 리포트에서 가장 큰 한계로 꼽았던 것이 *데이터에 대한 이해 부족*이었는데, 이번에는 CSV 분석과 Streamlit 기반 EDA를 통해 데이터 분포와 취약 지점을 직접 확인하고, 성능 저하의 원인이 데이터인지 학습 방식인지 분리해서 보려고 노력했다. 단순히 데이터를 늘리는 것이 아니라, **어떤 유형에서 왜 모델이 흔들리는지를 확인하고 그에 맞춰 증강과 학습 전략을 조정하는 과정**을 반복했다는 점에서 이전과는 확실히 다른 경험이었다.

또한 추론 단계에서도 많은 시행착오를 겪었다. GGUF 기반 대형 모델을 활용한 few-shot 추론 환경을 직접 구성하면서, 모델 크기 자체가 주는 추론 안정성과 맥락 유지 능력이 성능에 큰 영향을 준다는 것을 체감했다. 단순히 "큰 모델이 좋다"가 아니라, few-shot 구성 방식, 토큰 예산 관리, seed와 prompt 변화에 따른 출력 변동성까지 포함해서 추론 자체를 하나의 실험 대상으로 바라보게 되었다.

마지막으로 앙상블을 통해 얻은 깨달음도 컸다. 대회 막바지에 성능 향상을 위해 앙상블을 적극적으로 적용하면서 **이게 과연 '모델 개선'이라고 부를 수 있는가**에 대한 고민이 들었다. 구조나 학습을 더 바꾸는 것이 아니라, 이미 나온 결과들을 조합해 리더보드 점수를 끌어올리는 행위가 단순히 순위만을 위한 최적화처럼 느껴지기도 했다.

하지만 마스터님께서 이러한 접근이 오히려 최근 대회 환경에서 매우 일반적인 전략이며, 제한된 시간과 자원 안에서 최선의 결과를 도출하기 위한 합리적인 선택이라는 이야기를 해주셨다. 그 말을 통해, 여러 모델과 추론 결과를 어떻게 조합하고, 불확실성을 어떻게 줄이느냐 자체가 하나의 중요한 문제 해결 능력이 아닐까 하고 생각이 바뀌게 됐다.

3. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

아쉬웠던 점 중 하나는 협업 측면에서 Git, Jira 같은 도구를 충분히 활용하지 못했다는 것이다. 프로젝트 후반으로 갈수록 각자 실험에 집중하다 보니 일정 관리나 작업 흐름을 구조적으로 정리하는 데 한계가 있었다.

또 하나는 자원 제약이다. Tesla V100-SXM2-32GB라는 한정된 환경에서 최상의 결과를 내야 했기 때문에, 더 크고 다양한 모델을 자유롭게 실험하지는 못했다. 실제로 실험을 진행할수록 "결국 큰 모델을 쓰는 방향으로 수렴한다"는 느낌도 강하게 들었고, 대회라는 특성상 다양한 방법론을 충분히 병렬로 탐색하지 못한 점은 아쉬움으로 남는다. 오히려 대회보다는 **프로젝트 형태의 작업이 더 잘 맞겠다는 생각**을 하게 됐다.

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

이번 프로젝트를 통해, 성능 자체보다도 **실험을 설계하고 끝까지 검증하는 과정에서 가장 큰 재미를 느낀다는 점**을 분명히 알게 되었다. 그래서 다음 프로젝트에서는 더 명확한 가설을 세우고, 리더보드 압박에서 조금은 자유로운 환경에서 실험 중심의 접근을 해보고 싶다.

4-3. 곽나영

1. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했고, 어떤 깨달음을 얻었는가?

지난 프로젝트에서는 데이터 분석 없이 모델링 중심으로 작업을 진행했다. 또한 협업 측면에서는 협업 도구를 활용해 팀원들의 작업 상황을 파악하는 것을 학습 목표로 삼았다. 이번 프로젝트에서는 데이터 분석과 데이터셋 구축을 중심으로, 팀원들과 함께 데이터를 직접 확인하고 정리하는 과정을 수행했다. 특히 '수능형' 태스크에 적합한

데이터가 되기 위해 어떤 형태로 구축해야 하는지 고민하며 데이터 구축 방향을 설계했다. 현업에서도 데이터가 부족하거나 신규 모델 학습을 위해 거의 데이터가 없는 상태에서 데이터를 만들어야 하는 상황이 빈번하다. 이번 경험을 통해, 주어진 상황을 먼저 분석하고 그에 맞게 데이터를 구축·증강하는 과정 자체가 실무 역량으로 직결된다는 점을 체감했다. 협업 측면에서는 노션을 통해 작업 현황을 공유하고, 분업한 업무를 명확히 기록함으로써 팀원들의 진행 상황을 상시 파악할 수 있었다. 다만 다음 프로젝트에서는 문서화를 더 체계화해, 작업 공유 비용을 줄이고 팀 전체의 시간을 절약할 수 있도록 개선해야 한다는 점을 깨달았다.

2. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?'

현업에서 많이 활용되는 Jira와, 개발자로서 필수 역량인 Git을 적극적으로 사용해보고 싶었으나, 모델 학습과 실험에 시간을 우선 투입해야 하는 상황으로 인해 충분히 적용하지 못한 점이 아쉬웠다. 또한 리더보드 기반의 성능 경쟁 환경에서 순위를 의식하지 않을 수 없었고, 그 결과 큰 파라미터 모델 위주의 실험을 반복하는 데 치우친 면이 있었다. 데이터가 한국어 기반이므로 한국어 모델 학습도 시도했으나, 기대만큼의 성능 향상이 나타나지 않아 최종 제출 모델과 앙상블에는 반영하지 못했다. 무엇보다 "성과가 잘 나오지 않는 실험"에 시간을 과도하게 투입한 점이 아쉬웠다. 특히 Gemma 2 27B 모델 학습 이슈를 해결하려고 약 3일간 집중했는데, 더 빠르게 한계를 판단하고 다른 모델 실험으로 전환했다면 팀 전체 성과와 개인 성장 측면에서 더 효율적이었을 것이라고 생각한다. 실험이 막힐 때는 빠르게 원인을 정리하고 다음 실험으로 넘어가는 판단력의 중요성을 크게 느꼈다. 마지막으로 RAG와 Agent 작업을 직접 수행해보고 싶었지만, 팀 전체 인력이 모두 투입되는 것은 비효율일 수 있다는 판단으로 실제 적용을 시도하지 못한 점도 아쉬움으로 남는다.

3. 한계와 교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

이제 마지막 프로젝트만 남은 만큼, 팀에 더 실질적으로 기여하는 사람이 되기 위해 더욱 적극적으로 임하고자 한다. 또한 실험 과정에서 빠르게 상황을 판단하고 다음 단계로 넘어가기 위해, 문제–원인–결과를 구조적으로 정리하는 분석 습관을 강화하려 한다. 이번 프로젝트에서는 RAG와 Agent가 실제로 활용되었기 때문에, 팀원들이 작성한 코드를 기반으로 동작 원리와 설계 의도를 반드시 학습할 계획이다. 마지막으로 이번에는 적용하지 못했지만, 다음 프로젝트에서는 Git 기반 협업 관리를 도입해 코드 수준에서도 협업 프로세스를 체계화해보고 싶다.

4-4. 박서진

1. 학습 목표 달성을 위해 무엇을 어떻게 했는가?

본 프로젝트의 목표는 언어모델을 활용하여 수능형 객관식 문제를 해결할 수 있는 모델을 개발하는 것이다. 이를 위해 단순히 모델을 교체하는 데 그치지 않고, 베이스라인 코드 분석을 시작으로, 모델 규모, 추론 방식, 학습 전략 전반에 걸친 실험을 단계적으로 수행하였다. 특히 제한된 GPU 자원 환경에서도 대규모 모델을 활용하기 위해 Unsloth 기반 학습 환경을 구축하고, 다양한 파인튜닝 및 추론 방식을 비교, 분석하였다.

2. 어떤 방식으로 모델을 개선했는가?

모델 개선은 크게 세 단계로 이루어졌다. 먼저, 베이스라인으로 사용한 2B 규모의 소형 모델에서 대규모 Qwen 계열 모델로 전환하여, 모델 규모에 따른 성능 차이를 확인하였다. 이후 생성 기반 추론 방식의 한계를 분석하고, 출력 분포를 활용한 logit 기반 추론 방식으로 전환함으로써 추론 안정성을 개선하였다. 마지막으로 주제별 성능 분석을 통해 모델이 특정 주제(한국사)에서 상대적으로 취약함을 확인하고, 이를 보완하기 위해 데이터 증강과 Continued Pre-Training을 적용하였다.

3. 그 결과로 무엇을 달성했고, 어떤 깨달음을 얻었는가?

실험 결과, 단순한 LoRA 기반 파인튜닝이나 데이터 증강만으로는 성능 향상이 제한적임을 확인하였다. 반면, 증강된 데이터를 활용한 Continued Pre-Training을 적용했을 때 가장 큰 성능 개선을 달성할 수 있었다. 이를 통해 수능형 문제와 같이 도메인 지식이 중요한 과제에서는 출력 형식 조정보다 모델 내부에 지식을 학습시키는 과정이 성능 향상에 핵심적인 역할을 한다는 것을 깨달았다. 또한 Qwen 모델이 중국어 데이터 비중이 높은 모델이라는 점을 고려할 때, 한국사와 같이 특정 국가의 역사와 문화에 강하게 의존하는 영역에서 성능이 상대적으로 낮게 나타날 수 있음을 관찰하였다. 이를 통해 모델이 사전 학습된 데이터의 지역적 특성이 특정 도메인 성능에 상당한 영향을 미친다는 점을 인식하게 되었다.

4. 마주한 한계와 아쉬웠던 점은 무엇인가?

가장 아쉬웠던 점은 프롬프트 엔지니어링을 통해 generate 방식의 성능을 추가로 향상시키고자 했으나, 기대한 만큼의 결과를 얻지 못했다는 점이다. CoT 프롬프트를 적용하면 문제 해결 과정을 단계적으로 설명하도록 유도함으로써 모델이 정답에 더 근접한 추론을 수행하는 방식으로서 generate의 장점을 살려 좀 더 좋은 성능이 나올 것이라고 기대하였다. 하지만, 실제 실험에서는 오히려 성능이 저하되는 결과가 나타났다. 이러한 결과가 발생한 원인에 대해 추가적인 분석과 실험을 통해 개선 방안을 도출하고자 했으나, 시간적 제약으로 인해 해당 과정을 충분히 진행하지 못한 점이 아쉬움으로 남는다.

5. 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

가장 아쉬움이 남았던 generate 방식의 잠재력을 완전히 검증하지 못한 점을 고려하여, 프롬프트 설계와 추론 전략을 보다 체계적으로 개선하는 실험을 추가로 수행하고자 한다. 특히 CoT 방식이 오히려 성능 저하로 이어졌던 원인을 분석하고, 문제 유형에 따라 추론 방식을 선택적으로 적용하는 방향으로 모델 활용 전략을 확장해볼 계획이다. 또한 실험 전반에 걸쳐 Weights & Biases(W&B)와 같은 실험 관리 및 추적 도구를 적극적으로 활용하지 못한 점도 아쉬운 부분이다. 이로 인해 실험 설정, 하이퍼파라미터 변경, 성능 변화 과정을 체계적으로 기록·비교하는 데 한계가 있었으며, 일부 실험 결과에 대해 보다 정밀한 분석을 수행하지 못했다는 점에서 아쉬움이 남았으며 다음 프로젝트에서는 이런 보조 도구를 사용하여 모델의 성능을 좀 더 체계적으로 기록하고 시각화하는 방향으로 진행하고 싶다.

4-5. 김이슬

1. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했고, 어떤 깨달음을 얻었는가?

지난 프로젝트에서는 데이터 분석과 구조 설계 위주로 참여하면서, 모델링과 추론 분석을 깊이 있게 다뤄보지 못한 점이 개인적으로 아쉬움으로 남아 있었다. 이에 이번 프로젝트에서는 모델링과 추론 과정을 직접 맡아보고, 모델이 문제를 어떻게 이해하고 답을 도출하는지를 분석해보는 것을 학습 목표로 삼았다. 단순히 성능 수치를 개선하는 것보다, 수능형 문제에서 모델이 어떤 유형의 문제를 잘 풀고 어떤 문제에서 반복적으로 실패하는지를 파악하는 데 집중하고자 했다.

이를 위해 RAG 구조를 직접 구현하고 성능 테스트를 진행하면서, 단순한 성능 비교가 아니라 문제 유형별 추론 과정을 관찰했다. 특히 (가)~(라)와 같은 보기 포함 문제, 시기·순서 정렬 문제, '옳은 것/옳지 않은 것'을 판단하는 문제 등에서 모델이 어떤 정보를 참조하고 어떤 단계에서 추론에 어려움을 겪는지를 확인했다. 그 결과, 검색된 문서 내에 정답에 직접적으로 대응되는 정보가 포함된 경우에는 비교적 안정적으로 정답을 도출했지만, 여러 정보를 종합하거나 관계를 재구성해야 하는 문제에서는 추론이 흔들리는 경향을 보였다. 이를 통해 수능형 문제 풀이에서는 단순한 검색 증강만으로는 한계가 있으며, 추론 과정을 보조하고 점검하는 구조가 필요하다는 점을 체감했다.

2. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

프로젝트를 진행하며 가장 크게 느낀 한계는 ㄱ~ㄹ, (가)~(라)와 같은 지시 정보가 포함된 문제에서 모델이 실제로 찾아야 할 대상을 정확히 특정하기 어려웠다는 점이었다. 문제 유형별로 나누어 단계적인 실험을 진행하고, 검색 전략과 추론 흐름을 달리하는 시도도 해보았지만, 이러한 접근이 전체 테스트셋에 대해 일반화 가능한지까지는 충분히 검증하지 못했다. 또한 Agentic RAG 구조 역시 한국사 문제를 중심으로 적용되었기 때문에, 다른 과목이나 다양한 문제 유형 전반에서의 효과를 체계적으로 비교하지 못한 점이 아쉬움으로 남는다.

3. 한계와 교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

마지막 프로젝트인 만큼, 팀에 보다 실질적으로 기여하는 역할을 수행하고 싶다. 이를 위해 단순히 주어진 작업을 수행하는 데 그치지 않고, 프로젝트에 필요한 배경 지식과 기술을 사전에 학습하며 문제 해결 과정에 능동적으로 참여하고자 한다. 특히 이번 프로젝트에서 한계로 느꼈던 모델 추론 분석과 Agentic RAG 구조에 대해 추가로 공부하며, 팀 내에서 발생하는 기술적 이슈를 함께 고민하고 해결할 수 있는 역할을 맡고 싶다.

또한 다음 프로젝트에서는 실험 과정을 보다 체계적으로 관리하는 데에도 집중할 계획이다. 문제 정의부터 가설 설정, 실험 결과 정리까지의 흐름을 명확히 기록하고, Git 기반의 버전 관리와 실험 로그 정리를 통해 팀원들이 서로의 작업을 쉽게 이해하고 이어갈 수 있는 환경을 만들고자 한다. 나아가 이번 프로젝트에서 여러 시도를 병렬적으로 확장하기보다는 하나의 문제, 특히 '모델이 어떤 근거를 바탕으로 판단에 도달하는지'를 끝까지 추적해, 데이터·검색·추론 과정을 깊이 있게 파고드는 실험을 해보고 싶다.

4-6. 김윤희

1. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했고, 어떤 깨달음을 얻었는가?

본 프로젝트에서 나의 학습목표는 수능형 객관식 문제 풀이 태스크를 단순한 LLM 추론 문제로 다루는 것이 아니라, 데이터 구조와 문제 유형을 먼저 이해하고 이에 맞는 모델링 전략을 설계하는 것이었다. 이를 위해 원본 데이터를 하나씩 분석하며 지문, 질문, 선지, 보기 간의 역할과 상호 관계를 세분화했고, 문제 유형별로 모델이 반드시 참조해야 하는 정보의 범위를 정리하였다. 이 과정에서 단순히 데이터 수를 늘리는 것보다, 각 문제의 주제 세분류와 난이도, 오답 발생 패턴, 지시 정보의 명확성 등을 기준으로 데이터 품질을 점검하는 작업에 많은 시간을 할애했다. 특히 모델이 반복적으로 오답을 내는 유형이나 특정 선택지에서 혼동을 보이는 문제들을 중심으로, 데이터가 부족하거나 표현이 불명확한 경우를 식별하고 유사 유형의 문제를 추가 수집·증강하는 작업을 진행하였다. 이러한 데이터 가공 과정에서 동일한 모델이라도 입력 구성 방식과 데이터 표현에 따라 성능이 크게 달라질 수 있음을 직접 체감했고, 모델을 개선하기 이전에 데이터 관점에서 태스크를 재정의하는 것이 성능 향상의 핵심이라는 깨달음을 얻었다. 이 인식을 바탕으로 단순히 검색 결과를 사용하는 RAG가 아니라, 검색된 정보의 품질을 점검하고 필요 시 재검색을 수행하는 agentic RAG 구조를 설계하여, 문제 유형 이해와 데이터 기반 판단을 코드 수준에서 구현하고자 했다.

2. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

프로젝트를 진행하며 가장 크게 느낀 한계는 데이터 자체가 가진 구조적 제약이었다. 특히 ㄱ~ㄹ, (가)와 같은 지시 정보가 포함된 문제의 경우, 실제로 참조해야 할 범위가 paragraph에 명확히 드러나지 않는 경우가 많았고, 이는 모델이 정답에 도달하는 데 본질적인 어려움으로 작용했다. 해당 문제를 해결하기 위해 LLM 기반 범위 추론, span prediction 모델 fine-tuning 등 여러 접근을 구상했으나, test 데이터 역시 동일한 한계를 공유하고 있고 해당 유형의 샘플 수가 충분하지 않아 ablation 실험으로 이어가지 못했다. 또한 agentic RAG를 한국사 도메인에만 적용해 실험한 점 역시, 다른 과목으로의 일반화 가능성을 충분히 검증하지 못했다는 점에서 아쉬움으로 남는다. 기술적 시도 외적인 부분에서도 몇 가지 한계가 존재한다. 팀원들 간의 소통과 역할 분담 자체는 비교적 원활하게 이루어졌고, 각자 담당한 영역에 대해 책임감을 가지고 작업을 진행했지만, Git을 활용한 체계적인 버전 관리와 실험 기록 관리가 충분히 이루어지지 못한 점은 분명한 아쉬움으로 남는다. 실험 코드와 설정 변경이 빠르게 반복되는 과정에서 브랜치 전략이나 커밋 단위의 관리가 미흡했던 것 같다.

3. 한계와 교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

이번 프로젝트를 통해 얻은 가장 큰 교훈은 모델 성능 향상의 출발점이 모델이 아니라 데이터와 태스크 정의라는 점이다. 다음 프로젝트에서는 문제 유형별로 필요한 정보 범위를 명시적으로 어노테이션한 데이터셋을 구성하고, 이러한 구조적 정보가 실제 test 성능 향상으로 이어지는지를 체계적으로 검증해보고자 한다. 또한 agentic RAG 구조를 한국사뿐 아니라 다른 도메인에도 확장 적용하여, 검색 실패 패턴과 재검색 전략의 일반성을 분석해보고 싶다. 나아가 데이터 관점에서 문제를 정의하고, 이를 해결하기 위한 가설을 세운 뒤 실험적으로 검증하는 일련의 과정을 보다 정교하게 설계함으로써, 단순한 모델 튜닝을 넘어 문제 해결자로서의 역량을 더욱 강화하고자 한다.