Open-Domain QA Wrap-up Report

NLP-09팀 - 파란부스터
강다형, 김민석A, 김세훈, 박준범, 이태원, 허승환

1. 프로젝트 개요

1.1 개요

방대한 정보 환경에서 사용자가 원하는 질문에 대해 정확한 답변을 제공하는 것은 검색 기술의 핵심 과제이다. 본 프로젝트는 Wikipedia 데이터를 기반으로 한 ODQA(Open-domain Question Answering) 문제를 해결하는 것을 목표로 한다.

질문과 관련된 문서를 검색하는 Retrieval 단계와, 검색된 문서로부터 정답을 추출하는 Reader 단계로 구성된 two-stage 구조의 모델을 설계하고 성능을 개선하였다.

1.2 환경

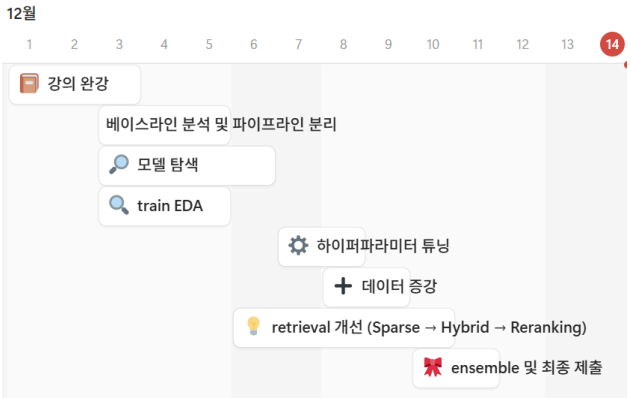
- 팀 구성 및 컴퓨터 환경 : 6인 1팀, V100 서버 3개를 VSCode와 SSH로 연결하여 사용
- 협업 환경 : Github, Notion, Wandb
- 의사 소통 : Slack, Zoom, 카카오톡

2. 프로젝트 팀 구성 및 역할

팀원	역할
강다형	베이스라인 코드 재설계, 실험관리 인프라 설정, Retrieval 모델 리서치 및 실험(sparse+dense), Github 관리
김민석	Retrieval 모델 리서치 및 실험(DPR), 데이터 증강(negative Sampling)
김세훈	EDA, 데이터 전처리, 데이터 후처리, 데이터 증강(negative Sampling)
박준범	MRC 모델 리서치 및 실험, Retrieval 모델 리서치 및 실험(Reranking)
이태원	MRC 모델 리서치 및 실험, 하이퍼파라미터 튜닝, 데이터 후처리, tapt, 앙상블
허승환	EDA, 앙상블

3. 프로젝트 수행

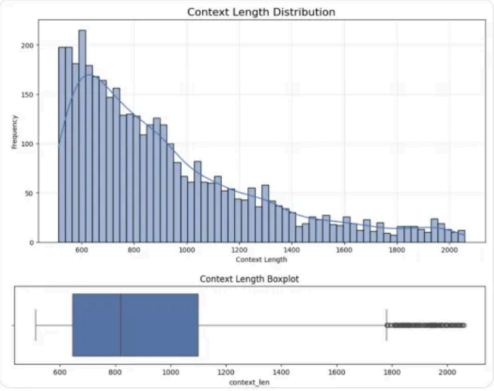
3.1 프로젝트 수행 타임라인



3.2 Data

3.2.1 EDA

- Train Context 길이 분석



<그림 1> train context 길이 분포

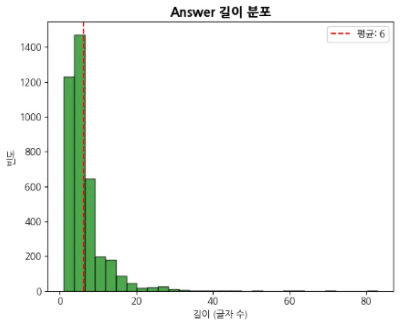
전체 context 길이에 대한 분포를 분석한 결과 평균적으로 700~1100자 범위에 분포하며, 500~900자 구간에 가장 밀집된 형태를 보인다.

Max Length	Eval/ EM	Eval/ F1
256	51.67	59.97
384(baseline)	55.83	65.19
512	55.72	67.11

- Max Length 기반 실험

이러한 context 길이 분포로 인해, 불필요한 chunking이 발생할 가능성을 고려하여 max_length 설정에 따른 성능 변화를 실험하였다. 예상대로 max_length 512에서 chunking이 가장 적었으며,baseline 대비 EM은 유사하나 F1 점수는 소폭 상승하는 것을 확인하였다. 반대로 256은 context truncation과 chunk 증가로 인해 성능이 크게 하락하였다.

- answer 길이 분포

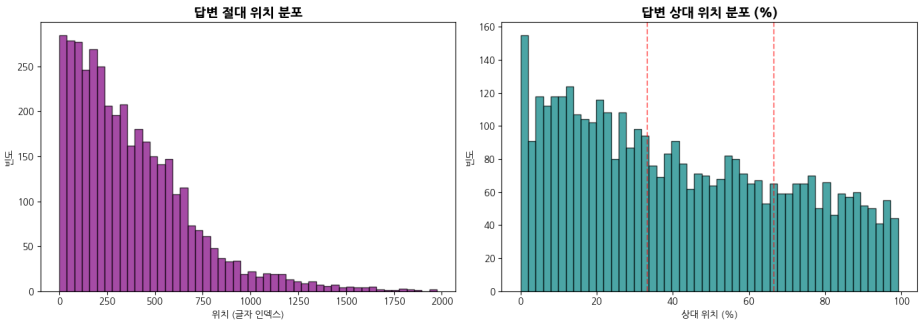


<그림2> Answer 길이 분포

answer 길이 분포의 분포가 평균이 6이고 대부분의 answer도 20자를 거의 넘지 않음을 확인하였다.

max length가 40이하인 점을 이용해 inference의 max_length_answer을 넉넉하게 50으로 설정하였다

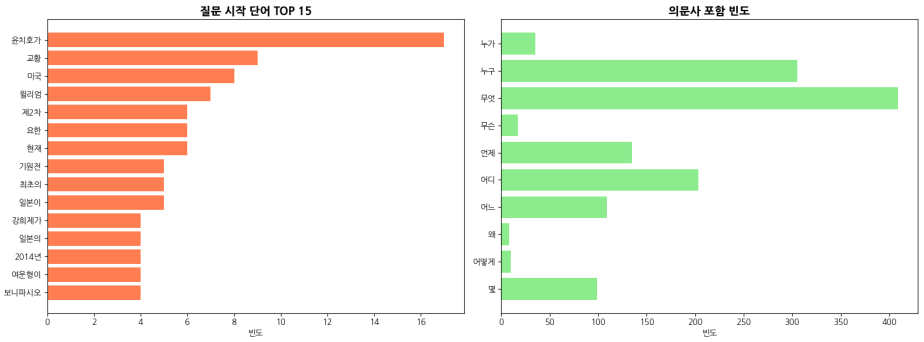
- 답변 위치 분석



<그림3,4>답변길이 절대,상대 분포

답변의 위치가 context의 앞에 위치한다는 것을 알 수 있었다. 이를 통해 만약 토크나이저의 한계로 context를 잘라야한다면 앞의 context를 유지하고 뒤의 context를 지우는 실험을 진행하려했으나 sliding window를 활용한 chunking을 베이스라인코드에서 지원해 이에 대한 실험은 진행하지 않았다.

- **질문 분석**



<그림5,6> 질문 시작 단어, 의문사 포함 빈도

질문 시작 단어의 빈도로 보아 질문과 context는 여러 도메인이 고르게 섞여있다는 것을 확인하였다.

의문사 빈도로 보아 무엇, 언제, 어디, 누구와 같이 단순 단어 답변을 요구하는 질문이 왜, 어떻게와 같이 어려운 답변을 요구하는 질문이 더 많다는 것을 확인하였다.

3.2.2 data 증강

기존 MRC 학습용 train 데이터는 각 질문에 대한 정답을 포함한 gold context만 제공되기 때문에, 실제 추론 단계의 retrieval 환경을 충분히 반영하지 못하는 한계가 있었다. 그래서 Negative Sampling을 적용하여 문맥적 다양성과 판별 난이도를 증가시키고자 하였다.

- Negative Sampling 실험
  - 실험 1 (BM25 → bi-encoder → cross-encoder)
    1. konply okt, BM25로 후보군 200개 추출
    2. 1의 후보군 중 bi-encoder로 유사도 점수 기반 top-k 후보군 30개 추출
    3. cross-encoder로 유사도 점수 기반 threshold 이내 top-k 최대 3개 추출
  - 실험 2 (BM25 → bi-encoder → 클러스터링)
    1. MeCab, BM25로 후보군 200개 추출
    2. 1의 후보군 중 DPR 임베딩 기반 semantic similarity 계산
    3. 유사도 분포에 따라 negative sample 5개 추출 (low 1개, Mid 1개, K-means 3개)

단순 concat을 사용하면 모델이 gold context의 위치와 답의 위치가 항상 앞이라고 학습할 위험이 있다고 예상되어 negative sample과 gold context의 위치를 shuffle하여 데이터셋을 만들었다.

Method	Eval/ EM	Eval/ F1
base	61.25	70.03
실험1	65.42	71.59
실험2	58.33	66.19

실험1의 결과로 보아 negative sampling을 통해 유의미한 모델 성능 개선을 이룰 수 있었다.

3.2.3 데이터 전처리

Train 데이터와 위키 문서에 공통적으로 존재하는 불필요한 공백, 특수기호, 위키 refernce 등 노이즈를 제거하여 chunking시에 쓸데없는 정보를 제거하는 것이 목적하였다.
적용한 전처리 항목은 다음과 같다.

- **공백 및 줄바꿈 정리**
  - 줄바꿈 문자 제거
  - 연속 공백 통일
  - question과 context에 동일한 전처리 규칙 적용

- **정규화 기반 노이즈 제거**
  - HTML 태그 제거 (총 7,570개 태그 제거)
  - 위키 reference 태그 제거
  - URL 정보 제거 (총 823개 URL 제거)

dataset	Eval/ EM	Eval/ F1
raw dataset	55.83	65.19
clean dataset	60.83	68.7

전처리 데이터셋을 사용하여 chunking시에 쓸데없는 정보를 제거하여 유의미한 성능 향상을 얻을 수 있었다.

3.2.4 데이터 후처리

validation에서 모델이 예측한 결과와 gold answer 간의 차이를 분석해 모델이 어느 부분에서 오답을 발생하는 지 분석하고 개선하고자 하였다.

오답원인	prediction 예시	gold answer 예시	비율(%)
스팬 확장	작가 베게티우스	베게티우스	26.15
문맥 선택 오류	벤쿠버	베이징	23.08
숫자/연도	1871년	1975년	10.77
단복수/단어형 mismatch	35년간	35년	7.69
따옴표/특수문자 mismatch	“””5월의 왕”””	“5월의 왕”	6.15

- 형태소 기반 조사 처리 (단복수/단어형 mismatch)

  형태소 분석을 통해서 명사 + 조사 구조를 식별하고, 의미에 영향을 주지 않는 조사만 제거하는 방향으로 정규화를 수행했다. Validation 기준으로 EM이 약 4개 개선되는 효과를 확인하였다.

- 형식 mismatch (따옴표/특수문자 mismatch)

  불필요한 괄호, 따옴표 및 중복 따옴표를 제거하여 단순 mismatch를 완하하고자 하였다. Validation 기준으로 EM이 약 3개 개선되는 효과를 확인했다.

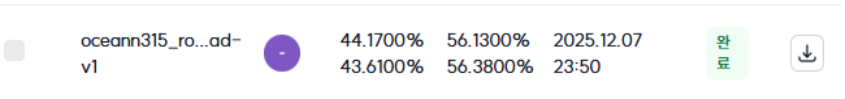
- 결과 분석

  두 방식 모두 리더보드상 점수를 동일하게 유지되었다. F1점수 기준으로는 형태소 기반 조사 처리의 경우 떨어졌고, 형식 mismatch 정규화는 F1점수에서 동일했다. 리더보드 채점시에는 어느정도 공백이나 중복 따옴표등을 처리해줬다고 판단된다.

3.3 Retrieval

초기 베이스라인 코드를 그대로 제출한 결과, 예상보다 낮은 점수를 기록하였으며, 이를 단순한 모델 성능 문제로 보기보다는 파이프라인 전체 구조 관점에서의 원인 분석이 필요하다고 판단하였다.

이에 QA 파이프라인을 Reader와 Retriever 단계로 분리하여 성능을 분석하였다. Validation 기준으로 Reader 단독 성능은 EM 약 0.7, Retrieval 단계의 Recall@10은 약 0.67에 그쳤다. 베이스라인 코드 구조상 최종 성능은 Reader의 EM과 Retriever의 Recall@K가 곱해진 값으로 근사 가능하다고 판단하였으며, 실제로 두 값을 곱한 결과는 약 0.47로, 베이스라인 제출 점수와 유사한 수준임을 확인하였다. 이를 통해 Reader 자체의 한계보다는 Retriever 성능이 전체 파이프라인의 병목으로 작용하고 있음을 명확히 인식하게 되었다.



<그림7> 베이스라인 score

이러한 분석을 바탕으로, Retriever 단계의 성능 향상이 전체 성능 개선에 가장 큰 기여를 할 것이라는 결론에 도달하였다. 이에 Retrieval 파이프라인을 Baseline Sparse Retrieval에서 시작하여, BM25 기반 개선, Sparse–Dense Hybrid 결합, 그리고 Cross-Encoder 기반 Reranking으로 단계적으로 고도화하는 접근을 선택하였다. 모든 실험은 Recall@K 지표를 기준으로 설계되었으며, 정답 문서를 얼마나 안정적으로 상위 후보군에 포함시킬 수 있는가에 초점을 맞추었다.

초기 Sparse Retrieval은 subword 기반 토크나이저와 단순 가중치 방식으로 구성되어 있었으나, 해당 방식은 성능 자체보다 확장성과 개선 여지가 제한적이라는 문제가 있었다. 이에 Sparse Retrieval이 전체 파이프라인의 기반이라는 점에 주목하여, 단순한 모델 교체가 아닌

Sparse Retrieval 자체의 안정성과 신뢰도를 높이는 방향으로 개선을 시도하였다. 토크나이저 선택과 문서 길이 보정 방식, 파라미터 조합이 Recall에 미치는 영향을 중심으로 BM25, BM25Plus, BM25L계열 알고리즘을 검증하였으며, 이를 통해 후보 문서 확보 단계의 품질을 우선적으로 강화하였다.

Retrieval 방법	Recall@1	Recall@5	Recall@10	Recall@20	Recall@50
TF-IDF(baseline)	27.9%	58.8%	67.1%	74.2%	84.6%
BM25	53.7%	80.8%	85.8%	90.4%	95.0%
BM25Plus + Kiwi (k=1.2, b=0.6 d=0.5)	55.0%	82.5%	86.2%	91.6%	93.8%
민석 DPR	47.61%	74.08%	78.94%	85.36%	
세훈 DPR	27.9%	44.5%	51.7%	68.5%	
nlpai-lab/KURE-v1	0.5264	0.7385	0.7968		
nlpai-lab/KoE5	0.5016	0.7082	0.7530		
Hybrid (BM25 + KoE5,α=0.5) + Reranking	80.00%	93.75%	95.00%	96.67%	97.67%

DPR 학습에는 앞서 Sparse Retrieval과 Reranking을 통해 생성한 negative sample을 활용하였으며, Bi-Encoder 구조를 사용하여 question encoder와 passage encoder를 분리 학습하고, positive 문서와 negative 문서 간의 표현 거리를 벌리는 방식으로 학습을 진행하였다.

다만, 문서 코퍼스 규모가 제한적이었고 BM25 계산 속도 제약으로 인해 후보 문서 pool을 200개 수준으로 제한할 수밖에 없어, 충분히 hard한 negative sample을 다량 확보하기 어려운 문제가 있었다. 이에 일부 hard negative를 포함시키는 동시에 in-batch negative 학습을 적용하여 학습 효율을 보완하였다. 또한 score에 temperature를 적용하여 학습 강도를 조절하였으며, 이에 hard sample의 영향력을 상대적으로 강화하기 위해 temperature를 0.1로 설정하였고, 해당 설정에서 train 및 validation recall이 더 안정적으로 상승함을 확인하였다.

그럼에도 불구하고, 학습된 DPR 모델의 성능은 최종적으로 사전 학습된 성능이 우수한 Dense 모델을 그대로 사용했을 때와 비교해 유사하거나 소폭 낮은 수준에 머물렀다. 이는 in-batch negative에 easy sample이 다수 포함되어 학습이 충분히 강하게 이루어지지 못한 점과, 문서 pool 자체에 hard negative가 제한적으로 포함된 구조적 한계 때문으로 판단하였다. 이러한 분석 결과를 바탕으로, DPR은 단독 Retriever로 사용하기에는 성능 개선 효과가 제한적이라고 판단하였으며, 최종 파이프라인에서는 Dense Retriever를 보조적 역할로만 활용하거나 제외하는 방향으로 결정하였다.

이후 Sparse Retrieval의 안정성을 유지하면서 의미적 다양성을 보완하기 위해 KoE5와 KURE 기반 Dense Retriever를 비교 검토하였고, 문서 단위 임베딩의 일관성과 Retrieval 안정성 측면에서 KURE가 본 태스크에 보다 적합하다고 판단하여 최종 Dense Retriever로 선택하였다. Sparse(BM25)와 Dense Retriever의 점수는 정규화 후 가중합 방식으로 결합하여, 키워드 기반 정확성과 의미 기반 확장성을 동시에 반영하고자 하였다.

마지막으로, Hybrid Retrieval이 확보한 후보 문서 집합이 후보 문서를 넓게 확보하는 데에는 유리하나 상위 Top-k문서의 순위 품질이 충분히 정밀치 않다. 이에 상위 문서 순위의 정밀도를 개선하기 위해 쿼리-문서 쌍을 동시에 인코딩하여 문맥적 관련성을 직접 평가하는 Cross-Encoder 기반 Reranker(BAAI/bge-reranker-v2-m3)를 도입하였다.

Reader 입력이 Top-K로 제한되는 환경에서는 정답 문서를 포함하는 것뿐 아니라, 이를 상위에 배치하는 것이 중요하다고 판단하였기 때문이다. 실험은 `test_hybrid_rerank_recall.py` 스크립트를 통해 수행되었으며, Hybrid Retriever → Reranker → Recall 계산의 전체 파이프라인을 검증하였다. 실험 결과, Reranker 적용 이후 Recall@K 전 구간에서 가장 높은 성능을 기록하였으며, 특히 Recall@10 기준에서 정답 문서를 상위로 끌어올리는 데 결정적인 개선 효과를 확인하였다.

3.4 Reader

3.4.1 모델 리서치

다양한 도메인에 걸친 문서에서 질문에 대한 정답 구간을 정밀하게 추출해야하므로 한국어 고유의 문법적 구조와 의미 정보를 깊이 있게 이해하고 있는 한국어로 이루어진 데이터로 사전학습된 모델을 Reader의 백본으로 채택하여 실험을 수행하였습니다.

- RoBERTa

model	Eval/EM	Eval/F1
Klue/BERT-base	56.25	65.80
Klue/RoBERTa-base	64.16	72.06
Klue/RoBERTa-large	67.5	75.36

BERT-base에 비해 RoBERTa 모델들은 동적 마스킹이라는 더 최신화된 방법으로 데이터를 학습하여 한국어 문맥을 더 잘 이해하기에 성능이 더 높게 나왔다. 또한 파라미터수가 0.3b인 large 모델이 0.1b인 base 모델보다 더 좋은 성능을 보여 RoBERTa-large 모델을 추후 실험의 base 모델로 설정하였다.

- RoBERTa 기반 모델

model	Eval/EM	Eval/F1
uomnf97/klue-roberta-finetuned-korquad-v2	71.25	80.0
oceann315/roberta-large-korquad-v1	72.50	80.59
HANTAEK_roberta-large-korquad-v1-qa-finetuned	70.83	80.27

roberta 모델을 기반으로 qa task에 맞게 파인 튜닝한 모델들을 허깅페이스에서 찾아 실험을 진행하였다. general 한 모델인 RoBERTa 모델보다 korquad 데이터로 파인튜닝한 모델들이 더 좋은 성능을 보였다. 이중에서도 korquad2로 학습한 모델(uomnf97)보다 korquad1로 학습한 모델(oceann315)이 더 좋은 성능을 보인 것으로 보아 복잡한 데이터인 korquad2 보다 상대적으로 단순한 데이터인 korquad1 형태가 우리의 대회 데이터와 더 유사하다고 생각하였다.

3.4.2 기타 모델들

model	Eval/EM	Eval/F1
monologg/kobigbird-bert-base	60.41	70.48
team-lucid/deberta-v3-base-korean	51.6	62.4
team-lucid/deberta-v3-xlarge-korean	47.9	59.3
team-lucid/deberta-v3-xlarge-korean (parameter-tuned)	57.9	67.0

kobigbird 모델은 bert 모델보다 한번에 읽을 수 있는 토큰의 수가 많아 대회의 데이터처럼 큰 context에 대해 좋은 성능을 보일 거 같아 실험을 진행했지만 RoBERTa 모델만큼 성능이 나오지 않았다.

허깅페이스에서 deberta 모델이 같은 크기의 roberta 모델보다 qa 태스크에서 더 좋은 성능을 보인다는 것을 확인하여 실험을 진행하였으나 가장 안 좋은 성능을 보였다. 하이퍼파라미터의 문제인 것 같아 다른 파라미터들로도 진행을 하였는데 가장 좋은 지표도 Roberta 기반 모델에 크게 미치지 못해 Roberta 기반 모델을 사용하기로 결정하였다.

3.4.3 하이퍼파라미터 튜닝

대회에 사용할 모델을 roberta 기반 모델로 정한 뒤 wandb 를 이용하여 하이퍼파라미터 튜닝을 진행하였다. 나중에 앙상블을 고려해 리서치해 찾은 모든 모델들을 튜닝하기로 결정하였다.

모델	기본 Eval (EM / F1)	튜닝 Eval (EM / F1)
klue/roberta-large	67.5 / 75.36	70 / 78.45
uomnf97/klue-roberta-finetuned-korquad-v2	71.25 / 80.0	72.5 / 81.05
monologg/kobigbird-bert-base	60.41 / 70.48	64.16 / 73.19
oceann315/roberta-large-korquad-v1	72.50 / 80.59	72.91 / 81.23
monologg/koelectra-base-v3-finetuned-korquad	59.58 / 68.75	60 / 67.32
HANTAEK_roberta-large-korquad-v1-qa-finetuned	70.83 / 80.27	72.08 / 80.92

하이퍼파라미터 튜닝을 통해 Eval/ EM 스코어를 소폭 상승시킬 수 있었다.

3.4.4 TAPT 적용

본 연구에서는 roberta를 베이스라인 모델로 채택하고, 모델의 도메인 적합성을 높이기 위해 TAPT(Task Adaptive PreTraining)전략을 도입하였습니다. TAPT는 제시된 논문에서 RoBERTa 모델의 성능을 유의미하게 향상시킨 것이 입증된 바 있어, 본 대회의 ODQA 태스크에서도 긍정적인 효과를 거둘 것으로 판단하였습니다. 구체적으로는 타겟 도메인인 학습 데이터셋의 context와 질문을 활용하여 MLM(Masked Language Modeling) 방식으로 모델을 추가 학습시켰습니다.

모델	Eval (EM / F1)	Public (EM / F1)
klue/roberta-large	67.5 / 75.36	41.25 / 52.17
klue/roberta-large(TAPT)	66.2 / 73.98	40.83 / 51.78

TAPT을 적용한 뒤 오히려 성능이 감소하였다. 이에 대해 ODQA 태스크는 하나의 도메인에 대한 질문의 답을 생성하는 것이 아니라 다양한 도메인에 대한 질문의 답을 생성하기 때문에 TAPT가 오히려 ODQA 성능에 도움이 안되고 문맥을 추론하는 언어 모델링쪽으로 가중치가 편향되었을 수 있다고 생각하였다.

따라서 2차 실험으로 roberta 모델을 korquad 데이터로 학습시킨 uomnf97모델도 TAPT 실험을 진행하였다.

모델	Eval (EM / F1)	Public (EM / F1)
uomnf97	71.25 / 80.0	44.17 / 56.42
uomnf97(TAPT)	70.25 / 80.0	44.17 / 56.42

uomnf97에도 TAPT를 적용한 결과 성능이 감소하였다. korquad 데이터로 학습시킨 모델도 감소한 것으로 보아 TAPT가 이번 대회 성능에 도움을 주지 못할 것 같아 사용하지 않기로 결정하였다.

3.4.5 Curriculum Learning

retrieval의 성능을 끌어올린 뒤 reader 쪽에서 병목이 발생하는 것으로 판단이 되어 reader 쪽에서 성능을 올릴 수 있는 방법으로 curriculum learning을 생각하였다. reader 모델은 train때는 gold context만을 가지고 답을 찾으며 학습이 진행되지만 test때는 retrieval에서 concat한 top k개(대부분 k=10)의 context에서 답을 찾기 때문에 학습보다 test에서 더 성능이 떨어진다고 생각하였다. 이를 보완하기 위해 curriculum learning 방식을 활용해 1epoch에서는 기존 train_dataset을 활용해 gold context에서 답을 찾고 2~3 epoch에서는 negative sample과 gold context를 셔플한 dataset로 학습을 시켜 inference와 비슷한 환경으로 학습이 될 수 있도록 하였다.

모델	Public EM	Public F1
base	65.42	75.70
Curriculum Learning	66.67	78.66

예상대로 inference와 유사한 환경에서 test를 진행한 결과 모델 성능이 개선되었다.

3.5 Ensemble

3.5.1 Weighted Soft voting

각 모델이 inference 시 예측한 start, end에 대한 logit값을 weighted soft voting 방식으로 구현하고자 하였다. 처음에는 roberta 모델만을 사용하지 않고 roberta 모델과 koelectra, kobigbird 모델을 합치려고 하였으나 서로 다른 토크나이저를 사용할 경우 최종 logit의 shape이 달라 voting이 진행되지 않는 이슈가 발생하였다. 그래서 가장 성능이 우수했던 robert-large, hanteck, uomnf97, oceann315(가중치 순) 4개의 모델을 soft weighted ensemble하는 것으로 결정하였다.

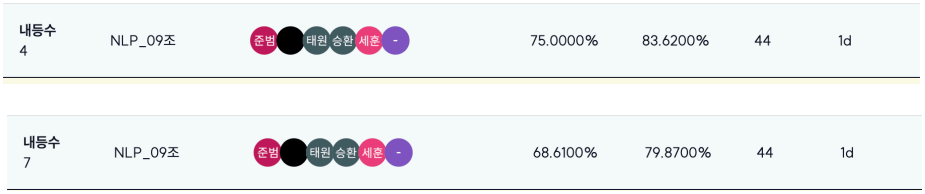
가중치	Public (EM / F1)	Private (EM / F1)
1:1:1:1	75.00 / 83.62	68.61 / 79.87
2:3:3:3	74.58 / 83.31	68.61 / 79.65
1:3:3:3	75.00 / 83.52	68.61 / 79.87

앙상블이 대회 마지막 날 구현되어 많은 실험을 하지 못했다. 그래서 최적의 모델 조합이나 가중치를 찾지 못하였다. 앙상블 코드 구현의 방식 때문에 roberta 이외의 모델을 사용하지 못해 일반화 성능이 떨어져 private에서 점수가 많이 감소한 것 같아 아쉽다.

---

4. 프로젝트 결과

Public - 4등 😀😀😀 , Private - 7등 😭😭😭



<그림8,9> Public, Private 최종 점수

---

5. 프로젝트 회고

5.1 잘한 점들

- 피어세션 시간 이외에도 많은 시간을 회의에 시간을 써서 자칫 느려질뻔한 프로젝트 진행속도를 지속적으로 점검하고 조율하였다. 회의와 Slack을 통해 각자가 수행한 실험 결과와 고민을 공유하려는 노력이 이루어졌다.
- 첫 프로젝트임에도 불구하고, 팀원 전원이 리더보드 마감 직전까지 성능 개선을 목표로 적극적으로 실험을 진행하였다.
- 실험 환경 설정, 데이터 파일 공유 구조 등을 초기에 정리하여 비교적 빠르게 실험을 시작할 수있었다.
- MRC, Retriever-Reader 구조, dense/sparse 등 처음 접하는 개념이 많았음에도 불구하고, 팀원들이 각자 학습한 내용을 공부하고 열심히 이해 수준을 맞추기 위해 노력하였다.
- 프로젝트 결과로 퍼블릭 리더보드 기준 3위를 기록하였다.
- 반복적인 실험과 회고 과정을 통해 팀 차원에서 많은 학습과 성장을 경험할 수 있었다.

5.2 시도했으나 잘 되지 않았던 것들

- 생성형 모델(Qwen-7b)을 학습해 validation 결과를 봤을 때 숫자/연도같이 roberta기반 모델이 잘 맞추지 못하는 문제를 맞추는 것을 확인했으나 시간이 부족해 생성형 모델과 추출형 모델을 앙상블하는 작업을 진행하지 못했다.
- retriever의 결과에 rerank적용을 했으나 reader와 연결하는 부분에서 문제가 생겨 디버깅하는데 시간을 써 결과가 대회 마감 이후에 나왔다.

5.3 아쉬웠던 점들

- **협업 및 코드 관리**
  - GitHub 기반 버전 관리가 체계적으로 이루어지지 않아, 후반부에 팀원 간 코드 실행 환경이 상이해지는 문제가 발생하였다.
  - 명확한 Git 컨벤션과 협업 규칙을 초기에 정하지 못해 코드 병합 및 관리에 어려움이 있었다.
- **실험 진행 상황 파악**
  - 팀원 간 전체 파이프라인에 대한 이해 수준이 상이했으며, 실험 진행 상황 공유가 충분히 구체적이지 못했다.
  - Notion, W&B 등을 활용한 실험 결과 아카이빙이 체계적으로 이루어지지 않아 팀 차원의 진행 상황을 파악하는 데 한계가 있었다.
- **목표 설정 및 운영 방식**
  - 프로젝트 시작 시 팀 차원의 목표와 우선순위가 명확히 합의되지 못하였다.
  - 회의 및 스크럼 진행 방식이 일정하지 않아 효율성이 다소 저하되었다.

5.4 프로젝트를 통해 배운 점 또는 시사점

- GitHub Issue를 중심으로 작업과 실험을 기록하는 방식이 협업과 실험 관리에 매우 중요하다는 점을 체감하였다
- 규모가 있는 작업일수록 개인 판단에 앞서 팀과 충분히 공유하고 논의하는 과정이 필요함을 배웠다.
- 개인 단위의 작업과 팀 단위의 작업은 접근 방식이 다르며, ML 프로젝트에서는 특히 설계 및 실험 관리의 중요성이 크다는 점을 인식하였다.
- 향후 프로젝트에서는 GitHub Issue, Slack, Notion 등을 활용한 실험 기록 자동화 및 협업 구조 개선을 시도할 계획이다.

---

6. 개인 회고

강다형_T8002	먼저 개인적인 회고부터 적어보겠습니다. 프로젝트 초기부터 가장 고민했던 지점은 실험 관리/재현, 모듈화에 적합한 ML 파이프라인 설계였습니다. 팀의 생산성 향상을 위해 고속도로와 같은 인프라를 깔아두고 싶었습니다. 처음엔 전체적인 그림이 잘 그려지지 않아서 막막하기도 했습니다. 그러나 이번 기회에 한번만 정리해둔다면 후속 프로젝트에서도 도움 될 것이란 확신이 있어서 나름의 시간을 할애했습니다. 이 과정에서 개인/팀 관점에서 소기의 성과를 거둔 것 같아 만족스럽습니다. 또 ML 엔지니어링에 필요한 기술과 요구 역량에 대해서도 깊이 생각해볼 수 있었습니다. 몇 가지 아쉬웠던 점(설계, 디자인 패턴, 리소스 관리, 협업 측면 등)은 후속 프로젝트에서 더 잘 보완할 수 있을 것 같습니다. 다음으로 제 시선에서 바라본 팀 차원의 회고입니다. 우선 모든 팀원이 협업의 중요성을 인식하고 '더 좋은 협업을 하자'라는 공감대가 형성됐다는 점에서 매우 고무적이라고 생각합니다. 별개로 스스로에 대한 반성을 몇 자 적어보자면. 팀에서 꺼내고 싶은 이야기가 있었음에도 스스로 확신이 없어 꺼내지 않았던 것, 전체 프로젝트 관리가 미흡했던 점 등을 적고 싶습니다. (팀 차원에서 함께 이야기했던 것들은 팀 회고록에 서술했습니다.) 마지막으로 프로젝트를 진행하며 겪은 시행착오에 대한 이야기입니다. 개별 실험에서부터 협업 과정까지 모든 과정에서 저를 포함한 모든 팀원들이 다양한 종류의 시행착오를 경험했습니다. 인상 깊었던 것은 모두가 이를 일회성 경험으로 끝내지 않고 더 나은 다음을 위한 발판으로 활용했다는 점입니다. 이러한 관점에서 저희 팀의 잠재력이 무척 높이 평가하고 프로젝트를 거듭할수록 더 우수한 퍼포먼스를 낼 것이라 확신합니다. 이 회고를 빌어 프로젝트 마감 직전까지도 집중력을 발휘하며 열심히 참여해준 모든 팀원에게 고맙다는 인사 꼭 전하고 싶습니다. 앞으로 남은 프로젝트들도 '함께' 잘 헤쳐나가면 좋겠습니다!
김민석_T8025	개인적으로 retriever에서 dense를 사용하면 크게 성능을 올릴 수 있다고 생각해서 이 부분을 다른 팀원분과 두명 이서 해보기로 했는데 하면 할수록 어려운 작업이고 해야 할 것이 점점 늘면서 큰 작업이었기 때문에 초반 작업이 너무 오래 걸려서 프로젝트 초반에 바로바로 성능을 테스트 할 수가 없었습니다. 특히 DPR학습에 필수적인 negative sample을 만들려고 하는 과정에서 이해 부족과 시간의 부족으로 좋은 품질을 가진 sample을 만들지 못해서 DPR학습에 실패했습니다. 만약 쉬운 것부터 시도했더라면 계속 성능을 봐가면서 기능을 더 빠르게 발전시켜나갈 수 있었을 것 같았기 때문에 다음 프로젝트부터는 당장 할 수 있는 것부터 시도 해보는 것을 우선순위로 잡는 것을 목표로 해볼 것 같습니다. 또한 사용했던 okt와 BM25가 너무 느렸기 때문에 그 대안으로 Mecab이란것이 있어서 설치를 해보려고 했지만 잘되지 않았기 때문에 만약에 이것을 설치할 환경이 제대로 갖추어진다면 더욱 빠르게 실험을 진행할 수 있었을 것 같습니다. 하지만 생성된 negative sample은 이후에 reader 학습 시에 정답 context에 붙여주는 passage로서 활약을했고 shuffle까지 생각하고 적용시켜 성능 향상의 결과로 이어졌던 부분이 나름 dpr에서 완전히 실패했던 것만은 아니었구나라는 생각이 들었습니다. 개인적으로 이번 프로젝트가 거의 깃허브를 이용한 첫 협업인데 처음부터 제대로 사용하지 못하고 중간부터 꼬이는 바람에 pull push를 제대로 못하게 됐었기 때문에 다음 프로젝트에는 좀 더 버전을 통일하고 브랜치를 잘 만들어 pr을 적극적으로 활용해보는 것을 목표로 하고 있습니다.
김세훈_T8037	이번 프로젝트는 MRC 파이프라인을 처음부터 끝까지 경험해본 첫 팀 프로젝트였다. 프로젝트 초반에는 Retriever-Reader구조, 데이터 특성부분등 개념이 익숙하지 않아 어디부터 손을 대야 할지 막막했던 기억이 난다. 그래서 개인적인 목표를 전체 파이프라인에 대한 이해를 높이고 팀프로젝트에서의 경험을 쌓는 것으로 설정하였다. 그중 데이터 분석, 전처리, negative sampling 설계, 오류 분석을 중점으로 프로젝트에 참여했다. 실험 과정에서 생각했던 것(예를 들어 low1개, mid1개, high3개 클러스트링으로 negative sampling등)과 달리 성능이 개선되지 않거나 오히려 하락하는 경우도 있었는데, 이 경험을 통해 단순히 복잡한 기법을 적용하는 것보다 정확하고 의미있는 분석이 더 중요하구나라고 생각이 들었다. 다만 가설을 세우고 실험을 진행 한 이후, 결과를 베이스라인과 단순히 비교하는선에서 멈췄다는게 아쉬웠다. 물론 시간적이 제약도 있었지만 멘토님이 왜 그런 결과가 나왔는지 여러번 파고들어서 분석하는 과정이 중요하다 했는데 이점을 충분히 수행하지 못 했다. 첫 팀 프로젝트였던 만큼 아쉬운 점도 많았지만, 실험 결과를 공유하고 실패를 함께 정리하며 팀 단위로 학습해 나가는 경험 자체가 큰 의미였다고 느낀다.
박준범_T8083	이번 MRC 프로젝트는 개인적으로 처음으로 대회 형식의 머신러닝 프로젝트를 끝까지 수행한 경험이었다. Retriever–Reader로 구성된 ODQA 파이프라인을 직접 설계하고 실험하면서, 단순한 모델 성능 개선을 넘어 문제를 정의하고, 실험을 통해 가설을 검증하며, 결과를 해석하는 전 과정을 경험하는 것을 개인적인 학습 목표로 삼았다. 프로젝트 초반에는 Retrieval과 Reader 전반에 대한 이해가 충분하지 않았고, 대회 환경이나 데이터 특성을 주도적으로 분석할 수 있는 수준에도 이르지 못했다. 초기에는 Reader 모델 탐색이라는 개별 테스크를 맡아, 한국어 사전학습 모델들을 중심으로 후보 모델을 조사하고 베이스라인 실험을 반복 수행하는 데에 머물렀다. 이 과정에서 모델 구조나 성능 수치 자체에는 익숙해졌지만, 해당 선택이 전체 파이프라인에서 어떤 의미를 가지는지까지는 깊이 있게 연결하지 못했고, 주어진 테스크를 다소 수동적으로 수행하고 있었다는 한계를 느꼈다. 이후 Retrieval 관련 테스크에 참여하게 되면서 비로소 ODQA 시스템 전체를 하나의 흐름으로 바라보게 되었다. Reader 성능이 Retrieval 결과에 의존한다는 점을 실험적으로 확인하면서, 단순히 모델을 교체하는 접근보다 Retrieval 단계의 설계와 튜닝이 중요할 수 있음을 인식하게 되었다. 이를 계기로 BM25 계열 Sparse Retrieval을 중심으로 형태소 분석기, 알고리즘 변형, 파라미터 조합에 따른 Recall 변화를 비교하며 Retrieval 단계가 최종 성능에 미치는 영향을 정량적으로 분석하는 방향으로 개인적인 관심과 실험 범위를 확장하게 되었다. Dense Retriever와 Reranker 실험을 진행하면서, 처음에는 Retrieval이 전부일 것이라고 생각했지

	만 실제로는 그렇지 않다는 점을 경험했다. Retrieval 단계에서 정답 문서를 포함시키는 것이 중요했지만, 동시에 Reader 모델이 해당 문서에서 정확한 정답을 추출하지 못하면 성능 향상으로 이어지지 않았다. 이 경험을 통해 ODQA에서는 Retrieval과 Reader 어느 한쪽만을 강조하기보다, 두 단계를 함께 고려하며 설계하고 실험하는 것이 중요하다는 인식을 갖게 되었다. 다만 이번 프로젝트에서 가장 아쉬웠던 점은 태도와 실험 공유 방식이었다. 실험 결과가 좋지 않았을 때 이를 기록하고 공유하지 않았고, 성능이 개선된 결과만 뒤늦게 정리하는 경향이 있었다. 그 결과, 제출 마감 직전에 새로운 방법을 기존 코드에 적용하려다 충분한 검증과 디버깅 시간을 확보하지 못했고, 최종 결과를 확인하지 못한 채 마무리되는 상황을 맞이했다. 이번 경험을 통해, 머신러닝 프로젝트에서 중요한 것은 단순히 좋은 성능을 내는 것이 아니라 과정을 투명하게 기록하고, 실패한 실험까지 포함해 축적하며, 이를 기반으로 의사결정을 하는 태도라는 점을 분명히 인식하게 되었다. 다음 프로젝트에서는 실험 성공 여부와 관계없이 모든 시도를 구조적으로 기록하고, 성능이 낮은 결과에서도 명확한 교훈을 도출하는 것을 목표로 삼고자 한다.
이태원_T8163	이번 프로젝트는 ODQA 태스크로 retrieval, reader 투 스테이지의 모델의 성능을 개선하는 것이었습니다. 저는 reader 모델 성능 개선을 중점으로 프로젝트에 참여했습니다. 처음에는 retrieval의 성능이 대회 점수의 upper bound라는 인식때문에 reader쪽에서는 모델 탐색과 하이퍼파라미터 튜닝정도로 끝날 줄 알았지만 retrieval 성능이 개선되었지만 오히려 reader의 성능이 감소하는 것을 확인하였습니다. 그래서 reader의 성능이 올리기 위한 방법을 inference와 train의 환경을 통일하는 것이라 생각했습니다. train에서는 gold context만을 가지고 학습을 진행하지만 inference에서는 10개의 context가 concat된 문서에서 답을 찾아야하기 때문에 reader 모델 입장에서 쉬운 방법으로 학습을 시켜놓고 실제로는 어려운 문제를 풀게하는 것이었습니다. 이를 일반화하기 위해 curriculum learning을 고안하였고 실제로 gold context만을 가지고 답을 찾는 eval에서는 성능의 큰 개선이 없었지만 리더보드 상에서는 유의미한 성능 개선이 있었습니다. 이런 경험을 통해 데이터나 프로젝트 상황에 관한 인사이트를 가지고 이를 실험을 통해 점수로 증명하는 일련의 과정이 굉장히 신선했고 재밌게 다가왔습니다. 팀적인 회고로는 팀원 모두 ml 팀 프로젝트에 처음 참여하면서 각자 코드가 통일되지 않고 실험이 공유되지 않는 문제가 생겼습니다. 각자 개발에 참여하여 결과가 생기는 개발 프로젝트가 아니라 열심히 실험을 해도 성능이 개선이 되지 않는다면 아웃풋이 생기지 않는 ml 프로젝트 특성상 모두가 성능이 개선이 되지 않으면 각자의 실험에 대해 공유가 되지 않아 문제가 생긴 것 같습니다. 또한, 각자 분업화된 일을 하다보니 서로가 서로의 task에 대한 이해가 부족해 최종 파이프라인을 합치는 과정에서 문제가 생길 수밖에 없었습니다. 따라서 이런 문제를 해결하기 위해서는 데일리스크럼과 피어세션을 활용해 데일리 스크럼에서는 어제 무슨 실험을 해서 결과가 어땠는지, 오늘은 어떤 실험을 할건지 공유하고 피어세션때는 각자 실험한 결과나 코드에 대해 리뷰를 진행해 팀원끼리의 전체적인 프로젝트 이해도 수준을 끌어올리기로 회고를 진행하였습니다. 다음 프로젝트에서는 체계화된 정리와 팀적인 협업을 목표로 참여하겠습니다.
허승환_T8218	이전의 개인프로젝트에서는 어떤걸 실험할지 자신이 판단하고 만지는게 가능했는데 팀으로 진행하니 제가 해보려고한 부분이 다른 팀원이 먼저 진행하고 있었고 서로 진행하는 부분이나 이게 어떤기능인지 어떻게 작동을 시키면되는지 알지 못해서 조금 붕떠있는 느낌을 받았습니다 그래서 다음 프로젝트를 진행하면 자신이 개발한 거에 대해서 더 자세하게 이야기하는 자리를 만들 것이고 slack, zoom, 카톡으로 진행을 했지만 그래도 정리를 할 수 있는 공간이 더 있으면 좋다고 생각이 들어서 팀원 notion을 이용해볼 것입니다