NAVER CONNECT | boostcamp AI Tech

NLP 기초 프로젝트

Open-Domain Question Answering (Machine Reading Comprehension)

NLP 15팀

김진재(팀장), 박규태, 윤선웅, 이정민, 임한택

1. 개요

이 문서는 2024년 9월 30일 (월)부터 2024년 10월 24일 (목)까지 진행한 Open-Domain Question Answering 프로젝트에서 NLP 15팀이 수행한 결과의 Wrap-Up 보고서이다.

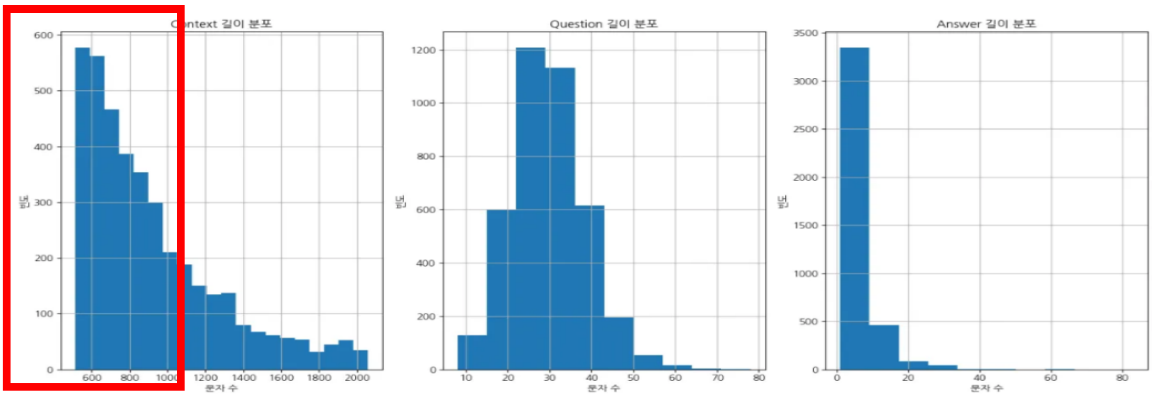
1.1. 개인별 역할

팀 내에서 개인별로 다음과 같은 역할을 담당하였다.

팀원	역할
김진재	(팀장) Basline 코드 작성 및 개선, 프로젝트 매니징 및 환경 관리, 조사 전처리 알고리즘 개발, 새로운 접근 방법론 제안, 앙상블
박규태	데이터 특성 분석, EDA, Retrieval 구현, 비교 실험 및 개선 (Hybrid Search, Re-Ranking, Dense 등), Reader 모델 Fine-Tuning
윤선웅	KorQuAD 1.0 데이터 증강, 모델 Fine-Tuning, Reader 모델 개선 (CNN Layer 추가), Retrieval 모델 구현(BM25), 앙상블
이정민	데이터 증강 (AEDA, Truncation 등), Question Dataset Tune, KorQuAD Dataset Tune
임한택	EDA, Retrieval 모델 개선 (BM25Plus, Re-Ranking Hyperparameter 최적화), Reader 모델 개선 (PLM 선정 및 Trainer Parameter 최적화)

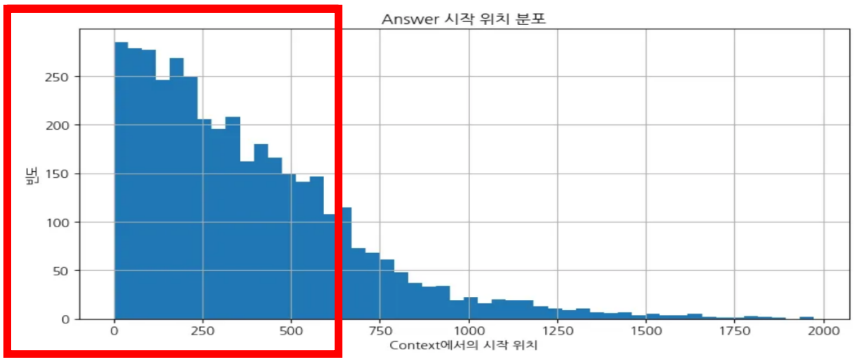
2. 데이터 분석

2.1. 길이 분포



대부분의 context가 1000자 이하로 집중되어 있어, 문맥을 슬라이싱하거나 중요한 부분만 남기는 전처리 전략이 필요하다고 가정하였다. 정보 손실을 최소화함과 동시에, 컴퓨팅 성능으로 인한 최대 길이 제한을 모두 고려할 수 있는 전략이 필요하다고 생각하며 모델링을 수행하였다.

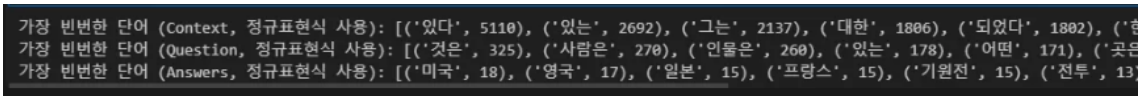
2.2. Answer Start 위치 분포



answers에는 answer_start가 나타난다. 이것은 질문에 대한 답을 Context에서 추출하기 위한 시작점을 알려준다. answer_start 위치는 대부분 문맥의 처음 부분(0~500자 사이)에 집중되어 있다. context의 초반부에 답변이 위치하는 경우가 많다는 것이다.

answer_start가 대략 650자 이후의 빈도는 급격히 낮아지며, 1000자 이상의 문맥에서 답변이 등장하는 경우는 매우 드문 것으로 확인된다. answer_start가 대부분 초반부에 위치하므로 모델이 초기에 문맥의 초반부에 더 집중할 수 있도록 하는 것이 효율적일 것이라고 생각했다. 예를 들어, 어텐션 매커니즘의 가중치를 초기 위치에 더 높이는 방식으로 기존 아키텍쳐를 변형할 수 있을 것이다.

2.3. 단어 출현 빈도



Question과 Answers에서 비교적 자주 등장하는 단어는 사람과 인물이다. 자주 출현하는 질문 유형에 모델이 더 집중할 수 있도록 전처리 전략을 세우는 것에 참고하였다.

2.4. Train 데이터에서 Context와 Question간의 키워드 유사성 분석

[진행 방법] Train Dataset만 불러와서, MeCab + StopWords 조합으로 tokenize_fn을 선언한다. 이후 TfidfVectorizer를 활용하여 토큰 별 TF-IDF를 만들어서 Context 별 토큰 중 값이 높은 5개의 토큰을 추출한다. 이 중 추출된 상위 5개의 토큰이 포함된 경우를 Count한다.

[결과] 3952개 중에서 2987개 포함

[결론] Context 와 Question의 키워드가 겹치는 경우가 많은 걸로 봐서, Context을 보고 Question을 만들었다고 생각했다. 이를 근거로 이번 프로젝트에서의 Retrieval 방식은 Sparse Embedding 방법이 모델 성능을 높이는 데 중요한 역할을 할 것이라고 가정했다.

3. 데이터 처리

3.1. 전처리 - Truncation

이 증강 기법은 'Cheap and Good Simple and Effective Data Augmentation for LowResource Machine Reading' 논문을 통해 알게 된 기법을 변형한 기법이다. 해당 기법은 하나의 Question에 대하여 Answer가 존재할 때, 정답 범위를 포함하는 Question Span을 좌우로 이동시켜 추가 학습 데이터를 생성한다. 본 논문에 영감을 받아 약간 변형하여 다음과 같은 연구를 진행하였다.

i. 데이터셋의 전체 길이 중 일부를 제거하는 Truncation 방법을 시도하였다. Truncation 된 Context의 길이가 512가 되도록 하였다.

ii. 증강을 사용한 데이터셋은 논문에서 가장 좋은 효율을 보인 전체의 70 ~ 80%를 random choice 한다.

iii. answer_start index을 중앙으로 하여, 전체 길이가 512가 되도록 왼쪽과 오른쪽 문장을 잘라낸다.

iv. 이렇게 나온 증강된 데이터를 원본 데이터와 Concat 하였고, 논문과 다르게 2중 fine-tuning을 하지 않고 1번만 학습하였다. 이것은 실험하는 대상 모델이 BERT와 같은 base model이 아닌 이미 사전 학습이 된 모델이기 때문에 2번의 학습은 오히려 과적합을 낳을 수 있다고 판단하였다.

Private 점수	EM	F1
원본	63.33	**74.49**
증강 모델	**65.42**	74.19

여러 번의 증강 결과 80%를 Random Choice를 통해 선별하고, Truncation 길이가 512가 되도록 증강을 하는 경우 평균적으로 EM와 F1 점수가 원본에 비해 1점 정도 향상하였다. 다만 70% ~ 80% 이상으로 데이터셋 개수를 더 늘릴 경우 과적합에 걸리게 되어 실제 train loss는 크게 낮아지지만, 반대로 Eval

과 Prediction 점수가 급격하게 낮아진다. 증강한 모델은 최종적으로 앙상블에 사용하는 후보 모델군으로 선택되었다.

3.2. 전처리 - PLM Fine-Tuning

기존 Korquad로 사전학습이 된 PLM에 다른 버전의 Korquad 데이터 셋을 증강 시키는 방법이다. 본 프로젝트에서 많은 모델을 korquad에 사전학습 된 모델을 사용하였는데, 각 모델에 서로 다른 Korquad 버전을 학습하여 모델의 Context 이해력을 끌어올리기 위한 방법이다. KLUE의 데이터셋은 korquad 1.0 버전과 완전히 동일하기 때문에 1.0에 대한 전처리 과정을 따로 필요하지 않았고, 반대로 2.0을 1.0 버전에 맞게 전처리 과정을 작업하였다. 또한, KorQuAD 2.0의 context는 html을 그대로 text로 변환하여, 1.0의 문장처럼 변형하기 위해 BeautifulSoup Html 태그 분석 라이브러리를 사용하여 전처리를 수행하였다.

또한 korquad 1.0 버전으로 변환된 데이터셋 중 일정 개수를 random choice하여 증강하였다. 아래 표는 1500개의 train 데이터를 추가로 증강한 결과 분석이다.

Private 점수	EM	F1
원본	**63.33**	**74.49**
증강 모델	61.67	72.39

실험 결과 증강이 점수를 떨어지는 것을 확인하였는데, PLM 모델이 Context에 대한 Depth가 충분히 학습되었거나, 2.0의 전처리 Context가 고르지 못하고 이질적이었기 때문이었다고 분석하였다.

3.3. 데이터 후처리 - 조사 LLM & 결과

Retriever과 Reader를 거친 Question에 대한 답변은, 생각보다 조사를 포함하고 있는 경우가 매우 많았다. 해당 조사를 제거하기 위해 LLM을 활용하였는데, 답변에 대한 조사를 제거할 것을 Prompt로 요청하고 LLM이 반환하는 제거된 결과를 최종 답안으로 제출하였다.

LLM의 예상치 못한 결과를 제어하기 위하여 Temperature를 0.1로 설정, Prompt Engineering을 통해 답변의 자유도를 최대한 관리하였다. 최종 Prompt에는 제시문과 답변의 예시, 그리고 답변의 형식 제약문을 담았으며 전문은 Github 코드에서 확인할 수 있다. 데이터 후처리를 통해 개선된 예시는 아래 표에서 확인할 수 있다.

질문	후처리 이전	후처리 결과
돌푸스에게 불특정 기간동안 하원이 잠시 쉬는 것을 건의 받았던 인물은?	대통령인 빌헬름 미클라스	**빌헬름 미클라스**
이유립씨가 1970년대 중반에 본인의 글을 기고하기 시작한 곳은?	'자유'지에	**'자유'지**
틸의 삼촌댁 방문을 걱정한 인물은?	그의 어머니	**어머니**
장하준 교수의 한국경제에 대한 생각에 대해 반론한 김상조 교수의 저서는?	<<종횡무진 한국경제>	**<<종횡무진 한국경제>>**

위와 같이 후처리된 결과를 Return하여 제출한 결과, 실제 Private 제출에서 유의미한 성능 개선이 있었던 것으로 확인하였다.

Private 점수	EM	F1
원본	61.67	73.10
증강 모델	**61.94**	**73.22**

4. 모델링

4.1. Reader 선정

ODQA는 한국어에 최적화된 모델을 사용해야 다양한 도메인에서 질문에 대한 답을 정확히 추출할 수 있다. 따라서 한국어가 특화된 사전학습 모델을 사용해야 한다고 생각했다.

[BERT 계열 vs RoBERTa 계열]

Model	에폭	EM (Eval)	F1 (Eval)
BERT-base	3	56.25	65.81
RoBERTa-base	3	64.17	72.06
RoBERTa-large	3	**68.33**	**77.40**

모든 환경을 동일하게 맞추고 학습 후 Evaluation을 기준으로 BERT와 RoBERTa 모델의 성능을 평가하였다. RoBERT는 BERT 기반으로 대규모 데이터에 대해 긴 훈련시간을 통해 성능을 개선한 모델로 일반화 능력이 우수하다. EM과 F1 점수를 통해 중요 단어나 문맥을 더 잘 파악할 수 있다고 판단해서 RoBERTa 계열을 리더 모델로 선정했다.

Base보다 Large의 파라미터 수, 헤드 수, 레이어 수, 히든 사이즈 수 등이 더 커서 EM과 F1이 더 잘 나왔다고 판단하였고, 프로젝트에서 진행되는 대부분의 실험은 Finetuning된 large 기반의 모델과 직접 Large기반의 모델을 Finetuning하여 사용하였다.

Model	EM (Public)	F1 (Public)	EM (Private)	F1 (Private)
RoBERTa-large + BM25	58.75	71.12	60.56	72.22
uomnf97/klue-roberta-finetuned-korquad-v2 + BM25	62.50	71.83	58.33	70.71
CurtisJeon/klue-roberta-large-korquad_v1_qa + BM25	65.83	75.41	**65.56**	**75.67**
HANTAEK/klue-roberta-large-korquad-v1 + BM25	**66.25**	**76.43**	61.39	71.91

[Model Customizing]

1. CNN Layer 추가

2020 삼성 SDS KorQuAD 1.0 성능 평가 솔루션에 따르면 classifier에 CNN layer들을 (CNN Conv1D(K=3) → Conv1D(K=1) → Relu → LayerNormalization으로 설계된 CNN Block을 5개) 추가한 CNN 아키텍처를 통해 성능 개선을 이뤄냈다는 자료가 있었다. 기존 RoBERTa 모델의 선형 레이어 출력에 CNN 레이어를 추가하여 연관 정보(문맥 정보)를 잘 학습하도록 시도하였다.

2. Head Customizing

Classification head를 다양하게 custom하여 모델의 예측 성능을 향상시키고자 하였다. 추가한 Layer들은 MLP, LSTM, Bi_LSTM, CNN이다. CNN을 제외한 Eval 점수가 가장 높았던 MLP를 제출하였으나 Public 리더보드 상 EM 점수가 하락하여 성능향상을 이뤄내지 못했다고 판단했다. 그 원인으로 RoBERTa 모델과의 구조적 부적합성, 이로 인한 과적합, 데이터셋 특징과의 불일치를 원인으로 추측한다.

Custom Layer	모델	에폭	EM(eval)	F1(eval)
MLP	CurtisJeon/klue-roberta-large-korquad_v1_qa	3	**71.5**	**79.12**
LSTM	CurtisJeon/klue-roberta-large-korquad_v1_qa	3	71.25	78.56
Bi_LSTM	CurtisJeon/klue-roberta-large-korquad_v1_qa	3	68.8	77.56
CNN	CurtisJeon/klue-roberta-large-korquad_v1_qa	3	70.0	78.83

커널의 크기를 3, 5, 7로 해보고, 딜레이션 비율도 1,2,4로 증가시키면서 3개의 Block도 시도해봤다. 또한 self-attention과 결합해서 문맥 정보를 더욱 강화시켜봤으며 LSTM을 사용해서 전체 문맥 정보를 잘 포착할 수 있도록 시도해보기도 했다. 하지만 결과적으로 CNN Block을 5개 사용하는 CNN 아키텍처가 가

장 성능이 좋았고 리더보드에서 EM을 약 1.5점 상승 시킬 수 있었다.

3. Dropout

최종적으로 확정된 5개의 CNN Block에서 과적합 방지를 위해 각 층마다 dropout을 적용하는 방법 (1)과 Fully Connected Layer 앞에만 두는 방법 (2) 두 가지를 고려했다.

Case	모델	EM(Public)	F1(Public)
(1)	CurtisJeon/klue-roberta-large-korquad_v1_qa	**66.67**	75.09
(2)	CurtisJeon/klue-roberta-large-korquad_v1_qa	66.25	76.01
X	CurtisJeon/klue-roberta-large-korquad_v1_qa	**66.67**	**76.46**

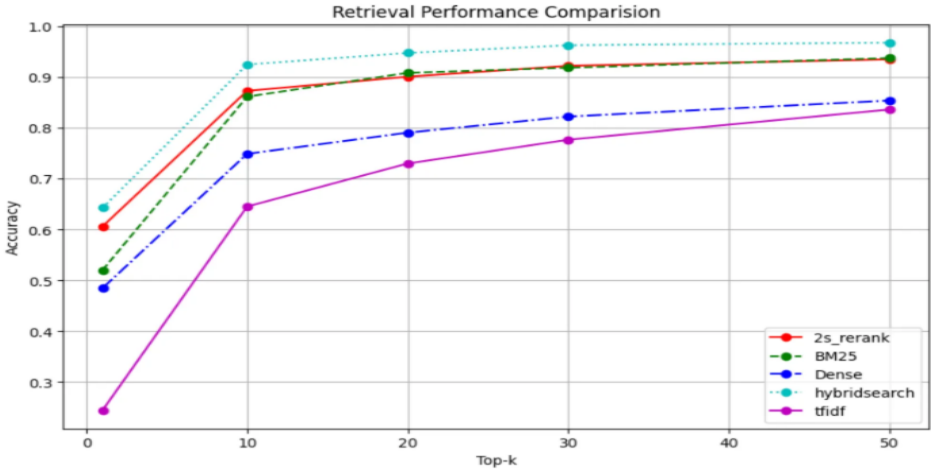
두 가지를 조합하면서 0.05부터 0.1까지 dropout을 적용하여 근소하게 Eval 에서는 성능이 좋아졌으나 리더보드에서는 성능이 떨어졌다. 결과적으로 Dropout 없이 CNN 레이어를 추가하여 모델을 학습시켰다.

4. Learning Rate

Hugging Face Trainer에서 상속받은 lr_scheduler_type은 기본적으로 'linear'로 설정되어 있으며 warmup_ratio도 0으로 되어있다. type으로 cosine, cosine with restart, inverted_sqrt, constant를 변경하며 비교해본 결과 cosine이 근소하게 linear보다 평가 성능에서 우수하게 나왔다. 이를 토대로 cosine과 warmup_ratio를 0.1로 설정하고 학습시켜본 것(1)과 linear(2)값으로 해본 결과는 다음과 같다.

Case	모델	EM(eval)	F1(eval)
(1)	CurtisJeon/klue-roberta-large-korquad_v1_qa	70.0	78.83
(2)	CurtisJeon/klue-roberta-large-korquad_v1_qa	**70.4**	**80.23**

4.2. Retriever 선정



훈련용 데이터셋 3952개 중에서 790개의 질문과 각 질문에 대응되는 지문을 정답으로 활용하였다. 각 질문들을 활용해 위키피디아 데이터에서 적절한 문서들을 top-k 만큼 retrieve 하였고 찾아온 문서들 안에 정답이 되는 지문이 존재하는 지 여부를 세고 정확도(Accuracy)로 성능을 평가하였다.

성능 비교를 위한 각 기법의 세팅으로 BM25을 활용할 때는 rank_bm25 라이브러리의 BM25Plus을 활용하였고 바닐라 BM25와 2s_rerank의 하이퍼 파라미터는 k1=1.7595, b=0.9172, delta=1.1490로 hybridsearch에서는 alpha=0.00601, k1=1.8377, b=0.5876, delta=1.1490로 진행하였다. 이 과정에서 일관된 성능 비교를 위해 DPR를 활용한 모든 기법에서 'intfloat/multilingual-e5-large-instruct' 모델을 사용했다.

4.2.1. TF-IDF

Train Data 에서 Context 와 question 간의 키워드 유사성 분석 결론으로 이번 프로젝트는 SPR 이 유리할 거라고 예상을 하였지만 5가지 기법 중에 가장 성능이 나오지 않았다.

4.2.2. BM-25, BM-25 Plus

BM25는 Bag of words(BoW) 모델에 기반한 확률적 언어 모델이다. 쿼리에서 등장하는 단어가 문서 내에서 얼마나 자주 등장하는지 평가하고(TF) 다른 문서들에서도 얼마나 자주 등장하는지(IDF)를 고려해서 가중치를 부여한다.

BM25Plus는 BM25의 기본 개념을 확장해서 빈도가 낮은 단어의 점수를 더 잘 반영한다. 이를 통해 term saturation 문제 등을 해결할 수 있을 것이라고 생각했다. BM25Plus에는 k1, b, delta 3가지 하이퍼파라미터가 있다. k는 단어가 여러 번 등장할수록 그 중요성을 얼마나 더 반영할지를 결정하는 값이다. 값이 크면 문서 내 단어의 반복이 더 중요하게 반영된다. b는 문서의 길이에 따라 단어의 중요성을 얼마나 조정할지를 결정하는 역할을 한다. 0에 가까울수록 문서 길이를 덜 중요하게 반영한다. delta는 문서에 단어가 전혀 등장하지 않더라도 최소한의 점수를 부여하여 검색 성능을 개선하는 역할을 한다. 값이 클수록 드문 단어가 더 중요해진다. 이 세 가지 파라미터를 통해 문서와 쿼리의 매칭을 좀 더 정교하게 조정할 수 있었다. 해당 방법론을 활용한 결과 리더보드 기준 EM 점수는 63.33 에서 64.17로 약 0.8점이 상승했다.

수동으로 조정하며 실험하는 데는 많은 시간이 소요되었기 때문에 Optuna를 활용해 최적의 값을 자동으로 탐색했다.(탐색한 최적의 값: k1=1.7595, b=0.9172, delta=1.1490) 이를 통해 리더보드 기준 EM 점수가 64.17에서 65.83으로 약 1.7점 향상되었다. 추가적으로 진행된 리트리버 비교 실험에서는 위 그래프에서도 볼 수 있다시피 데이터 특성 상 SPR이 단일 모델에 적용했을 때 가장 높게 나왔다. 심지어 Two stage Re-ranker와는 Top-1을 뽑아내는 성능을 제외하면, 거의 비슷한 성능을 보여주었다.

4.2.3. DPR

전상민 조교님이 진행해주신 '리트리버 고도화와 데이터 증강' 라는 주제로 진행된 오피스 아워에서 소개해주신 DPR 모델들의 벤치마크를 참조하여 모델을 선정했다. 'intfloat/multilingual-e5-large-instruct' 라는 모델로 진행하였고, 해당 실험에서는 밑에서 두 번째로 낮은 성능을 기록하였다.

4.2.4. Hybrid Search

해당 기법은 두 가지의 단일 기법들을 동원하여서 최종 결과에서 alpha 값(두 기법 중 최종 score에 얼마나 반영할지 결정)을 통해 해당 값들을 결합하고 최종 score로써 활용했다. 이번 실험에서는 BM25와 시멘틱 서치 중 DPR 방식을 사용했다. 또한 hybridsearch 에서는 위에서 언급한 alpha 와 BM25에서 사용되는 k1, b 등의 하이퍼파라미터가 존재했고 이들을 Optuna 을 통해서 DPR 과 결합 시에 최적의 성능을 낼 수 있는 값을 탐색했다. 그 결과로 나온 하이퍼파라미터로 실험을 진행하였다.

해당 실험 결과로는 좋은 성능을 보여주었던 SPR에 DPR의 결과를 더해줘서 문맥 정보를 더 잘 포착할 수 있었으며 가장 좋은 성능을 보여줬다.

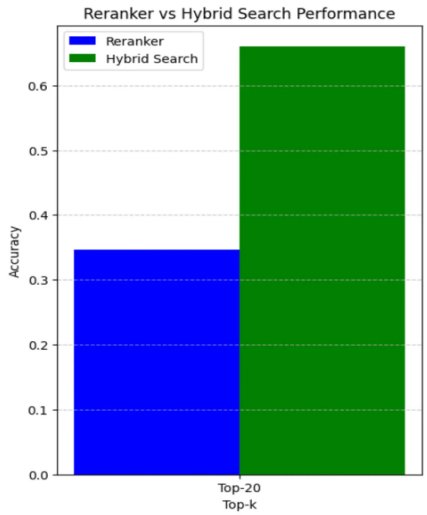
4.2.5 Re-ranker(2-stage)

Cross-Encoder를 사용한 재순위화는 다르게 bi-Encoder인 DPR을 재순위화에 활용했다. 따라서 1단계에서는 BM25Plus로 top_k개의 문서를 검색하고, 2단계에서는 top_k개의 문서에서 DPR (upskyy/bge-m3-korean)로 재순위화하여 최종 rerank_top_k개를 선택했다. 이때 DPR 모델은 Marker-Inc-Korea[3] 깃허브를 참고해서 사용했다.

Model Name	F1	Recall	Precision	mAP	mRR	NDCG
paraphrase-multilingual-mpnet-base-v2	0.2368	0.4737	0.1579	0.2032	0.2032	0.2712
KoSimCSE-roberta	0.3026	0.6053	0.2018	0.2661	0.2661	0.3515
...	...	...	...	...	...	...
bge-m3	0.4342	0.8684	0.2895	0.3436	0.3436	0.4757

실험 과정에서 1단계에서 top_k개를 몇 개로 지정하느냐에 따라서 2단계 순위화 되는 점수에 영향을 줬다. 그리고 2단계에서 rarank_top_k를 너무 낮거나 높게 잡을수록 reader 모델의 성능이 떨어지는 것을 확인할 수 있었다. 여러 실험을 통해 최종적으로는 top_k=500, rerank_top_k=20를 사용했다. 결과적으로 BM25Plus만 사용했을 때보다 EM이 약 1.5점 정도 상승하는 것을 확인할 수 있었다.

추가적으로 진행된 리트리버간 성능 비교 실험에서는 첫 번째 step 에서 뽑고자 하는 3 * k 개의 문서를 retrieve 하고 해당 문서들을 가지고 두 번째 step 에서 k 개의 문서를 retrieve 하는 방식으로 진행되었다. (top-20 개를 뽑기 위해서 첫 번째 단계에서 60개를 retrieve 후에 두 번째에서 최종적으로 20개를 retrieve) 그 결과 top-1 를 뽑는 task에서는 다른 타 기법에 비해서 매우 좋은 성능을 보여주었으며 최종적으로 하이브리드 서치 다음으로 높은 성능을 달성했다.



[Re-ranker와 Hybird Search 정확도 비교]

리트리버간 성능 비교 실험에서 가장 좋은 성능을 보여줬던 두 방법을 비교했다. 실험 방식은 top-20개의 문서를 뽑아 실제 쿼리와 대응되는 정답이 가장 확률이 높은 첫 번째 문서와 얼마나 많이 매칭, 맞추는 지를 카운트하고 이를 쿼리 수로 나눈 정확도(Accuracy)로 성능을 평가하였다. 하이브리드 서치는 위의 실험의 세팅과 동일하고 Re-ranker 는 첫단계에서 500개 다음으로 20개를 뽑는 과정으로 진행되었고 나머지는 동일하게 세팅되었습니다. 실험 결과 하이브리드 서치가 Re-ranker 에 비해서 약 두 배 가량 좋은 성능을 보여주었다.

5. 앙상블

예측 결과를 앙상블하여 최종 예측을 생성하는 soft voting 방식의 앙상블을 구현하였다. 모델의 예측결과가 담긴nbest_predictions.json 파일을 불러와 동일 질문에 대해 여러 모델의 응답을 수집한다. 답변들의 확률을 합산하여 가장 높은 점수를 받은 답변을 최종 답변으로 예측하는 방식을 선택하였다.

아래는 앙상블에 사용한 모델의 nbest_prediction의 json 파일명과 EM(Public)이다.

번호	모델+기법	EM(Public)
1	uomnf97+BM25+CNN	66.67
7	Curtis+CNN+dropout(only_FC_0.05)+BM25Plus	66.25
8	Curtis+Truncation	66.25
9	HANTAEK_hybrid_optuna_topk20(k1=1.84)	63.15
10	HANTAEK_hybrid_optuna_topk20(k1=0.73)	63.75
11	HANTAEK_hybrid_optuna_topk10(k1=0.73)	63.75
12	uomnf97+BM25	67.08
13	uomnf97+CNN+Re_rank500_20	67.08
14	curtis+CNN+Re_rank500_20	65.42
15	nlp04_finetuned+CNN+BM25Plus+epoch1	67.5

이론적으로 hybrid와 rerank를 사용하면 더 성능이 잘 나온다고 하였으나, Public에서는 BM25를 사용한 것이 더 성능이 잘 나왔다. 이를 통해 Public 데이터가 BM25에 조금 더 편향되어 있는 것을 파악하여 Private을 대비하여 더 다양한 기법과 모델 후보군을 최대한 다양하게 구성을 했고, Eval 기준 EM 68이상을 기준으로 삼았다.

6. 채택 방법론 및 결과

아래는 최종 채택된 방법론 및 결과이다.

최종제출	Ensemble	EM (Public)	EM (Private)
O	모델 7,8,9,10,11,12,13 1:1:1:1:1:2:3 앙상블 + 조사 LLM	**77.08**	71.11
O	모델 14,8,15,10,11,12,13 1:1:1:1:1:2:3 앙상블 + 조사 LLM	**77.08**	71.67
	모델 1,7,8,9,10,11,12 평균앙상블 + 조사 LLM	76.67	71.67
	모델 7,8,9,10,11,12,13 1:1:1:1:2:2:2 앙상블 + 조사 LLM	76.67	70.83
1st SOTA	모델 15,9,10,11,12,13 1:1:1:1:3:3 앙상블 + 조사 LLM	75.42	**74.17**
	모델 7,8,9,10,11,12,13 1:1:1:1:1:2:3 앙상블 + 조사 LLM(n=5)	75.42	71.67
	모델 7,8,9,10,11,12,13,14 1:1:1:1:1:1:2:2 앙상블 + 조사 LLM	74.58	71.11
2nd SOTA	모델 14,8,9,10,11,12,13 1:1:1:1:1:2:3 앙상블 + 조사 LLM	74.58	**72.22**

Public 성적 중 가장 높았던 EM을 기록한 77.08 성적을 제출하였으며, 해당 방법론은 Private 리더보드에서 EM 71.67을 기록하여, 전체 기록 중 최종 2등을 달성하였다. 제출하지 않은 결과를 포함하였을 때 최고 성적 1등과 2등은 각각 74.17, 72.22를 기록한 다른 앙상블 방법론이었다.

Rank	Team Name	Team Member	EM	F1	Entries	Final
My Rank 2	NLP_15조		77.0800%	83.8000%	129	4d
1	NLP_04조		77.5000%	86.5800%	135	3d
2	NLP_15조		77.0800%	83.8000%	129	4d
3	NLP_11조		76.6700%	85.4700%	42	3d
4	NLP_08조		76.2500%	84.4600%	88	2d
5	NLP_13조		72.5000%	80.9300%	93	2d

Rank	Team Name	Team Member	EM	F1	Entries	Final
My Rank 3	NLP_15조		71.6700%	83.6800%	129	3d
1	NLP_04조		72.2200%	82.8000%	135	3d
2	NLP_08조		71.6700%	82.1200%	88	2d
3	NLP_15조		71.6700%	83.6800%	129	3d
4	NLP_11조		71.6700%	82.3200%	42	2d
5	NLP_01조		67.2200%	77.8800%	41	2d

7. 결론

이번 프로젝트에서는 Truncation을 활용한 전처리, 다양한 변형을 시도한 Reader과 Hybrid Search를 활용한 Retreiver, LLM을 활용한 전처리를 이용하여 최종 EM 71.67을 기록하였다. 해당 성적은 Public Leaderboard 기준 16팀 중 2등, Privage Leaderboard 기준 16팀 중 2등 (공동 3팀)을 달성하였다.

8. 참고문헌

1. Karimi, A., Rossi, L., & Prati, A. (2021). AEDA: An easier data augmentation technique for text classification. Findings of the Association for Computational Linguistics: EMNLP 2021. arXiv preprint arXiv:2108.13230. https://doi.org/10.48550/arXiv.2108.13230
2. Van, H., Yadav, V., & Surdeanu, M. (2021). Cheap and good? Simple and effective data augmentation for low resource machine reading. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. arXiv preprint arXiv:2106.04134. https://doi.org/10.1145/3404835.3463099
3. https://github.com/Marker-Inc-Korea/AutoRAG-example-korean-embedding-benchmark
4. https://seokhee0516.tistory.com/entry/Open-domain-question-answeringODQA-3-Retrieval-Experiment-ElasticSearch엘라스틱-서치
5. https://velog.io/@autorag/어떤-한국어-임베딩-모델-성능이-가장-좋을까-직접-벤치마크-해보자
6. https://www.youtube.com/watch?v=ovD_87gHZO4

NAVER CONNECT | boostcamp AI Tech

NLP 기초 프로젝트

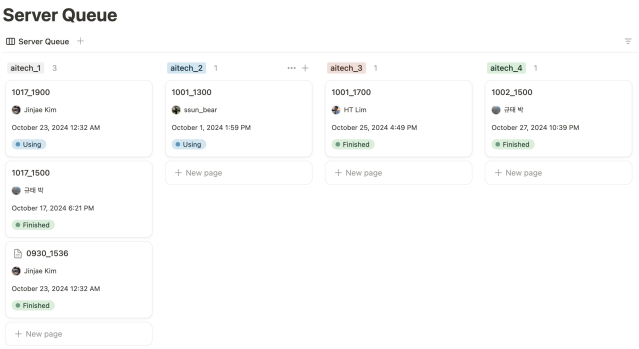
(개인 회고록)

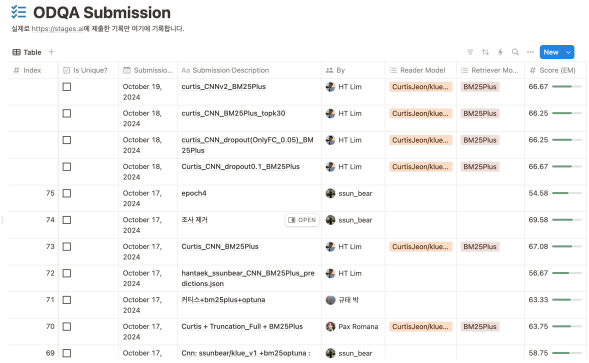
목 차

1. 김진재

팀 내 역할

본 Open-Domain Question Answering 프로젝트 (이하 ODQA)에서는 팀장으로서 프로세스 전체를 매니징하는 역할을 담당하였습니다. 매 회의의 목표를 설정 및 수립하고, 팀 내 그라운드 룰을 설정하여 프로젝트에서 해와야 할 일과 규칙을 설정하였습니다. 팀 내 서버를 관리하기 위해 새로운 방안을 제시하기도 하였으며, 이전 프로젝트에서 아쉬웠던 점인 역할 분배나 태스크 진행 정도 파악을 개선하기 위해 Notion과 Github를 적극적으로 활용하는 방안을 제안하였습니다. 작성된 코드와 팀 내 Git Flow를 관리하였으며, 모든 Issue와 Pull Request를 검수하고 Merge하는 담당 역할을 수행하였습니다. 그 외 프로젝트의 목표 달성을 위한 팀 내 서포팅을 위해 최선을 다했습니다.





학습 목표 달성을 위해 노력한 점

최초에는 LLM 모델만을 활용한 Training-Free 방법론을 고안하고 직접 제안하기도 하였으나, 해당 방법론의 성공 가능성이 생각보다 높지 않다는 점을 감안하고 조사 전처리 기능으로 활용하기 위해 방법론을 바로 전환하였습니다. 사용 결과 이전 답변에 비해 조사나 불필요한 수식어구 제거가 잘 이루어지는 점을 확인하였습니다. 그 외 모델을 활용한 앙상블 제안 등, 팀 내 다양한 창의적인 방법론을 제안하고 이를 실천에 옮기기도 하였습니다.

한계점 및 아쉬운 점

1. 프로젝트 과정 중간에 학회 일정으로 잠시 부스트캠프 휴가를 다녀오게 되었는데, 이 시기에 충분한 매니징을 하지 못한 점이 아쉬움으로 남습니다. 특히 이 시기부터 각자 작성해온 방법론을 통합하기 위해 많은 노력을 시도하여서, 학회 방문 이후 Follow-Up 하는 데에 비교적 많은 시간을 소모하였습니다.
2. 베이스라인 코드를 배포하였는데, 소통의 오류로 각자 다른 코드를 사용하여 모델링을 수행한 점이 아쉬웠습니다. 각자 다른 코드를 사용하다보니, 이러한 부분을 직접 맞추는 데에 생각보다 매우 많은 시간과 자원이 소모되었습니다.
3. 생각에서 그쳤던 제안 방법론이 많아서 아쉬웠습니다. 이렇게 하면 될 것 같다고 생각했던 것만큼 모든 것을 깊이 있게 파고들 수 없었던 것 같아서 조금은 아쉬웠습니다.

다음 프로젝트에서의 개선

다음 프로젝트에서는 아쉬운 점을 기반으로 좋은 프로젝트 성과를 위해 아래 내용을 고쳐나가고자 합니다.

우선, 프로젝트에서 진행되는 내용이 서로에게 더 적극적으로 공유될 수 있는 분위기를 조성하는 것이 필요하다고 생각합니다. 조금 더 깊이 있는 내용을 실시간으로 공유할 수 있도록 방안을 마련하고자 합니다. 또한, 다양한 방법론을 시도하기 위해서 시간 투자를 더 많이 할 것 같습니다. 프로젝트에서 적은 시간 투자가 아니었지만 개인적으로 더 많은 방법을 시도해보고 싶다는 아쉬움이 남아있어, 다음 프로젝트에서는 더 많은 시간을 투자할 것 같습니다.

2. 박규태

무엇을 하였나?

이번에는 요즘 핫한 RAG의 기반이 되는 테스크인 MRC / ODQA 주제로 컴피티션을 진행하였다. 이번 대회에서는 쿼리에 맞게 적절한 문서를 찾아오는 retriever 와 그 문서를 활용해서 적절한 답을 추출해내는 reader 모델 이렇게 크게 두 가지의 구조로 구성되어 있었습니다. 그래서 성능을 개선을 하기 위해서는 이 두 가지의 구조들의 성능이 모두 좋아야할 필요가 있었습니다. 저는 그 중에서 retriever 쪽에 흥미가 가고 이번 테스크에서 성능을 높이기 위해서는 핵심 역할이라고 생각하여 이에 집중하여 작업을 진행하였습니다. 가장 처음에 시작할 때는 막연히 전에 DPR을 활용해서 진행했던 프로젝트의 경험으로 해당 리트리버 방식을 사용하면 된다고 생각했습니다. 그러나 오피스아워 등에서 작업 진행 전에 데이터의 특성을 먼저 분석하는 것이 중요하다는 것을 듣고 주어진 쿼리와 정답간의 키워드 유사성을 분석하였습니다. 그 결과 쿼리와 답의 키워드가 겹치는 부분이 매우 많아서 SPR 이 오히려 성능이 좋다는 것을 알게되어 이를 중심으로 진행을 하였습니다. 그 후 SPR과 DPR 의 특성을 모두 적절히 활용해서 더욱 높은 성능을 뽑아내는 하이브리드 서치와 Re-ranker 을 구현해보고 이를 Optuna 을 통해서 최적화를 진행해보면서 유의미한 성능 향상을 볼 수 있었습니다. 또한 리트리버 간의 성능 비교 실험 등을 통해서 성능 측정을 진행하였습니다. 그 밖에도 Reader 모델의 파인튜닝을 통해서 성능 향상을 시도하였습니다.

좋았던 점

이번 대회에서는 전 대회에 조금 소홀했던 git 을 더욱 적극적으로 활요해서 각 기법 별로 브랜치를 파서 진행보았고 discussions, issues 등을 체계적으로 활용해 볼 수 있어 좋았습니다.

RAG에서 실제로 활용되는 문서를 retrieve 해오는 과정을 실제로 공부하고 구현해 볼 수 있어서 너무 좋았습니다. 또한 해당 리트리버들의 성능을 서로 비교해보고 데이터 특성에 따른 성능 차이도 분석해보면서 더욱 깊은 시야가를 가지게 된 거 같아 좋습니다.

아쉬웠던 점

이번 대회에서는 retrieval 쪽에 집중해서 진행을 하면서 비교적 reader 모델에 대해서는 조금 해이했던 것 같습니다. 다음에도 유사한 테스크를 하게 된다면 그 때는 reader 모델쪽에 무게를 두어서 진행해보고 싶습니다.

마찬가지로 이번에는 wandb을 활용해서 학습 로그와 기록을 남기면서 진행되었습니다. 그러나 이번에는 이런 기능을 프로젝트에서 적절하게 활용하지 못한 거 같아서 아쉬움이 남습니다. 다음 프로젝트에서는 wandb에 대한 기능을 어떻게 하면 더욱 잘 활용할 수 있을 지에 대해서는 더 고민이 필요할 거 같습니다.

3. 윤선웅

팀에서의 나의 역할

지난 프로젝트에서는 팀원들이 각자의 역할과 진행 상황을 명확히 공유하지 않아 혼란이 있었습니다. 이를 개선하기 위해 이번 프로젝트에서는 팀원들의 진행 상황을 수시로 체크하고, 각 태스크의 진행 척도를 명확히 파악하는 데 중점을 두었습니다. 이를 통해 팀원들이 자신의 작업을 더 잘 이해하고, 필요한 경우 지원할 수 있도록 하였습니다.

특히 이번 프로젝트에서는 리더 모델의 성능 향상을 위해 여러 가지 파인튜닝 기법을 시도했습니다. 외부 데이터셋이 허용된 만큼, 데이터 증강을 위한 방안을 강구하였습니다. KorQuAD 1.0과 2.0을 분석하여, 이 데이터셋이 우리의 모델에 어떻게 기여할 수 있을지를 탐구했습니다. 이에 따라, 다양한 질문 유형과 답변 형식을 고려하여 데이터를 준비하고, 이를 통해 모델이 보다 다양한 질문에서도 잘 작동할 수 있도록 하였습니다.

또한, 앙상블 기법을 활용하여 여러 모델의 예측 결과를 통합함으로써 최종 모델의 성능을 높이는 데 기여했습니다. 이러한 과정에서 팀원들과의 협력이 매우 중요했으며, 각자의 전문성을 살려 모델을 최적화하는 데 집중했습니다.

나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?

ODQA 시스템에서는 질문에 맞는 Passage를 찾는 Retriever 모델과 적절한 답을 추출해내는 Reader 모델의 두 가지 구조로 나뉘어 있습니다. 저는 Reader 모델에 더 많은 시간을 할애하여, 이 모델의 성능을 극대화하는 데 집중했습니다. 구체적으로는 다양한 아키텍처를 실험하고 하이퍼파라미터 조정을 통해 모델의 응답 정확도를 높이기 위한 여러 가지 방법을 모색했습니다.

리더 모델의 성능을 향상시키기 위해 2020 삼성 SDS KorQuAD 1.0 성능 평가 솔루션을 분석했습니다. 이 과정에서 CNN 레이어를 추가하고, 조사 제거 후처리 방법론을 제시하는 등의 방안을 구현하여 리더 모델의 성능을 높였습니다. 이러한 개선 작업은 실질적으로 리더보드 EM(Exact Match) 점수를 크게 향상시키는 결과를 가져왔습니다.

추가적으로, KorQuAD 1.0과 2.0을 활용하여 모델을 증강하는 파인튜닝을 진행했습니다. KorQuAD 데이터셋의 학습 데이터 최대 길이가 2064 토큰으로 설정되어 있었기 때문에, Passage 길이가 학습 데이터의 길이보다 크지 않은 KorQuAD를 활용하여 klue/roberta-large 모델을 파인튜닝했습니다. 이를 통해 다양한 질문 유형에 대한 모델의 응답 정확도를 높이는 데 기여했습니다.

마지막으로, 학습이 완료된 모델은 Hugging Face에 업로드하여 성능을 평가했습니다. 이러한 일련의 과정은 제 학습 목표를 달성하는 데 큰 도움이 되었으며, Reader 모델의 성능 향상에 기여했다고 자부합니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

ssunbear-kor...rank2		62.5000% 66.6700%	72.3300% 75.5200%	2024.10.16 17:57

파인튜닝한 모델의 성능을 평가할 수 있는 기준으로는 Public 리더보드와 Eval 값이 있습니다. 제가 Hugging Face에 업로드한 모델(ssunbear/klue_roberta_large_finetuned_korquad_v1)은 Public 리더보드에서 낮은 EM(Exact Match) 점수를 기록하여 앙상블 모델에는 사용되지 않았습니다.

하지만 Private 리더보드가 공개된 후, 제가 파인튜닝한 모델이 단일 모델 SOTA를 달성했다는 사실을 확인하게 되었습니다. 이는 제가 진행한 파인튜닝 작업과 모델 구조의 개선이 실제로 효과적이었다는 것을 입증하는 결과였습니다.

이 모델이 Private에서 좋은 성능을 보였지만, 아쉽게도 해당 모델을 더욱 발전시키지 못한 것이 아쉬웠습니다. Public과 Private 성능의 차이를 사전에 알 수 없었기 때문에, 다음 프로젝트에서도 이와 같은 문제를 극복하기는 어려울 것이라고 생각합니다.

다만, Public과 Private 리더보드 간의 성능 차이는 확실히 존재할 수 있다는 점을 다시 한 번 깨달았습니다. 이를 통해 향후 프로젝트에서는 더 철저한 검증 절차를 마련하고, 모델의 성능을 평가할 때 Public뿐만 아니라 Private에서도 좋은 결과를 낼 수 있도록 준비하는 것이 중요하다는 교훈을 얻었습니다.

앞으로는 Public 리더보드에서의 성과뿐만 아니라, Private 리더보드에서도 좋은 성과를 낼 수 있도록 다양한 평가 방법을 도입하고, 여러 상황을 고려한 모델 개발을 할 계획입니다.

한계/ 교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

이번 프로젝트에서는 팀 템플릿 코드가 다양하게 분산되어 있어 협업이 다소 비효율적이었습니다. 따라서, 다음 프로젝트에서는 리더가 제공한 하나의 일원화된 템플릿 코드를 사용하여 팀원들이 보다 일관되게 작업할 수 있도록 하고자 합니다. 이를 통해 코드의 가독성과 유지 보수성을 높이며, 팀원들이 각자의 작업에 더 집중할 수 있는 환경을 조성할 수 있을 것입니다. 또한, 팀원들이 리더의 템플릿 코드를 잘 활용하여 시간을 절약할 수 있는 프로젝트 진행이 필요하다고 생각합니다. 통일된 템플릿을 사용하면 중복 작업을 줄이고, 코드 리뷰와 디버깅 과정에서도 효율성을 높일 수 있을 것입니다.

지난 프로젝트보다 Git에 대한 이해도가 많이 향상되었습니다. 하지만, 다른 팀원들이 어떤 커밋을 했는지 명확하게 파악하기 위해서는 매번 커밋 내역을 일일이 확인해야 했습니다. 이는 시간 소모가 크고, 협업 과정에서 불필요한 혼란을 초래할 수 있었습니다. 따라서, 다음 프로젝트에서는 Slack과 Git을 연동하여 특정 커밋이 이루어졌을 때 자동으로 알림을 받을 수 있는 시스템을 구축하고 싶습니다. 이를 통해 팀원 간의 소통을 원활하게 하고, 각자의 작업에 대한 이해도를 높일 수 있을 것입니다.

이와 함께, Git의 브랜치 관리와 PR(Pull Request) 과정에서도 더욱 체계적인 접근을 시도하고자 합니다. 각 팀원이 작업하는 브랜치에 대한 규칙을 명확히 하고, PR 리뷰를 통해 코드 품질을 높이는 방안을 마련할 것입니다. 결론적으로, 다음 프로젝트에서는 템플릿 코드의 통일성과 협업 도구의 효과적인 활용을 통해 팀의 생산성을 높이고, 보다 원활한 소통과 협력이 이루어질 수 있도록 노력할 것입니다. 이러한 경험은 향후 프로젝트에서도 큰 도움이 될 것이며, 지속적인 개선을 통해 팀의 역량을 강화할 수 있는 기회가 될 것입니다.

4. 이정민

한 것

저는 이번 프로젝트에서 데이터 전처리 증강과 같이 Reader 쪽에서 성능을 향상 시킬 수 있는 방법론을 많이 연구하였습니다. 우선 데이터 전처리 및 증강에서는 Sentence Shuffle, Sentence Truncation, Korquad 변환, Question Text 증강 등 Question과 Context의 텍스트를 수정하는 방향으로 증강하는 기법을 연구하였습니다. 논문 등을 참고하여 기법의 성능과 한계를 공부하였으며, 이 중 Truncation 기법은 실제 원래 모델보다 점수가 향상되어 최종 앙상블 기법에 사용된 모델이 되었습니다.

좋았던 점

가장 좋았던 것은 저번 프로젝트보다 다 깊이 있게 torch, transformers 아키텍쳐를 살펴본 점이었습니다. 이번 프로젝트는 저번 프로젝트와 다르게 데이터셋의 규제가 KLUE를 제외하고는 걸려있지 않았습니다. 즉 다른 커스텀 데이터셋을 가져올 수 있었고, 저는 이것을 이용하여 여러 가지 Huggingface나 기타 Github의 데이터셋을 transformers와 datasets 라이브러리에 연동하는 것에 대하여 배워보자는 마음으로 데이터 전처리 쪽을 많이 담당하였습니다.
그 결과 transformers 라이브러리에서 데이터 셋을 가져오는 것에서부터 training Argument의 설정, 그리고 최종적으로 trainer 클래스 및 실행 함수 수행 까지의 전체적인 프로세스를 잘 읽을 수 있는 정도로 성장한 것 같습니다. 특히 데이터 전처리와 후처리는 datasets 라이브러리의 .map 메소드를 이용하여 특정 disk나 huggingface에서 불러오는 데이터셋들을 Batch로 변환하는 함수인데, 이 함수를 잘 사용하는 법, 인자를 추가로 주는 법 등을 배울 수 있게 되었습니다.

부족했던 점

가장 부족했던 것은 wandb를 처음 쓴 것에 대한 미숙함입니다. wandb는 자동으로 train, eval, test의 정보를 기록하여 마치 텐서보드처럼 metric을 활용할 수 있는 특징이 제일 큰 장점인데, 이것을 잘 활용하지 못한 것 같습니다.
basecode로는 metric이 만들어 질 때의 timestamp를 이용하여 로깅을 시작하고 이것이 wandb API에 넘어가게 되어 자동으로 기록되는데, 문제는 훈련과 검증을 너무 많이 해서 중간에 기록을 보려고 할 때 50개~100개 사이의 기록을 계속 살펴보면서 시간을 낭비한 것 같습니다.
다음에는 basecode가 이렇게 작성되었다면 이것을 훈련 이전에 wandb init을 하는 과정에서 run 이름을 강제로 설정하게 하고, 이것을 eval과 prediction에도 적용하게 하여 실제 기록을 이용할 때 바로바로 볼 수 있게 수정을 할 것 같습니다.
또한 다음 프로젝트부터는 팀원과의 논의를 통해 실제 run을 생성할 때 그 이름 규칙을 어느 정도 정해야 할 필요성을 느꼈습니다. 보통 논의를 할 때 모델과 기법을 중심으로 점수 비교와 작업 내용을 설명하는데, 이것이 실제 글의 내용과 약간 다를 때가 있었기 때문입니다.

5. 임한택

ODQA 프로젝트를 진행하면서 전반적으로 체계적인 방법론을 적용하려 노력했습니다. 이전 STS 프로젝트에서 단순히 모델만 바꿔가며 앙상블했던 것과는 다르게 데이터 분석을 통한 문제 정의부터 시작하여 구체적인 가설을 세우고 검증하는 방식으로 진행했습니다. 특히 VSC Extentions Todo-Tree를 활용하여 실험 계획을 효율적으로 관리했고, 이를 통해 각 과정에서의 성과를 명확히 확인하면서 진행할 수 있었습니다.

협업 측면에서도 많은 성장이 있었습니다. GitHub을 통한 코드 관리와 버전 제어 능력이 향상되었고, Discussion을 활용한 팀원들과의 의사소통도 원활히 진행할 수 있었습니다. Branch 관리, Pull/Push 등의 기본적인 Git 기능을 능숙하게 다룰 수 있게 된 것은 큰 성과였습니다.

하지만 아쉬운 점도 있었습니다. 베이스라인 코드나 추가 기능을 구현한 코드를 GitHub에 올리고 팀원들과 논의하는 역량이 부족하다는 것을 느꼈습니다. 또한 WandB Sweep을 활용한 하이퍼파라미터 최적화를 시도하지 못한 것과, 시간 제약으로 인해 구현하지 못한 여러 아이디어들이 있었던 것이 아쉽습니다. 상대적으로 저번 프로젝트보다 GitHub의 Issue, Discussion, Review 등을 잘 활용했으나, 팀원들과 비교해서 아직 충분히 활용하지 못한다고 느꼈습니다.

앞으로의 프로젝트에서는 이러한 부족한 점들을 보완하고자 합니다. 전체적인 흐름을 이해하고 코드를 구현하는 역량을 키우고, 다양한 실험적 시도를 위한 시간 관리를 철저히 할 계획입니다. GitHub의 다양한 협업 기능을 학습하여 팀 프로젝트 진행 시 더 효율적인 협업이 가능하도록 하겠습니다. 특히 혼자서도 베이스라인 코드 작성과 feature 브랜치 관리를 할 수 있는 역량을 키워서 팀원들과 협업함에 있어서 불편함이 없도록 할 것입니다. 이번 프로젝트의 경험을 통해 배운 점들을 잘 활용하여 앞으로의 프로젝트에서는 더 나은 결과를 만들어낼 수 있을 것으로 기대합니다.