

문장 내 개체간 관계 추출

- NLP-05 Wrap-up Report -



브레멘 음악대

변성훈 서보성 이도현

이상민 이승우 이예원

<목 차>

I. 서론

- 1. 프로젝트 개요 ----- 3
- 2. 프로젝트 팀 구성 및 역할 ----- 3

II. 본론

- 1. 프로젝트 수행 절차 및 방법
 - 1-1 공통 환경 설정 ----- 3
 - 1-2 프로젝트 수행 및 완료 과정 ----- 4
- 2. 프로젝트 수행 결과
 - 2-1 Hyperparameter and Analyzing Evaluation Metrics ----- 4-5
 - 2-2 Data Analysis ----- 5-7
 - 2-3 Modeling ----- 7-9
 - 2-4 Optimization ----- 9
 - 2-5 Ensemble ----- 9-10

III. 결론

- 1. 자체 평가 의견 ----- 10

IV. 개인회고

- 참고문헌 ----- 20

I. 서론

1. 프로젝트 개요

관계 추출(Relation Extraction)은 문장이나 텍스트에서 두 개체(대상) 사이의 관계를 식별하고 분류하는 작업이다. RE 작업은 정보 검색 및 추출, 지식 그래프 구축 등 다양한 응용 분야에서 중요하게 활용된다. 프로젝트의 목표는 문장과 문장의 두 개체가 주어졌을 때, 이 두 개체 사이의 관계를 자동으로 추출하도록 하는 것이다.

2. 프로젝트 팀 구성 및 역할

변성훈: 데이터 시각화, 데이터 분석 전반, 데이터 증강, StratifiedKfold 구현

이도현: wandb/sweep 세팅, domain adaptation, ensemble 구현

서보성: sequential 모델, K-Epic 구현 및 실험, koelectra 성능 실험

이예원: entity 전반적, restriction + penalty, binary classification 구현 및 실험

이상민: random seed 설정, special entity 관련 구현 및 실험

이승우: git 세팅 및 브랜치 관리, loss 관련 구현 및 실험, 기타 리서치 및 아이디어이션

II. 본문

1. 프로젝트 수행 절차 및 방법

1-1 공통 환경 설정

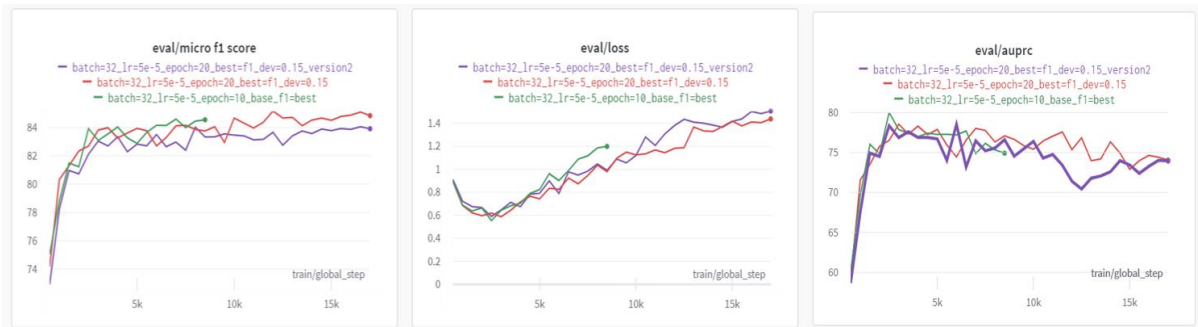
기능을 많이 구현하게 되면 argument 로 인자를 받기에 불편하게 될 것 같아, yaml 파일로 config를 관리하고, wandb와 sweep을 사용하여 하이퍼 파라미터 조정 및 성능 확인을 하였다. 실험의 성능을 확인하기 위해 seed 고정을 하고 진행하였으며, 기본 베이스라인 코드를 github에 올릴 수 없기에 보편적으로 사용되는 방법인 entity를 punctuation으로 구분하는 방법과 entity type을 special token으로 구분하는 방법을 미리 구현을 해 놓고, git 세팅을 하였다.

1-2 프로젝트 수행 및 완료 과정

		100%		2023-05-05	2023/5/19		
Data				2023-05-05	2023/5/18	13	
데이터 분포 및 특징 분석	변성훈	100%		2023-05-05	2023/5/18	14	
entity type 교차 증강	서보성	100%		2023-05-06	2023/5/7	2	
중복 데이터 제거	변성훈	100%		2023-05-07	2023/5/8	2	
no relation undersampling	이도현	100%		2023-05-08	2023/5/9	2	
per:place_of_residence masking 증강	변성훈	100%		2023-05-17	2023/5/17	1	
Baseline				2023-05-05	2023/5/19	14	
seed 고정	이상민	100%		2023-05-05	2023/5/7	3	
yaml 파일 및 기초 모듈화	서보성	100%		2023-05-07	2023/5/8	2	
git 세팅 및 branch 관리	이승우	100%		2023-05-05	2023/5/19	15	
Model				2023-05-05	2023/5/16	11	
sota 논문 및 아이디어 서치	서보성, 이승우	100%		2023-05-05	2023/5/7	3	
entity punct, entity special 구현 및 실험	이예원	100%		2023-05-05	2023/5/7	3	
continual pretraining (MLM)	이도현	100%		2023-05-08	2023/5/14	7	
다양한 모델 기본 성능 테스트	모두	100%		2023-05-08	2023/5/9	2	
entity special 변형 및 다양한 실험	이상민	100%		2023-05-09	2023/5/16	8	
binary model	이예원	100%		2023-05-09	2023/5/10	2	
DRP 구현	이예원	100%		2023-05-11	2023/5/12	2	
sequential model 구현 및 실험	서보성	100%		2023-05-11	2023/5/16	6	
K-epic 논문 방법 실험 (koelectra, roberta-large)	서보성, 이예원	100%		2023-05-13	2023/5/16	4	
entity type restriction	이예원	100%		2023-05-14	2023/5/16	3	
some layer freeze + LSTM	이승우	100%		2023-05-15	2023/5/16	2	
add source to special token	서보성	100%		2023-05-17	2023/5/17	1	
Ensemble				2023-05-14	2023/5/16	2	
StratifiedKFold	변성훈	100%		2023-05-14	2023/5/15	2	
Softvoting ensemble	이도현	100%		2023-05-15	2023/5/16	2	
Visualization				2023-05-05	2023/5/10	5	
wandb/sweep 세팅	이도현	100%		2023-05-05	2023/5/7	3	
confusion matrix, precision-recall, ROC	변성훈	100%		2023-05-09	2023/5/10	2	
Optimization				2023-05-08	2023/5/12	4	
다양한 loss 구현 및 실험	이승우	100%		2023-05-08	2023/5/12	5	
etc				2023-05-18	2023/5/19	1	
발표 준비	변성훈, 이예원	100%		2023-05-18	2023/5/18	1	
발표	변성훈	100%		2023-05-19	2023/5/19	1	
코드 정리 및 마지막 모듈화	서보성, 이승우	100%		2023-05-19	2023/5/19	1	

2. 프로젝트 수행 결과

2-1. Hyperparameter and Analyzing Evaluation Metrics



위와 같이 2500 step 정도를 기준으로 loss와 auprc가 낮아지지만, f1 score는 완만하게 증가하는 경향을 다음과 같이 해석했다. F1 score에서는 no relation이 기준이 되는 TP, FP, FN은 계산에 포함하지 않고 auprc와 loss는 계산에 포함시킨다. 그렇기에 no relation의 차이를 원인으로 가정하고, 모델이 만약 no relation 쪽으로 편향된 예측을 한다고 생각해보았다. 원래 no relation이 아닌 라벨로 잘못 예측하던 데이터들이 편향되어 no relation으로 예측한다면, 기존의 FP가 사라지게 되어 f1

score가 증가한다. 그러나 그 경향이 test data에서는 맞는 라벨을 no relation으로 편향되게 하는 정도가 더 크기에 오히려 제출 score는 하락시킨다 판단하였다. 그렇기에 loss가 증가하는 포인트인 2500 step 즉 3 epoch을 최적의 epoch으로 고정하였다. 이외의 hyperparameter는 dev dataset을 train dataset에서 분포를 맞추어 0.15 만큼 떼어낸 것을 바탕으로 sweep을 해보니 warmup=400, lr=2e-5, decay=0.1 이 최적이라 판단했다.

평가 매트릭 auprc는 라벨에 대한 성능을 평균내는 것이고 f1 score는 주어진 데이터에 대한 성능을 평가하는 것이기 때문에, 만약 private에서 라벨의 분포가 다를 수 있다는 점을 고려해 auprc를 정규화 성능이라 보았다.

2-2. Data Analysis

(1) 중복데이터, 잘못된 라벨 제거

Train 데이터에서 sentence가 완전히 동일한 데이터를 확인 해 보았을 때, 우측과 같은 결과가 나왔다. sentence와 label이 동일한 경우가 1684 개, sentence와 subject entity이 동일한 경우가 1263개 있다는 의미로 중복된 column 조합마다의 개수를 나타낸 것이다. subject entity와 object entity 가 하나라도 다르다면 그 데이터는 다른 데이터로 보고 subject entity와 object entity가 같은 경우만을 확인하였

```
label : 1684
sub : 1263
obj : 877
sub, obj : 47
sub, label : 392
obj, label : 645
sub, obj, label : 42
```

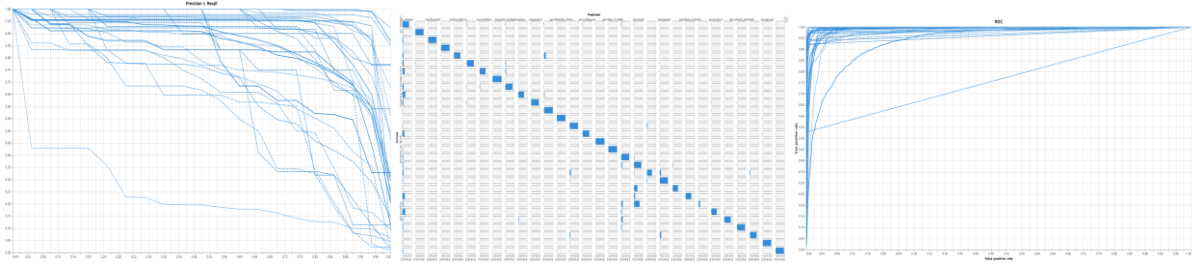
다. subject entity, object entity 가 같은 47개의 데이터 중에서 42개는 label까지 동일한 완전히 중복되는 데이터이므로 제거해주었다. label이 다른 5개의 데이터는 눈으로 확인해서 잘못된 라벨인 데이터 5개를 제거하고 추가적으로 subject type이 per이지만 label이 org: top_members/employees로 되어있는 데이터 4개를 제거해주었다. 그러나 klue/roberta-large로 train dataset을 제외한 모든 변인을 일치시키고 제출해본 결과 micro f1 score: 72.8183 → 72.5625 auprc: 79.0017 → 78.9392로 약간의 성능하락이 있기에 사용하지 않으려 했으나 실수로 데이터 증강을 중복 제거한 데이터셋에 대해 진행하였다.

(2) 데이터 시각화

Wandb를 이용해 dev set에 대한 ROC graph, precision-recall graph, confusion matrix를 확인할 수 있도록 아래 이미지와 같이 구현하였다. 이 그래프들을 통해 해당 모델이 어떤 label을 학습하지 못하고 있는지를 다음과 같이 파악할 수 있었다.

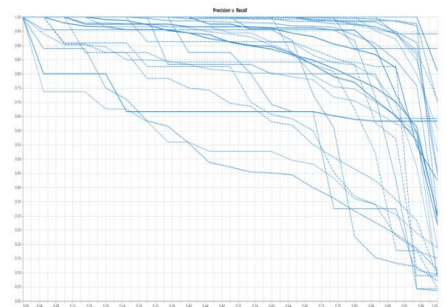
per: place of residence와 per: origin이 매우 유사한 의미를 가지는 라벨이고 데이터를 확인해보니 사람도 제대로 분류하기 어려울 정도로 유사하다. 거기다 per: place of residence는 193개, per:

origin은 1234개로 데이터 개수의 차이가 있기에 per: place of residence가 per: origin으로 잘못 예측하고 있었다.



(3) 데이터 증강

위와 같은 문제를 해결하기 위해 per: place of residence 라벨의 데이터를 증강해 per: origin과 조금이라도 맞추어 주고자 했다. sentence에서 entity를 제외한 하나의 어절을 랜덤하게 선택하여 [MASK]를 씌워주는 방식으로 2배 증강하여 다시 실험을 해본 결과 micro f1 score: 76.1570 → 76.4541, auprc: 80.3244 → 81.9893으로 특히 auprc의 성능향상을 확인할 수 있었고 실제로 precision-recall graph에서도 해당 라벨의 성능이 상당히 향상했음을 알 수 있었다.



(4) 데이터 전처리

모델이 데이터를 학습할 때, 라벨을 예측하기 위해 중요한 정보인 entity word와 type 정보를 강조하기 위해 아래와 같이 데이터를 전처리 하였다.

〈Something〉는 @ § PER § '조지 해리슨' @이 쓰고 # ^ ORG ^ '비틀즈' #가 1969년 앨범 《Abbey Road》에 담은 노래다. : 이 문장에서 '조지 해리슨' 는 '비틀즈' 의 'PER' 이다. - (1)

〈Something〉는 @ § 인물 § '조지 해리슨' @이 쓰고 # ^ 단체 ^ '비틀즈' #가 1969년 앨범 《Abbey Road》에 담은 노래다. : 이 문장에서 '조지 해리슨' 는 '비틀즈' 의 '인물' 이다. - (2)

subject, object entity word는 @, # 로 type은 §, ^ 로 구분해주었다. §와 같은 경우 [UNK]로 구분되는데도 불구하고 사용한 이유는 다른 특수문자는 모두 이미 train, test에서 사용되어 사용되지 않는 특수문자를 사용할 필요가 있다고 생각했고, *과 §를 각각 사용해 성능을 비교해본 결과 미세하게 §를 사용하는 쪽이 더 높았다. 또한 기존 sentence에서는 entity word에만 '로 강조하고 추가된 description에서는 word와 type모두 '로 강조하는 것이 evaluation에서 성능이 더 높았다.

klue/roberta-large에서는 classification을 위한 hidden state로 맨 앞의 @, # 두 개만 이용하는

것이 가장 성능이 좋았고, monologg/koelectra-base-v3-discriminator 에서는 [CLS]와 앞 뒤 @, # 4 개 모두를 사용하는 것이 가장 성능이 좋았으며 가장 성능이 낮은 조합과 1점가량 차이가 났다. klue/roberta-large로 (2)와 같이 전처리 후, 두개의 punctuation만으로 classification을 했을 때, f1 score: 72.2705 → 73.9184, auprc: 76.5282 → 79.9759 정도의 성능 향상이 있었다.

entity word를 강조하기 위해 special token을 사용한 방법 또한 다양하게 사용해 보았다.

〈Something〉는 [S:PER] '조지 해리슨' [/S:PER]이 쓰고 [O:ORG] '비틀즈' [/O:ORG]가 1969년 앨범 《Abbey Road》에 담은 노래다. - (3)

〈Something〉는 [SUBJ] 사람 '조지 해리슨'이 쓰고 [OBJ] 단체 '비틀즈'가 1969년 앨범 《Abbey Road》에 담은 노래다. : 이 문장에서 '조지 해리슨' 는 '비틀즈' 의 '인물'이다. - (4)

(3) f1 score: 72.8183 → 71.3360, auprc: 79.0017 → 77.2479 (punctuation 비교)

(4) f1 score: 73.1196 → 71.7836, auprc: 79.1763 → 77.2422 (punctuation 비교)

(4)의 경우 [CLS]와 [SUBJ], [OBJ]를 이용해 classification을 하는 것이 가장 성능이 좋았고 그럼에도 불구하고 새로운 special token을 추가해 새로 학습을 시키는 방법이기 (3), (4) 모두 punctuation 보다 심지어 base보다도 낮은 성능을 보였다.

〈Something〉는 [SUBJ] 사람 '조지 해리슨'이 쓰고 [OBJ] 단체 '비틀즈'가 1969년 앨범 《Abbey Road》에 담은 노래다. : 이 문장에서 조지 해리슨은 '인물'이고 비틀즈는 '단체'이다. - (5)

대회 마지막에 발견하여 punctuation에 적용은 못하였지만, (4)에서 (5) 방식으로 뒤의 description으로 바꾸었을 때 f1 score: 71.7836 → 72.4420, auprc: 77.2422 → 79.3848의 성능 향상을 보여 (5) 와 같은 방식처럼 word와 type에 대한 정보를 최대한 많이 주는 방식이 가장 좋은 것 같다고 추측된다.

2-3. Modeling

(1) Domain Adaptation

klue/roberta-large 모델로 train dataset에 대하여 Masked Language Modeling으로 추가적으로 pretraining을 진행하고 pretrained 된 모델로 RE task에 대해 finetuning을 진행하였다. MLM pretraining eval/loss: 1.661 (0 step) → 1.426 (10,000 step)처럼 klue/roberta-large가 사전 학습 시 사용한 데이터셋과 RE task의 데이터셋이 유사하여 loss가 크게 떨어지지 않은 것으로 추측되며, 정확한 원인은 모르지만 그래서 성능 향상이 없었을 수도 있다. 또한 시간 상 진행하지 못하였지만 RE task에서 중요한 entity word 만을 masking 해서 예측하도록 하거나 NER task로 pretraining

을 진행했다면 더 좋았을 것 같다.

(2) Binary Classification → Multi Classification

no relation이 아닌 라벨이 no relation으로 향하는 것을 방지하고 싶어 우선 relation과 no relation으로 분류하는 학습을 따로 진행하고자 하였다. binary classification을 진행하는 모델과 multi classification을 진행하는 모델이 독립적으로 그리고 연속적으로 학습하는 두가지 방법으로 구현해 실험해 보았다.

[독립적 학습]

모델1. train.csv로 no relation 과 relation 구분 학습

모델2. train.csv에서 no relation 라벨 제거한 데이터셋으로 29개 라벨 분류 학습

Inference: 모델1으로 no relation 먼저 구분 후, relation에 대해 모델2로 29개의 라벨로 분류

→ klue/roberta-small 기준 micro f1 score: 60.7637 → 56.0278

[연속적 학습]

모델1. train.csv로 no relation과 relation 구분

모델2. inference에서는 모델 1의 결과를 train에서는 실제 label의 no relation, relation에 대한

description (sub word랑 obj word는 관계가 없다.) 으로 model2의 input에 추가

→ 30개의 라벨로 분류

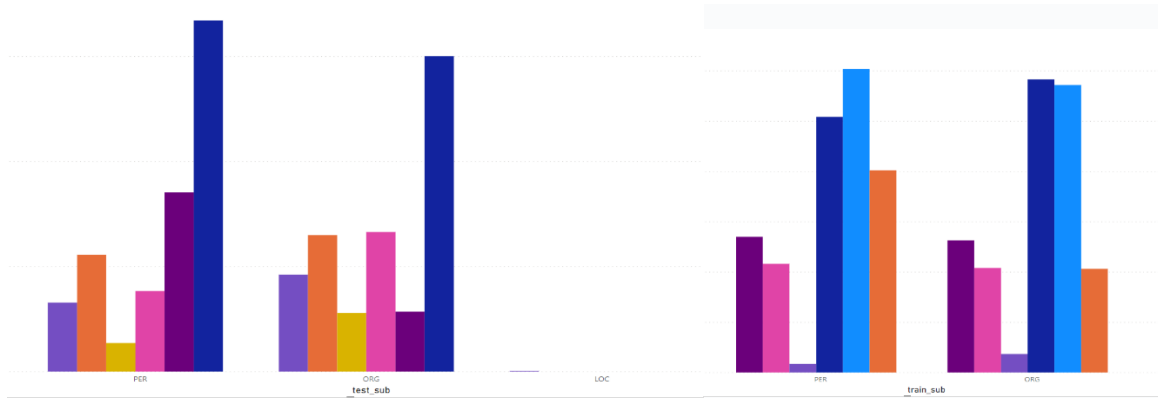
(두개의 loss를 더한 값을 최소화하는 방향으로 backpropagation이 진행된다.)

→ klue/roberta-large 기준 micro f1 score: 72.2705 → 70.4588, auprc: 76.5282 → 72.4160

학습 당시에 첫번째 모델의 성능이 두번째 모델에 영향을 미치지 않도록 학습시킨다 하여도 inference 과정에서는 영향을 미칠 수밖에 없다. 그렇기에 첫번째 모델의 성능이 중요한데 생각보다 30개의 라벨이 아닌 2개의 라벨로 분류하는 작업조차 성능이 좋지 않았기에 성능 하락이 있을 수밖에 없었다. 그리고 연속적으로 학습한다면 이진 분류와 다중 분류의 loss가 동시에 최소화시키는 방향으로 학습한다는 점 또한 성능 하락의 원인으로 보인다.

(3) Entity Type Restriction

subject와 object의 entity type의 조합으로 가능한 라벨만을 예측하도록 restriction을 준다면 더 예측을 잘할 것이라 기대하였다. 그러나 아래 그래프와 같이 entity type의 조합에 따른 데이터가 test와 train이 많이 다르기에 train 데이터를 토대로 엄격하게 제한을 주는 것은 위험하다 판단되어 약간의 제한만을 두었다.



우선 entity type의 조합마다 다른 classifier를 사용하는 방법은 train data에서 type마다의 데이터가 불균형하고 충분하지 않기에 과적합이 일어난다고 추측된다. 그래서 하나의 classifier로 해당 조합에서 가능한 라벨들의 loss만을 계산하거나 weight를 주어 해당 라벨을 중점으로 계산하는 방법을 시도해보았으나 f1 score: 73.1196 → 72.9463, auprc: 79.1763 → 78.0597 정도의 성능 하락을 보였다.

2-4. Optimization

데이터 불균형 문제 해결을 위해 Focal loss($FL(p_{t_i}) = -\alpha * (1 - p_t)^{\gamma} * \log(p_t)$)를 사용했다. 튜닝 결과 $\alpha=0.8$, $\gamma=3.0$ 일 때 86.117 → 86.181로 최적의 값을 내었으나, 일반적인 성능 향상을 이뤄내지는 못했다.

Label Smoothing으로는 0.1의 비율을 사용하였으며, 약간의 성능 향상이 있기도 했으나 가장 성능이 잘 나오는 punctuation 스타일의 전처리에서는 성능 향상을 보이지 않아 사용하지 못하였다. 또한 양상블을 고려했을 때, 성능 향상이 미미하다면 맞는 라벨을 확실하게 맞추는 경우가 더 좋을 것 같아 일반적인 Cross Entropy를 사용하였다.

Data Analysis의 confusion matrix에서 확인한 문제는 크게 두 가지였는데, per: place of residence를 per: origin으로 예측한다는 문제점과 relation인 label을 no relation으로 예측한다는 것이다. 이를 보완하기 위해 per: place of residence의 라벨에 weight를 주었더니 per: place of residence가 약간 잡히기도 했으나 미미했고 다른 라벨의 성능에 약간의 영향이 있어 성능 향상이 없었다. 또한 no relation이 아닌 라벨을 no relation으로 예측할 경우의 loss를 2배로 늘리는 식으로 penalty를 주기도 하였으나 이 또한 효과가 없었다.













2-5. Ensemble

양상블 방법으로는 StratifiedKFold와 Seed Ensemble을 사용하였다. fold는 5를 사용하였고 StratifiedKFold의 경우 f1 score: 73.7912 → 76.1570, auprc: 80.2272 → 80.3244 정도의 성능 향상

을 두개의 Seed로 StratifiedKFold 앙상블을 한 경우 f1 score: 73.7912 → 75.8320, auprc: 80.2272 → 81.2161 정도의 성능 향상을 보였다. 둘을 모두 사용하면 더욱 성능 향상이 일어날 것이라 생각했으나 StratifiedKFold 단독으로 사용하는 것이 가장 성능이 좋았다.

- 데이터 전처리의 (2) 방식 + augmentation + seed = 77 + StratifiedKFold
- 데이터 전처리의 (1) 방식 + seed = 12 + StratifiedKFold
- 데이터 전처리의 (2) 방식 + augmentation + seed = 23, 124 + StratifiedKFold

최종적으로 사용한 모델은 위 3개로 첫번째 모델과 두번째 모델을 제출 스코어를 바탕으로 soft voting을 하였고 이 두 모델을 제출한 점수와 세번째 모델을 제출한 점수를 2:1 비율로 가중치를 주어 soft voting을 하였다.

1 (-)	NLP_05조	     	77.0249	81.2780	101	3d
2 (1▼)	NLP_05조	     	75.6050	83.7164	101	3d

Ⅲ. 결론

1. 자체 평가 의견

우선 최종적인 스코어는 높았으나, 그 과정 속에서 아쉬웠던 점은 분명히 있었던 것 같다. 이번 대회에 경우 적용할 수 있는 가설이 다양했기에 실제로 우리 팀도 다양한 가설을 구현해 보았다. 대부분의 가설이 실제 성능 향상으로 이어지지는 않았는데, 가설을 세울 때 바로 구현을 해보기 보다는 그 가설이 나온 이유와 해결하고자 하는 문제점을 제대로 파악하고 시도를 해봐야 하는 것 같다. 그 점이 부족했다고 느꼈으며, 또한 성능 향상이 되지 않을 경우 바로 다음 가설로 넘어가는 것이 아닌 원인을 제대로 분석하고자 하는 의지가 부족했다고 생각한다.

그리고 성능을 올리는 요인이 복잡한 모델링 같은 엄청난 지식을 요구하는 것뿐만 아니라 사소한 데이터 전처리나 모델을 약간 수정하는 것에서 나올 수 있기에 다양한 실험을 해보는 것이 좋다는 것을 알았다.

협업 관련해서 아쉬기도 하고 다음 대회부터 도전할 과제이기도 한 부분이 git을 제대로 활용하지 못했다는 점이다. 대회의 스코어에 집중하는 것도 좋지만 협업을 경험하기 위해서 다

음 대회부터는 번거롭더라도 git의 기능들을 활용해보기로 하였다.

IV. 개인 회고

1. 서보성

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

지난 프로젝트에서 모델의 정확한 성능을 수치로 측정하지 못했다는 점이 가장 아쉬웠다. 그래서 이번 프로젝트의 목표를 정확한 성능을 수치로 측정하는 것으로 잡았으며, 또한 모델의 성능을 높이기 위해서 가설을 세울 때 좋은 방향성을 가지고 가설을 세울 수 있도록 문제를 인식하는 것도 목표로 잡았다.

이러한 목표들을 달성하기 위해서 다양한 조건에 대해서 모델의 성능을 측정하였으며 조건별로 그 수치를 정리했다. 특히 시드 고정과 같이 항상 같은 결과가 나오도록 하여 정확한 비교가 가능하도록 했다. 그리고 문제를 정확하게 인식하기 위해서 팀원들과의 많은 소통을 하였으며 모델의 예측 결과에 대한 분석을 했다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

이전에도 언급했듯이 지난 프로젝트에서는 모델의 정확한 성능을 수치로 측정하지 못했다. 그래서 이번 프로젝트에서는 다양한 시도들을 기록하고 다양한 실험의 결과들을 공유하고 의견을 나누는 작업을 했다. 또한 모두 같은 방식으로 기능에 따른 모델의 성능만을 비교하기 위해서 다른 변수들을 제거한 베이스라인 코드를 구성하여 여러 가설들의 실험을 진행하는 변화를 시도했다. 정확한 성능을 수치로 측정한 것과 다른 변수들을 제거한 베이스라인 코드를 만든 것은 해당 가설이 성능이 향상이 되는지 아닌지를 바로 확인할 수 있는 효과를 얻을 수 있었으며, 이를 기반으로 다른 가설들을 추가하여 더 좋은 모델의 성능을 낼 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

많은 가설을 세우고 실험을 진행하는 것은 좋았지만 해당 부분에만 집중해서 지난 프로젝트에서 잘 진행했었던 실험의 내용을 정리하는 작업을 진행하지는 못했다. 또한 Sequential하게 제작한 BERT 모델에서 발생할 수 있는 문제에 대해서 팀원들과 충분히 의견을 나누었지만 제작이 오래 걸렸던만큼 아쉬움이 남아서 다른 가설들을 할 시간이 부족해졌다는 아쉬움도 있다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

지난 프로젝트에서 부족했던 부분을 이번 프로젝트에서 개선했지만, 지난 프로젝트에서 잘했던

부분이 이번 프로젝트에서는 부족했다. 많은 부분을 개선하는 것도 중요하지만 현재 잘했던 것들을 계속 잘하는 것이 중요하다고 생각하기 때문에 다음 프로젝트에서는 실험의 내용을 잘 정리하는 작업과 모델의 정확한 성능을 측정하는 것 등을 잘 유지하는 것을 시도해보고 싶다.

나는 어떤 방식으로 모델을 개선했는가?

먼저 'K-EPIC: Entity-Perceived Context Representation in Korean Relation Extraction' 논문을 참고하여 어떤 token을 이용하여 계산하는 것이 성능이 높은지 여러 방법으로 실험을 진행했다. 다음으로 Sequential하게 두 BERT 계열의 모델을 이어서 붙인 새로운 모델을 만드는 방법으로 첫 번째 BERT 모델은 두 entity와 관계가 있는지, 즉 no_relation인지 아닌지를 분류하는 이진 분류 모델이다. 두 번째 모델은 첫 번째 이진 분류를 바탕으로 설명(discription)을 추가하여 30개의 라벨로 다중 분류하는 모델로 구현했다. 마지막으로 'monologg/koelectra-base-v3-discriminator' 모델을 이용하여 앙상블을 할 수 있을 정도로 모델의 성능을 높이는 hyper parameter 튜닝 및 source 토큰 추가 등과 같은 작업을 진행했다.

내가 해본 시도 중 어떠한 실패를 경험했는가? 실패의 과정에서 어떠한 교훈을 얻었는가?

모델을 개선하는데에 있어서 K-Epic을 제외하고는 성능 향상을 얻지는 못했다. 특히 Sequential하게 두 BERT를 이어 붙인 모델의 성능이 나오지 않는 이유는 앞 BERT 모델이 뒷 BERT의 학습을 방해하는 역할을 해서 성능을 떨어트리는 문제가 발생한다는 것을 멘토님과 마스터님의 말씀을 듣고 알게 되었다. 이러한 문제는 가설을 세우는 과정에서 가설을 구현했을 때 발생할 수 있는 문제를 정확하게 인식하지 못한 점에서 발생한 것으로 생각했다. 그래서 이번 시도를 통해서 문제 인식을 통해서 가설을 세우는 것도 중요하지만 해당 가설이 문제가 없는지도 생각해보는 과정을 거친다면 더 좋은 방향성을 가지는 가설을 세울 수 있을 것이라고 생각했다.

협업 과정에서 잘된 점 / 아쉬웠던 점은 어떤 점이 있는가?

지난 프로젝트에 비해서 협업 과정에서 잘된 점은 소통이다. 팀원들과 많은 소통을 하고 부스트캠프의 코어타임뿐 아니라 시간에 관계없이 소통을 해서 의견을 나누고 더 좋은 방향성을 가지고 프로젝트를 진행할 수 있었다는 것이 잘된 점이라고 할 수 있다. 또한 많은 의견 교환을 통해서 세운 가설들을 누가 실험할 것인지를 계획을 세워서 진행하여 프로젝트가 효율적으로 돌아갈 수 있도록 했다는 점이 좋았다. 하지만 Github의 issue나 hugging face의 versioning같은 다양한 협업 툴을 사용해보지 못했다는 점에서 아쉬움이 있다.

2. 이에원

이번 프로젝트에서 나의 목표는 무엇이었는가?

구현에 많이 참여해서 좋은 코드를 작성하고 코드를 한눈에 보고 이해할 수 있는 능력을 기르고자 하였다. 또한 실험을 할 때, 통제 변수를 일정하게 유지하고 조작 변수만을 바꿈으로써 성능을 정확하게 확인하는 것을 목표로 삼았다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

코드를 작성하는 데에 있어서 이미 있는 함수가 있다면 그 함수를 이용해서 코드를 간결하게 유지하려고 노력하였다. 변수를 통제할 때, 초기에 고정해 놓았던 hyperparameter를 엄청난 차이가 있지는 않은 이상 (epoch은 엄청난 차이가 있어서 중간에 변경) 끝까지 유지하며 다양한 실험을 진행해서 성능 차이를 확인해 볼 수 있었다. 그러나 최적의 hyperparameter가 바뀔 경우에 빠르게 고정값들을 바꾸고 실험을 하는 것이 필요하다고 생각한다.

나는 어떤 방식으로 모델을 개선했는가? 그리고 어떠한 깨달음을 얻었는가?

Entity word와 type에 대한 정보를 special token, punctuation 그리고 description으로 넣는 방법을 구현했다. 이 과정에서 예상했던 것보다 성능 향상이 나오지 않았기에 이 원인을 파악하고 최적의 조건을 찾기 위해 다양한 조건들을 바꿔가며 최종 형태를 만들었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

이번 프로젝트에서 정말 많은 실험을 하고 제출을 했는데, 절대적 양이 많다 보니까 그 내용을 체계적으로 정리하는 것이 너무 어려웠다. 나만 알면 되는 것이 아니라 팀원들도 이것을 바탕으로 실험을 한다면 더 다양한 실험을 할 수 있기에, 보기 좋게 정리를 하고 싶었지만 그렇게 하지 못하였다.

또한 나 개인의 hyperparameter는 통일하여 실험하였지만, 중간에 여러모로 바뀌는 일이 많으면서, 팀원들끼리 hyperparameter를 다르게 사용하고 있어 각자의 제출을 보고는 비교를 하지 못하였다.

또한 의사소통을 열심히 하고 많이 한 것은 같으나 그럼에도 내 일에 바빠 성능을 듣고 좋지 않으면 코드도 확인해 보지 않았다. 팀으로서 서로 확인하고 보완해 나가야지 놓친 부분을 찾을 수 있는데 그 점이 부족해서 성능이 오를 수 있었음에도 놓친 부분이 많았을 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

우선 실험이나 제출을 할 때마다 미루지 말고 기록을 해놓는 것이 좋을 것 같다. 또한, 논문 구현 같은 어려운 것 그리고 너무 많은 것에 욕심부리지 말고 최소한의 것을 완벽하게 해내는 것을 목표로 삼고 싶다. 그치만 우선 다음 대회는 점수보다는 git을 더 활용해보며 협업 방식에 더 익숙해지는 것이 1차 목표이다.

3. 이승우

두 번째로 경험해 보는 대회 프로젝트였다. 첫 대회를 통해 문제에 논리적으로 접근해, 문제를 파악하고 그에 알맞은 해결 방법을 제시하는 것이 중요하다는 것은 알고 있었다. 나의 문제는 상대적으로 부족한 경험과, 그렇기에 문제에 맞는 해결 방법을 찾는 능력이 좋지 않다는 것이었고, 따라서 다양한 경험, 다양한 리서치를 해보는 것을 목표로 잡았다.

RE라는 task 자체를 처음 직면하다 보니 먼저 task 자체에 대한 이해를 높이하고자 일반적으로 적용하면 좋은 것들 위주로 정보를 찾아 논문 및 글을 빠르게 찾아 읽었고, special entity와 그와 관련한 다양한 tokenizing 양식들, 다양한 loss 등 각종 방법론을 빠르게 이해하고 팀원들에게 전달할 수 있었다.

하지만 task 자체에 대한 이해도만 높인 채 직면한 문제 상황은 완벽히 이해하지 못한 채로, 문제점을 파악하는 것을 잊은 채 일반적인 방법에만 집중하고 말았다. 대표적으로 focal, label smoothing, LDAM 등 직접 적용을 시도해 본 다양한 loss들이 그러했다. 이론적인 내용을 이해한 뒤 “이건 적용하면 좋을 수밖에 없다”는 생각으로 상황 분석을 소홀히 한 채 실험을 진행한 결과 별다른 성과를 얻지 못한 채 최종 제출에 사용하지 못했다. 주어진 상황에 대한 이해를 철저히 하고 그에 맞게 설정해서 적용했다면 충분히 성능이 올랐을 것이라 생각하기에 큰 아쉬움이 남는 부분이다.

물론 이 경험 덕에 일반적으로 적용하기 좋은 것들에 대한 이해도 및 구현 능력, 구현 속도가 크게 개선되었고 다양한 지식도 얻을 수 있었기에 목표를 어느 정도 달성했지만, 되돌아보니 결국 지난 대회에서의 목표였던 “문제를 파악하고 그에 맞는 해결 방법을 제시하는 것”을 멀리하고 무작정 적용해 본 셈이 되었다. 막바지에 들어서야 confusion matrix를 들여다보다 특정 label이 잘못 분류되는 문제 상황을 캐치하고, 팀원들의 발 빠른 해결책 구현으로 결과에 유의미하게 적용되고 나서야 그것을 상기할 수 있었다.

추가로 git flow를 통한 협업 경험이 많고 자신이 있었기에 관리를 전담했다. 팀원들에게 각종 convention과 branch 전략을 알려준 후 따르도록 하고 모든 conflict를 처리했다. 물론 팀원 모두의 코드를 완벽하게 관리하기는 어려웠고, 결국 대회 후반 및 종료 이후 코드 관리와 리팩토링에 꽤나 많은 시간을 쏟았지만, 결과적으로는 역할을 잘 수행해 낸 것 같다.

내가 할 수 있는 일 중에선 git 관리는 늘 별것 아닌 능력이라고 생각해 왔는데 처음부터 끝까지 관리해 보며 굉장히 중요한 일이라는 사실을 깨달았다. 관리하는 작업도 재밌었고, 든든하다는 말에 자신감도 얻었다. 다음엔 더 많은 방법론을 배우고 익힌 후 팀원에게 알려 같이 체계적인 관리를 시도해 보고자 한다.

프로젝트 자체에는 꽤 익숙해졌다. 다음 프로젝트부터는 현재 주어진 상황(특히 데이터)에 대해 완벽하게 이해하는 과정을 반드시 거칠 생각이다. 이를 기반으로 프로젝트 진행 과정 전체를 꿰뚫으며 철저하게 원인을 분석하고 그에 대한 개선 방법을 찾아 나가는 "스토리 라인"을 만드는 방향으로 움직이는 것을 목표로 할 것이다.

4. 이도현

[나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?]

부스트캠프에서 배운 내용과, 오피스아워에서 조교님께서 말씀하신 부분을 프로젝트에서 활용해보고자 하였다. 또한, 코어타임때는 핸드폰을 침대에 던져두고 학습을 하였다. 미션은 수행하지 못하였지만 실습은 최대한 진행하고자 하였으며, 코드를 짤 때 구글링을 통해 나오는 블로그를 참고하기보다 정식 Document를 더 활용하고자 노력했다.

[나는 어떤 방식으로 모델을 개선했는가?]

효율적인 실험 관리와 시각화를 위해 W&B를 세팅하였다. 데이터 불균형 문제를 Undersampling으로 해결하고자 하였으며, MLM을 우리 데이터에 적용함으로써 Domain adaptation을 적용해보았다. 마지막으로, 앙상블을 통해 모델 성능을 향상하였다.

[내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?]

Domain adaptation 이후 RE fine tuning (eval/f1: 81.564 -> 82.283 at 2000 step)

- Roberta-large에 MLM으로 Domain adaptation을 적용한 이후 Relation Extraction에 Fine tuning을 하였을때, 기존 Roberta-large에 Domation adaptation을 적용하지 않고 바로 Fine tuning을 하였을때보다 동일 step에서 약 0.7점의 eval/f1_score 상승이 있었다. 하지만 성능 향상이 크게 없었으며, 리더보드에 제출한 best 모델과 동일한 메소드를 적용시킨다고 하더라도 성능이 오를 것이라는 확신이 없어 실제 제출하지는 않았다. 성능 향상이 크게 없었던 이유는 Domain adaptation 시에 모델의 loss가 크게 떨어지지 않았기 때문인 것 같은데, klue/roberta-large가 사전학습 시 사용한 데이터와 본 Task의 데이터가 유사하여 그런 것으로 추측한다.

- soft ensemble을 통해 모델의 성능이 크게 향상되었다. 너무 간단한 방법론으로 모델의 성능이 오르니 조금 허무하였지만, 마스터님께서 말씀하신 것처럼 현업에선 앙상블을 많이 사용하지 않으므로, 앙상블을 제외하고 모델의 성능을 극대화할 수 있는 역량을 키워야겠다.

[전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?]

Fine tuning하기 전 Domain adaptation하는 방법론이 실용적으로 많이 사용되는 테크닉이라고 하여서 시도해보았으나, 큰 성능 향상이 없어서 아쉬웠다.

[마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?]

아직도 코드를 구현하는 능력이 많이 부족한 것 같다. 디버깅을 자주 하면서 코드의 흐름을 분석하는 것이 굉장히 중요할 것이라고 생각한다. 코드 구현 능력을 향상하기 위해서 부단히 노력해야 할 것 같다!

[한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?]

코드 구현에 대한 연습을 꾸준히 해서, 내가 생각하고 있는 아이디어를 코드로 큰 어려움없이 구현할 수 있는 사람이 되고싶다. 따라서, 디버깅을 자주 시도하고 Document를 많이 활용하면서 생산성 있는 사람이 되고싶다!!

5. 이상민

1. 대회에 들어가면서 목표

모델의 구현을 많이 해보기

2. 내가 한 것

2-1. 모델 구현

Custom Tokenizer 이용: [SUBJ], [OBJ], [PER], [ORG], [LOC] 등을 special token으로 이용

2-1-1. [CLS] 토큰을 이용해서 모델 만들기

예) [CLS] [SUBJ] [PER] 이순신은 [OBJ] [DAT] 조선시대의 무인이다

예시에서 [CLS] [SUBJ] [PER] [OBJ] [DAT]들의 hidden state를 이용해 classifier layer 구성

2-1-2. [CLS] 토큰 제외

예) [CLS] [SUBJ] [PER] 이순신은 [OBJ] [DAT] 조선시대의 무인이다

예시에서 [SUBJ] [PER] [OBJ] [DAT]들의 hidden state를 이용해 classifier layer 구성

2-1-3. entity type을 special token으로 넣지 않고 단어로 넣기

예) [CLS] [SUBJ] PER 이순신은 [OBJ] DAT 조선시대의 무인이다

예시에서 [CLS] [SUBJ] [OBJ] 등의 hidden state를 이용해 classifier layer 구성

2-1-4. entity type을 한국어 단어로 넣기

예) [CLS] [SUBJ] 사람 이순신은 [OBJ] 날짜 조선시대의 무인이다
예시에서 [CLS] [SUBJ] [OBJ] 들의 hidden state를 이용해 classifier layer 구성

2-2. 문장에 설명 추가

Description = 이 문장에서 subj_word는 “subj_type”이고 obj_word는 “obj_type”이다.

그리고 dicription을 sentence와: 로 이어준다.

예시) [CLS] [SUBJ] 사람 이순신은 [OBJ] 날짜 조선시대의 무인이다: 이 문장에서
이순신은 ‘사람’이고 조선시대는 ‘날짜’이다.

2-3. 내가 한 것들의 평가

Punctuation이 성능이 잘 나왔지만 special token을 이용해서 더 좋은 성능을 내보고 싶은 욕심에 계속해서 여러가지를 시도해보았다. 가장 좋았던 모델은 2-1-4에 설명함수를 추가한 것이었다. 문장이 한국어로 이루어진 문장이며, 이순신의 type을 한국어(의미를) 문장에 넣어준 것이 [CLS], [SUBJ], [OBJ]에 객체 관계를 잘 넣어주지 않았나 생각이 들고, 문장에 설명을 추가하여 객체 간의 type들을 labeling 할 때 한 번 더 주목시킨 것이 좋은 결과를 만들어 냈다고 생각했다.

3. 느낀점

모델을 이론적으로 생각하고 공부하는 것도 중요하지만, 실제로 동작을 하는 방법을 알아야 할 것 같아서 이번 대회에서는 모델의 구현에 집중을 하고 싶었다. Input을 어떤 형태로 받는지, forward에서 어떻게 차원을 맞추어, 어떤 부분을 넘겨줘야 하는지 등 모델의 작동 방식에 맞춰 Custom Model을 만들어 보았다. 그 과정에서 Hugging Face의 Docs를 읽는 법과 디버깅을 하는 방법에도 조금은 친숙해진 것 같아서 개인적 성장을 이룰 수 있었던 대회가 되었던 것 같다.

4. 다음에 해보고 싶은 것

모델에 대한 공부를 했으니, 다음 번엔 데이터에 대해 공부를 해보고 싶다. 데이터를 보는 법, 데이터를 증강하는 다양한 방법 등을 시도해보고 싶다. 모델도 중요하지만 좋은 결과는 좋은 데이터에서 나온다고 생각하기 때문이다. 앞으로도 이렇게 매 대회마다 작은 목표를 두고 이루어 가서 부스트 캠프가 끝날 때는 기본과 기초에 대해서는 잘 알고 있는 캠퍼가 되고 싶다

6. 변성훈

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

저번 STS Task 와 비슷하게 데이터셋에서 **label** 의 분포가 불균형 할 뿐만 아니라 **sub_type** 과 **obj_type** 의 분포도 불균형 하였고, 사람이 눈으로 직접 보아도 label 이 무엇인지 헷갈리는 문장도 매우 많은 데이터였습니다. 그렇기 때문에 데이터의 불균형성에 대해 분석하고, 모델의 성능을 저하시키는 원인을 분명히 밝힌 후 이를 해결하고자 하였습니다.

○ 우리 팀과 나의 학습목표는 무엇이었나?

저번 대회 때 아쉬웠던 점은 많은 가설을 세웠지만, 이것들이 정리가 되지 않아서 대회의 막바지에 방향성을 잃었던 것입니다. 그래서 이번 대회에서는 팀원들과 상의 후 프로젝트를 진행할 환경을 다같이 세팅하고, 정리하여 마지막까지 방향성을 잃지 않고, 공동의 목표를 수행하고자 하였습니다.

○ 개인 학습 측면

Confusion Matrix와 Precision and Recall 그래프를 통해 저희는 모델 성능 저하의 원인을 **per:place_of_residence**를 예측해야 할 것이 **per:origin**으로 예측을 하고 있었기 때문이라고 생각하였습니다.

그 이유는 **per:place_of_residence**가 Precision and Recall 그래프에서 유독 심하게 기형적인 모양새를 띄고 있었을 뿐만 아니라 Confusion Matrix에서 예측을 실패하는 것을 직접 확인 했기 때문입니다.

그리하여 특정 label의 잘못된 예측을 개선할 수 있는 방법이 무엇인지 알아내는 것에 중점을 맞춰 학습을 진행하였습니다.

○ 나는 어떤 방식으로 모델을 개선했는가?

Evaluation을 하기 위한 데이터를 나눌 때에 Train의 분포를 유지하고자 이러한 기능을 구현해 놓은 **scikitlearn**의 **StratifiedKFold**를 사용하여 Train을 진행하였고, 실제로 그냥 KFold를 사용했을 때보다 눈에 띄는 성능 향상을 보여주었습니다.

또한 raw data의 전처리를 통해 성능을 올릴 방법을 다같이 고민하였고, **per:place_of_residence** 데이터의 sentence에서 random으로 [MASK] 토큰을 넣어주는 작업을 진행하였습니다. (이는 Word 단위로 진행하였고, **sub_entity**와 **obj_entity**는 제외한 random index를 선택하였습니다.)

○ 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

1) **StratifiedKfold** (f1 : 73.7912 -> 76.1570 // auprc : 80.2272 -> 80.3244)

2) **Data augment** (f1 : 76.1570 -> 76.4541 // auprc : 80.3244 -> 81.9893)

결과적으로 저 두 방법 모두 유의미한 점수 향상을 야기하였고, 이로 인해 어느정도 향상된 모델 성능을 보장할 수 있게 되었습니다.

저번 STS Task에서는 데이터를 아무리 전처리하여도 큰 성능의 변화를 얻지 못하였지만, 이번 RE Task에서는 성능 향상에 전처리가 크게 기여하였기 때문에 하나의 Task에서 사용한 방법론이 다른 Task에서는 큰 위력을 발휘할 수도 있다는 것을 깨닫게 되었습니다.

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

저번에는 주로 데이터 전처리와 시각화 역할을 맡았기 때문에 baseline code를 건들이는 것을 많이 경험해 보지 못하였는데, 이번 기회에 KFold를 구현하고, wandb 기능을 추가하면서 전체적인 base code을 분석해 볼 수 있는 기회가 생겨 좋았고, 팀원들이 가설을 구현한 코드도 하나씩 읽어보면서 전체적인 모델 학습의 큰 흐름을 구체적으로 알아갈 수 있었던 것 같습니다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

AI의 특성상 한 번의 훈련이 오래 걸리고, 특히 이번 대회에서 best model로 선정한 모델이 Roberta-large였기 때문에 더욱 시간이 촉박했습니다.

모든 가설을 구현하기는 힘들지만 "이 가설은 정말 효과적일 것 같은데" 라고 생각이 들었던 가설들을 써보지 못하게 되는 것은 항상 아쉬운 것 같습니다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇일까?

다음 대회에서는 시간에 쫓기지 않도록 초기 환경 세팅에 신경을 쓰고, 팀원들과 계획을 세울 때에도 같은 문제점을 겪지 않도록 주의해야겠다고 생각했습니다.

또한 하이퍼 파라미터가 모델의 성능을 크게 바꿀 수 있다는 것을 몸소 느꼈고, 특정 Task에서 안 좋았던 가설이 다른 Task에서는 좋을 수도 있다는 것도 알게 되었기 때문에 지금까지 시도했던 가설들을 폐기하는 것이 아니라 다음 대회에서도 적용할 기회가 있다면 빠르게 적용하여 시간을 단축해보고 싶습니다.

참고문헌

Yuna, H.; Suhyune, S.; Midan, S.; Jungwoo, L.; Heuseok, L. "K-EPIC: Entity-Perceived Context representation in Korean Relation Extraction" Appl. Sci. 2021, 11, 11472.

Wenxuan, Z.; Muhao, C. "An Improved Baseline for Sentence-level Relation Extraction" (2022)