문장 내 관계 추출

Relation Extraction(RE)

Wrap-Up 리포트

2023.05.02 ~ 2023.05.18

NLP-4조(데굴^2)

곽민석_T5014

이인균_T5150

임하림_T5178

최휘민_T5221

황윤기_T5229

1. 프로젝트 개요

a. 프로젝트 주제

프로젝트 주제는 관계 추출(Relation Extraction)로써, 문장의 단어(Entity)에 대한 속성과 관계를 예측하는 문제이다. 관계 추출은 지식 그래프 구축을 위한 핵심 구성 요소로 비구조적인 자연어 문장에서 구조적인 triple을 추출해 정보를 요약하고, 중요한 성분을 핵심적으로 파악할 수 있다. 문장, 단어에 대한 정보를 통해서 문장 속에서 단어 사이의 관계를 추론하는 모델을 학습시켜 모델이 언어를 잘 이해하는 지 평가해보는 것이 이번의 주제이다.

b. 프로젝트 구현 내용, 컨셉, 교육 내용과의 관련성 등

- 주어진 문장 내에서 Entity 사이의 언어적인 관계를 찾는다.
- Entity에 대한 정보는 'word(단어)', 'start_idx(시작 인덱스)', 'end_idx(끝 인덱스)' 가 주어진다.
- 해당 정보들을 이용해서 정해진 관계 Class(Label)를 예측한다.
- Task에 가장 적합한 Pre Trained-Model, 구조를 리서치하고, 아키텍쳐 변경, 하이퍼파라미터 튜닝을 통해 가장 좋은 지표를 갖는 모델을 구성한다.
- 외부정보를 사용하지 않고, 문장 내에서의 수정 또는 정보 추가를 통해 모델의 성능을 개선한다.
- Entity의 위치정보를 최대한 모델이 학습할 수 있도록 Entity Marker, Type Entity Marker 등의 처리를 시행한다.
- Entity에 대한 정보를 모델이 조금 더 이해할 수 있도록, Entity에 대한 Contexts를 추가한다.
- 가장 단일 모델의 성능이 좋은 구조를 바탕으로 여러 Seed를 추가로 학습시켜, 앙상블을 진행하여 결과를 도출했다.
- Train Data에 대한 Confusion Matrix를 표시하여, 모델이 제대로 학습을 하지 못한 부분에 대해서 고민해보고, 해결하기 위한 솔루션을 제시하는 방향으로 진행하였다.

c. 활용 장비 및 재료(개발 환경, 협업 tool 등)

(팀 구성 및 컴퓨팅 환경) 5인 1팀으로 각자에게 지원되는 V100 서버를 활용하여 각자 VSCode 혹은 PyCharm 등 편한 IDE를 사용한다.

(협업 환경 및 의사 소통) Notion, Github, Wandb, Slack, Zoom

d. 데이터셋 및 프로젝트 구조

전체 데이터에 대한통계는 다음과 같다.

- Train Data : 32,470개
- Test Data : 7,765개

데이터셋의 구조는 다음과 같다.

id	sentence	subject_entity	object_entity	label	source
0	〈Something〉는 조지 ...	{'word': '비틀즈', 'start...	{'word': '조지 해리슨', ...	no_relation	wikipedia
1	호남이 기반인 바른미...	{'word': '민주평화당', '...	{'word': '대안신당', 'st...	no_relation	wikitree

id	샘플 순서 id
sentence	subject_entity의 단어와 object_entity의 단어가 포함된 문장
subject_entity	단어, 단어의 시작 인덱스, 단어의 끝 인덱스 및 유형 정보를 포함한 주체 개체
object_entity	단어, 단어의 시작 인덱스, 단어의 끝 인덱스 및 유형 정보를 포함한 목적 개체
label	주어진 문장의 주체 개체와 목적 개체 사이의 관계를 나타내는 레이블
source	문장의 출처(wikitree, wikipedia, policy_briefing)

Label의 목록은 다음과 같다

no_relation	관계없음	org:product	조직과 제품 관계
org:top_members/employee	조직과 상위 구성원 관계	org:founded_by	조직과 설립자의 관계
org:members	조직과 직원의 관계	org:dissolved	조직의 해체
org:number_of_employees/members	조직의 직원 수 또는 구성원 수	org:political/religious_affiliation	조직의 정치적 또는 종교적 소속
org:place_of_headquarters	조직의 본사 위치	org:founded	조직의 설립 여부
org:alternate_names	조직의 대체 이름, 별명	org:member_of	조직과 구성원 관계
per:product	개인과 제품 관계	per:parents	부모관계
per:employee_of	직원관계	per:schools_attended	개인이 다닌 학교
per:children	자녀	per:date_of_death	사망일
per:place_of_residence	거주지	per:date_of_birth	출생일
per:alternate_names	별명, 대체 이름	per:place_of_birth	출생지
per:other_family	다른 가족 구성원	per:place_of_death	사망지
per:colleagues	동료	per:religion	종교
per:origin	출신지	per:title	직책, 칭호
per:siblings	형제자메	per:spouse	배우자

e. 프로젝트 팀 구성 및 역할

이름	역할
곽민석	- 실험을 위한 Sweep 및 config 추가 - 하이퍼 파라미터 튜닝 - 이진분류 및 세부 분류 모델 모델링 - 모델 탐색 및 실험 - 코드 개선
이인균	- EDA - 세부적인 측정 지표 추가 - 클래스에 따른 weighted loss 구현 - Label smoothing 구현 - Loss Function 구현
임하림	- 모듈화 작업 - Bert-base, RoBERTa에 embedding layer 추가하기 - MT5 모델링 - 하이퍼 파라미터 튜닝 - 코드 리팩토링
최휘민	- EDA - 전처리 성능 실험(중복 데이터 제거, 한자처리) - 일반화 성능 실험(중복 단어 조합 제거, Downsampling) - Data Augmentation 실험(Easy Data Augmentation 기법) - SOTA 모델 탐색, 모델 실험, 예측 결과 분석
황윤기	- 논문 리서치 - Data Split - Modeling(RBERT, Improved Baseline) - 전처리 구현(Add Contexts, Typed Entity Marker Punct.) - 하이퍼 파라미터 튜닝 - Github Projects 환경 구성 - Loss Function 리서치 및 구현

2. 프로젝트 수행 절차 및 방법

날짜별 소항목은 Github의 해당 날짜에 merge한 마지막 commit ID이다.

해당 Commit에 각 항목에 대한 작업이 모두 포함되어 있다.

2023/05/09

- 394646e291
  1. 토의된 결과를 바탕으로 코드 모듈화 진행.
  2. 이번 프로젝트 진행 시 사용될 Issue template, Kanban board 등 설정.

2023/05/10

- 845c85d126
  1. 데이터셋을 불러올 때 "Typed Entity Marker Punct" 적용.
     - 이를 위한 Custom model 생성.
  2. 실험 결과의 통일성을 위해 SEED 고정 작업 시행.
  3. 실험 관리를 위하여 config parser 도입.

2023/05/11

- 39e7b0704f
  1. 모델이 어떠한 label에 대해 에러가 발생하는지 알기 위해 Confusion Matrix 적용.
  2. Train data에서 일부를 evaluation data로 사용하는 함수 추가.
  3. 실험 결과의 통일성을 위해 SEED 고정 작업 시행.

2023/05/12

- 5436ba69c1
  1. 다수의 중복되는 데이터에 대해 data loading 시 중복 제거 적용.

2023/05/15

- dc7c0f7a6d
  1. 모델의 성능 비교와 hyper-parameter 튜닝을 위해, Wandb logging과 sweep 적용.
  2. RBert model과 Electra model을 custom model에 추가.

2023/05/16

- dca03a385e

1. Bert-embedding Layer를 추가한 custom model 추가.
2. RBert model에 context를 추가하여 학습하는 부분 추가.
3. Sweep관리를 위한 sweep json parser 적용.

2023/05/17

- 5de824abc5
  1. Entity marker punct 부분의 Token 양식 변경.

2023/05/18

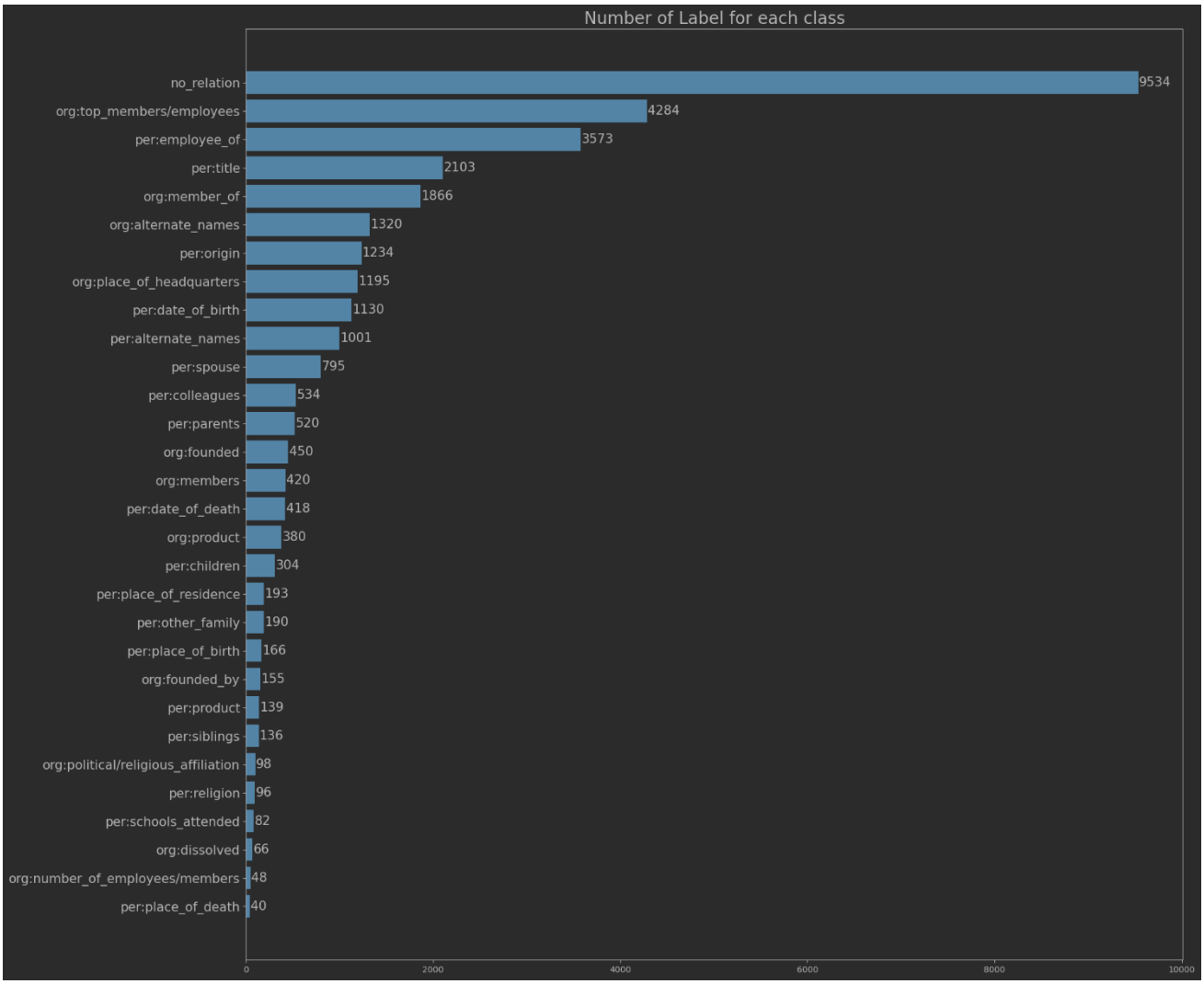
- 3684c0dd9e
  1. Asymmetric Loss Function 적용.
  2. Label smoothing 적용.
  3. Class별 가중치 적용.
- 이전까지의 작업을 바탕으로 결과가 좋았던 모델에 대해 앙상블 도입.

- 점수(micro-f1) 성적 기준으로 가중치를 주어, 앙상블 진행

3. 프로젝트 수행 결과

a. 탐색적 분석 및 전처리 (학습데이터 소개)

I. Class별 Data 개수 분포



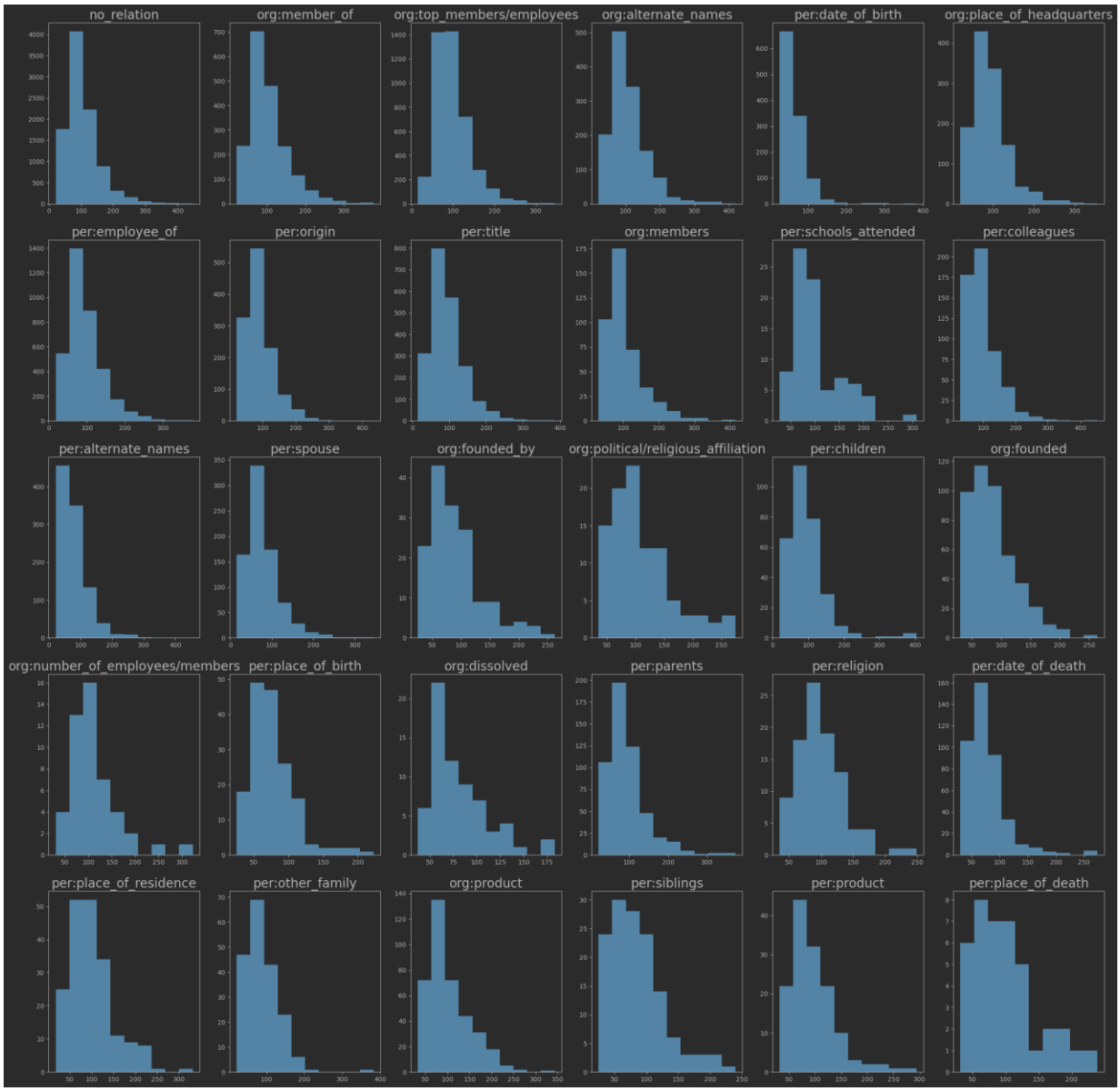
- 우리가 수행할 KLUE-RE Dataset은 Class의 개수 편향이 심하게 되어있었다. `no_relation`의 비율이 압도적으로 많았기에, Negative Label(no_relation)에 대한 False Negative의 처리가 중요할 것으로 예상되었다.
- Label이 no_relation인 데이터를 Downsampling 하여 실험했을 때, Train 데이터로부터 분리된 검증 데이터의 점수에는 미세한 성능 향상이 있었다. 하지만 Test 데이터에서는 성능이 하락하였고 일반화 성능 하락과 Train 데이터와 Test데이터의 데이터 분포 불일치를 예상하였다.
- Label 이 no_relation 인 데이터 중에 중복 단어 조합을 제거하여 Downsampling 하여 일반화 성능을 높이는 방법을 시도해 보았다. EDA 를 하였을 때 Train 과 Test 데이터의 subject 와 object 의 단어 조합이 다른 분포를 가지는 것을 확인하였다. Train 데이터는 전체 32470 개의 데이터이지만 subject 와 object 의 단어 조합은 17727 개로 많은 데이터가 단어 조합을 공유하고 Test 데이터는 전체 7765 개 데이터에서 7616 개의 단어 조합으로 대부분 독립적인 조합을 가졌다. Train 과 Test 둘 다 겹치는 단어 조합은 단 207 개였다. 그래서 문장을 이해하는 것이 아닌 단어의 조합만 보고 no_relation 을 판단하는 가능성을 줄이고자 Label 이 no_relation 인 데이터에 한하여 중복 단어 조합을 다양하게 제거해 Downsampling 을 해보았지만 Test 데이터에서 성능향상은 없었다. 모델이 단어의 조합으로

Label 을 예측할 가능성이 있다는 가설이 틀린것으로 보였고 no_relation 에 대한 Downsampling 자체가 효과가 없었기 때문에 사용하지 않았다.

- 부족한 Label 에 대한 Data Augmentation 으로 EDA(Easy Data Augmentation)[1] 논문에 나오는 4 가지 방법을 구현하고 실험했는데 생성된 문장의 품질이 떨어졌고 학습을 했을 때도 Test 데이터에 대한 성능 하락이 있었다. 그래서 이 방법은 사용하지 않는 것으로 결정했다.

[1] https://arxiv.org/pdf/1901.11196.pdf (EDA, Jason Wei et al. 2019)

II. Class 별 문장의 길이 분포



- Label별 문장 길이는 큰 차이가 없었기에, 데이터 전처리에 있어서 중요도가 떨어질 것으로 예상했다.

III. Unique Sentences의 개수

```
Numbers of Sentences in Train DataFrame :  32470
Numbers of Unique Sentences in Train DataFrame :  28803
```

- Train 데이터에서 중복되는 데이터가 많았다. 동일한 Sentence가 존재하는 데이터의 개수는 7090개 이였고, subject_entity 또한 중복되는 데이터가 9314개, object_entity 또한 9846개였다.
- sentence, subject_entity, object_entity가 모두 동일한 중복데이터가 93개 있었고 이중에 label만 다른 경우도 있었다. 이것을 통해 주어진 데이터가 input과 label이 정확히 일치하지는 깨끗한 데이터가 아닌것을 알 수 있었다. 중복 데이터를 하나만 남기고 나머지를 제거하고 학습을 진행했지만 눈에 띄는 성능 변화는 없었다. 앞서 말한 기존 데이터가 완벽한 일관성이 아닌점과 제거한 데이터가 소수인게 원인으로 생각할 수 있었다.

b. 모델 선정 및 분석

KLUE/Bert-base

- Baseline 점수: micro-F1: 66.0 / auprc: 65.29
- bert는 transformer의 encoder들을 참고해서 만든 모델로 RE-task나 문장 해석에 뛰어난 성능을 보여서 적용해 보았지만 parameter 갯수가 1.1억개로 RoBERTa-large보다 개수가 작아서 성능이 아쉽게 나왔다.

MT5

- Baseline 점수: micro-F1: 53.2 / auprc: 52.29
- MT5 모델은 기존에 있던 T5(text-to-text transformer) 모델을 101개의 다국어에서 작동될 수 있게 google에서 만든 것이다. 기존의 T5 모델이 Transfomrer 구조를 많이 닮아 encoder와 decoder 모두 가지고 있어서 주로 생성모델로 많이 쓰인다. 하지만 이번 task에 적용해보고 싶어서 encoder 부분을 hugginface library에서 가져와서 평가를 해봤지만 성능이 너무 낮게 나와서 안 쓰기로 결정하였다.

KLUE/RoBERTa Small, Base

- Base line 점수: micro-F1: 55.0 / auprc: 48.5
- RoBERTa-large에 비해서 성능이 너무 떨어져서 RoBERTa에 새로운 기법을 빠르게 결과를 얻어서 비교해보는 용도로 주로 이용했다.

KLUE/RoBERTa Large

- base line 점수: micro-F1: 71.4 / auprc: 76.5
- Parameter 개수가 3.5억개로 제일 많고, 성능이 다른 코드와 비교해서 압도적으로 높아서 많이 사용했다.

c. 모델 개요

- 한국어 데이터(KLUE)로 사전학습된 KLUE/RoBERTa-large 모델을 학습 데이터를 이용해 RE-Task에 맞게 미세조정하였다
- SuRE 논문[2]를 참고하여 자연어로 "subject Entity는 {subj}이다. object Entity는 {obj}이다. {subj}는 사람이다. {obj}는 장소이다." 와 같은 문맥을 추가하여 더 많은 정보를 넣었다.
- RBERT 논문[3]을 참고하여, Entity Embedding에 대한 정보를 추가하여 분류 레이어에 더 많은 입력을 받게끔 변형하였다.

[2] https://arxiv.org/pdf/2205.09837.pdf (SuRE, Keiming Lu et al. 2022)

[3] https://arxiv.org/pdf/1905.08284.pdf (Enriching Pre-trained... Shanchan Wu et al. 2019)

d. 모델 평가 및 개선

1. Baseline 코드(BERT-base)
   - 평가: micro-f1 ***66***
   - 문제점 : Entity들에 대한 정보 전달 부족
   - 해결 방향 : Entity Marker[4]를 표기하여 Entity 시작 토큰에 대한 정보 전달
2. Improved Baseline(RoBERTa-Large, [Subject Entity + Object Entity Embedding] + Classifier)
   - 평가 : micro-f1 ***70.8140***
   - 문제점 : Entity에 대한 정보 부족(Type 전달 X 및 전체적인 내용 전달 불가)
   - 해결 방향 : Typed Entity 부분에 해당하는 전체적인 Average 전달
3. RBERT(RoBERTa-Large, [CLS] Representation + [Subject Entity + Object Entity Embedding Average] + Classifier)
   - 평가 : micro-f1 ***72.3577***
   - 문제점 : 특정 Label 같은 경우 예측 확률이 많이 떨어짐
   - 해결 방향 : Entity Type에 대한 Contexts를 추가(SuRE Reference)
4. Include Contexts
   - 평가 : micro-f1 ***73.8058***
   - 문제점 : 일반화 성능 부족
   - 해결 방향 : Seed 조정으로 데이터 분포를 다르게 하여, 앙상블 진행
5. Seed Ensemble
   - 평가 : micro-f1 ***75.3651***

e. 일반화 성능 올리기

- 일부 라벨에서는 train 데이터와 test 데이터에서 분포에 차이가 있었기 때문에 더 나은 일반화 성능을 위해 손실 함수를 계산할 때 클래스별 가중치를 줘서 집중적으로 학습을 시켰다. 도입 결과 퍼블릭 리더보드에서는 큰 점수 차이가 없었고 프라이빗 리더보드에서의 성능 하락폭을 2.0에서 0.5로 줄였다.
- 일반화 성능을 올리기 위해 데이터 편향을 줄이는 downsampling 과 data augmentation 을 진행하였지만 Test 데이터에 대한 성능을 떨어졌기 때문에 사용하지 않았다.

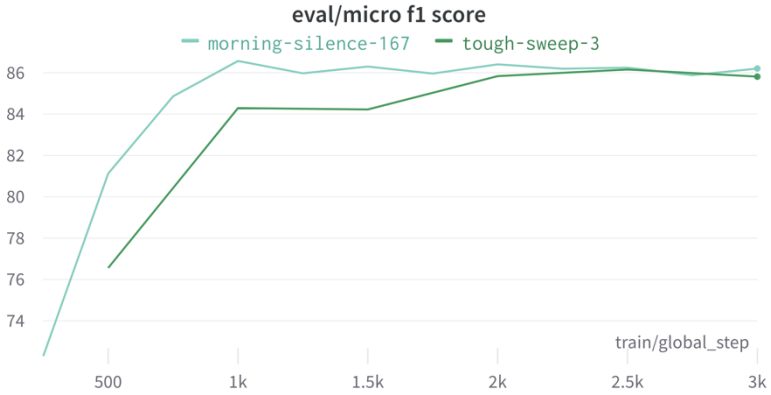
f. 시연 결과

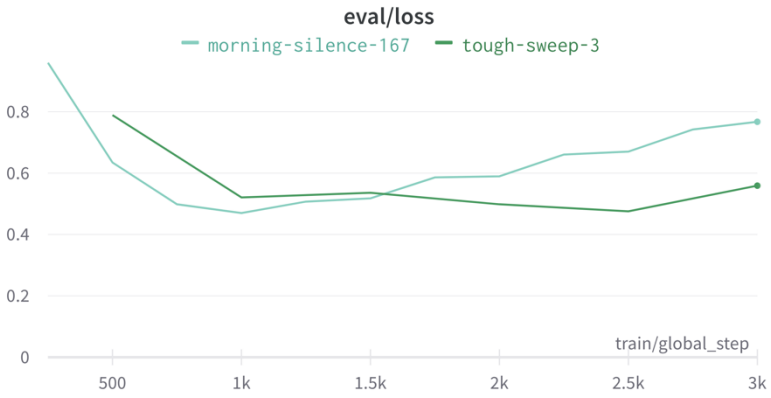
- train.csv에 대한 모델의 예측 결과
  - 문장: 사랑스러운 '프린세스 프링'의 이름은 봄의 공주님: Princess(s)Pring이란 뜻으로 탄생, 부활, 청춘 등 여러 가지 의미를 담아 생일왕국의 공주님의 이름이 되었다.
  - Subject: ('프린세스 프링', 'PER')
  - Object: ('공주', 'POH')
  - 예측: per:alternate_names

[4] https://arxiv.org/pdf/2102.01373v4.pdf (Wexuan Zhou et al. 2022)

- 예측 확률 상위 3개: {'per:alternate_names': 0.42, 'per:title': 0.11, 'no_relation': 0.07}

- Wandb training 결과





4. 자체 평가 의견

- **잘한 점들**
  - Github를 이용해서 main code를 계속 발전시키는 형태가 좋았다.
  - Github project를 통해서 서로의 진행상태를 알고 있어서 점진적으로 발전시키는 형태가 좋았다.
  - Github Issue와 PR을 적극적으로 이용해 정보 공유가 가능해서 좋았다.
- **시도했으나 잘 되지 않았던 것들과 개선사항**
  - 성능 개선이 되었는지 지표를 꼼꼼하게 기록하지 못했다. Wandb를 더 적극적으로 쓰고 기록도 꼼꼼히 해야 한다.
  - 문제와 가설을 정의하려고 했으나 구체적으로 잘 구상하지도 못했고 적극적으로 하지도 못했던 것 같다. 단기간에 해결될 문제가 아니라서 꾸준히 체계적으로 접근하기 위해 노력해야 한다.
  - 시도한 Data augmentation이 성능 향상이 이루어지지 않았다. 좀 더 많은 방법을 시도해야 한다.
- **아쉬웠던 점들과 개선사항**
  - 맡은 일을 진행할 때 스스로 하는 건 좋지만 너무 길어진다면 팀적으로나 개인적으로 좋지 않은 거 같다. 프로젝트 시간 관리 툴을 도입해야 한다.
  - 서로가 찾은 정보를 확인하고 알아가는 시간이 적극적으로 있으면 좋을 것 같다. 별도의 notion 페이지를 만들어서 체계적으로 공유해야 한다.
  - Wandb 로깅처럼 성능과 상관은 없지만 프로젝트에서 중요한 일을 하는 코드를 구현하는 게 늦어져서 프로젝트를 할 때 불편함이 있었다. 우선 순위를 적고 그에 맞춰서 작업을 하도록 해야한다.
  - 성능 지표를 공유하기 위한 작업이 늦었다. 이 역시 우선 순위를 맞춰서 필요한 작업을 먼저 해야 한다.
- **프로젝트를 통해 배운 점 또는 시사점**
  - 논문 리서치를 통해 성능을 끌어올릴 수 있고 다양한 아이디어와 코드 구현을 배울 수 있다는 점을 배웠다.
  - 다양한 데이터 증강을 시도했지만 실패했다. 이후 다른 조의 발표를 보면서 더 나은 아이디어를 얻을 수 있었다.

5. 개인 회고

곽민석

1. 성능 향상을 위한 노력

- 이번 Task는 이전 STS와 다르게, 데이터 셋 자체가 크고 모델에 Feeding하는 방식이 일반적으로 Bert에 Pre-training 시킬때와는 조금 달라 처음 접했을 때 어려움이 있었다.
- 먼저 시도해본 것은, GPT-NER: Named Entity Recognition via Large Language Models에서 참조하여 개체명 구분자를 추가하여 실험해보았다.
- 여기서 한가지 실수가 있었는데, 먼저 논문에 대해 너무 표면적으로 읽어보았다.
- 또한 해당 논문에 대해 예시 코드가 없어 시도하는데 어려움이 있었다.
- 위의 문제로 인하여 멘토님이 추천해준 방식을 시도해보았다.
- 먼저 데이터에 대해. no-relation인지 relation이 있는 데이터인지 먼저 판별한다.
- elation이 있는 데이터로 판별되는 경우 세부적인 relation(no-relation을 제외한 29개의 label)을 판단한다.
- 실험을 진행하게 된 이유로는 데이터의 심각한 불균형이였다.
  - Label의 갯수가 no-relation이 다른 label 보다 월등히 많았기 때문이다.
- 이 방식에 대해 먼저 no-relation을 제외한 데이터에 대해 실험을 진행했었다.
- 해당 데이터에 대해 Micro-F1 Score가 약 91%가 나왔다.
- 위의 결과에 대해 진행해도 좋다 생각되어 binary classification도 진행하였다.
- 해당 데이터에 대해서도 Micro-F1 Score가 약 91%가 나왔다.
- 이 두 모델을 relation이 나왔을때 relation을 뽑는 방식과 binary probability와 나머지 29개의 label에 대한 probability를 곱하여 softmax를 하는 방식 두 가지를 해보았으나, 결과가 좋지 않았었다.

2. 전과 비교해서, 새롭게 시도한 것과 효과

- 이번에 하게되는 Task에 대한 논문을 먼저 찾아봤었다.
  - 이를 통해, 다른 연구에서 이번 Task를 어떤 방식으로 접근할 것인지 알게 되어 방향성을 얻게 되었다.
- Github Issue와 PR을 이용하여 정보와 코드를 공유하였다.
  - 이전부터 적용하자는 얘기가 나왔었는데, 적용시켜보니 서로 어떤 작업을 하고 있고 앞으로 어떤 작업이 남았는지 볼 수 있어서 좋았었다.

3. 한계점과 아쉬운 점 그리고 해결 방안

1. 처음 GPT-NER 논문을 접근하였을 때 해당 논문의 코드 제공 여부를 살펴보았어야 했다.
   1. 코드가 주어졌을 때 조금 더 접근하기가 쉽기 때문에 코드 제공 여부를 보고 읽어보라 멘토님이 조언하셨다.
2. 다른 팀원의 기술적 follow up이 어려웠다.
   1. 변명아닌 변명이지만, 내가 하고있는 분야와 다른 팀원이 작업하고 있는 분야가 햇갈리거나, 잘 까먹게 된다.
      3. 이렇게 됐을 때 팀원들에게 걸림돌이 될수 있는데, 최대한 작업 내용에 대해 공유할 때 질문을 많이 하는식으로 지식의 동기화가 필요할것같다.

4. 논문을 엉성하게 읽었다.
    1. 논문의 방식을 따라하려면 먼저 논문의 내용을 최대한 이해해야한다. 전부 다는 아니여도, 어느정도 정리는 된 상태에서 작업에 들어가야할것 같다.
    2. 그렇지 않았을 경우 일을 여러번하거나, 좋은 정보임에도 성능이 나지 않아 버려질수 있기 때문이다.

4. 다음 프로젝트에서…

저번 STS Task의 경우 어느정도 기여도도 있었고, 성능이 잘 나와 Project의 희망편이였다면, 이번은 조금은 깨져보고 배울점이 많았던 프로젝트였던 것 같다.

다음 프로젝트에서는 논문 선택에 대해 있어서 어느정도 기준을 가지고 고르고, 고른 논문에 대해 정리가 끝난 후 프로젝트에 접근하는 방식으로 진행할것이다. 이러한 flow는 이전 section에서 본것과 같이 문제점에 대한 어느정도의 방책으로 만들어보았다.

또한 팀원들의 작업물에 대해 follow up 이 어려웠었는데, 이 점에 대해서 최대한 질문을 많이 던지거나, 작업물에 대한 논문 리뷰를 해보는식으로 접근해봐야 할것 같다. 팀원들의 작업물이 어쩌면 내가 진행하고 있는 작업에 큰 영향을 끼칠수 있고 또 해당 작업물에서 영감을 얻을 수 있기 때문이다.

이인균

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

저번 프로젝트 문장 분류에서는 EDA와 데이터를 중심으로 다뤘기 때문에 이번 프로젝트에는 모델을 직접 수정 및 개선하는 것이 목표였다. RE-MC 모델(Baek H-R, Choi Y-S. Enhancing Targeted Minority Class Prediction in Sentence-Level Relation Extraction. Sensors. 2022; 22(13):4911. https://doi.org/10.3390/s22134911)에 대한 논문을 읽고 저자의 깃허브에 나온 코드를 보며 이해하기 위해 노력하였다.

강의 영상에서 배웠던 다양한 기법들을 직접 적용하는 것도 또 하나의 목표였다. Label smoothing을 다양한 방식으로 적용하였고 코드를 더 깊이 이해할 수 있었다.

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

나를 포함해서 모든 팀원이 깃허브를 더 잘 쓰기 위해 노력했고 이 점은 매우 성공적이었다. 팀이 5명으로 구성된 가운데 나는 32%의 issue를 적을 정도로 pr, 이슈, 칸반보드 등의 기능을 적극적으로 활용하였다. 이를 통해 편리하게 서로의 진척도를 확인할 수 있었고 이후 회사에서 쓸 툴에 익숙해질 수 있었다.

작업에 도움이 되는 코드를 구현하였다. 파일을 저장할 때 time 함수를 써서 유일하면서도 이해할 수 있는 이름을 만들었고, 혼동 행렬을 출력하는 함수를 구현하여 현재 모델의 아웃풋의 성능을 다방면으로 분석할 수 있었고, mixed-precision 기능을 활용해 모델 학습 시간을 40% 단축했다. 이러한 변경 사항들을 통해 이전 프로젝트보다 더 편하게 작업할 수 있었다.

기존에는 단순히 인자값을 변경하여 다양한 손실 함수 변경을 시도하였으나, 이번에는 직접 focal loss를 구현하였다. 더 많은 자료를 찾아봐야 했고 코드를 더 주의깊게 보았어야 했다. 그러나 성능은 아주 약간이나마 낮아졌기 때문에 대신 황윤기 팀원이 구현한, 이것과 유사하면서 성능이 좋아졌던 asymmetric loss function을 사용하기로 하였다.

모델을 변경할 때 3번가량 실험을 하여 성능 지표가 정말로 개선이 바뀌었는지 혹은 단순한 운에 의한 것인지 파악하였다. 이를 통해 변경 사항을 main 브랜치에 적용시킬지에 대한 여부를 확실하게 정할 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

단순히 허깅페이스에서 불러오는 것이 아닌 모델을 직접 수정하는 것을 시도했다. 하지만 논문과 깃허브 코드를 이해하는 데에 시간이 너무 오래 걸렸고 그것을 기존 베이스 라인 코드 형식으로 바꾸는 데에는 더 오랜 시간이 걸릴 듯하여 해당 개선을 시도하지 못했다. 강의를 듣는 데에 시간이 오래 걸려 프로젝트에 쓸 여유 시간이 부족했던 점이 아쉬웠다. 논문을 빨리 잘 이해하는 법은 시간이 가면 자연스럽게 익힐 것이라 생각한다.

이번 프로젝트에서는 모델링에 집중하고 싶어서 place_of_residence에 큰 오차가 있는 것을 발견했지만 데이터의 개선점을 주의깊게 고려하지 않았다. 이후 다른 조의 프로젝트 발표 내용을 듣고 데이터가 성능에 큰 영향을 미친다는 점을 깨달았고 팀원들과 더 의견 공유를 하지 않았던 점이 아쉬웠다.

Label smoothing을 허깅페이스 내부 인자 형식으로 구현할 때는 오류가 생겼었다. 이 점을 해결하기 위해선 코드를 오래 봐야할 것이기 때문에 일단은 디버깅을 시도하지 않았다. 디버깅을 직접 시도했으면 허깅페이스 코드에 대해 더 깊은 이해를 할 수 있었을 텐데 그러지 않았던 점이 아쉽다. 이후 모델이 바뀐 뒤 그 형식에 맞춰서 직접 구현을 할 때는 오류가 생기지 않았어서 이 방식으로 구현을 하였다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

논문을 읽는 데에 조금이나마 익숙해졌기 때문에 다시 한번 모델링 개선을 시도하고 싶다.

버그가 생겼을 때 허깅페이스 내부 코드를 들여다보지 않았던 점이 아쉽다.

임하림

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- Baseline 코드를 더 명확하게 이해하기 위해서 기능에 따른 모듈화를 진행했다.
- 성능이 잘 나오는 RBERT 말고 앙상블을 위해 다른 모델을 쓰고 싶다는 생각에 다른 모델들을 가져오거나 기존 모델을 개량하는 작업들을 했다.
- MT5에서는 encoder와 decoder를 다 가지고 있어서, encoder 부분을 가져와서 task에 적용시켰다.
- entitiy에 대한 정보가 부족한 것 같아, RoBERTa-large와 bert-base 모델을 위해 entitiy에 해당하는 token에 1을 추가해서 input의 차원수를 늘려 tokenizing을 진행한 다음 huggingface에 있는 코드를 참고해서 input과 embedding layer를 추가하고 entity를 강조할 수 있도록 추가 했다.
- Entity marker와 같이 적용하면 효과가 훨씬 좋다고 해서 같이 적용해보았다.

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

base line 코드와 어떤 task인지 정확히 인지하고 넘어가지 않았어서 한번 자세히 읽고 이해하려고 노력을 해봤다.

그 결과로 어떤 코드인지 잘 이해하고 어떤 방식으로 작동이 되는 지 알았지만, 질문을 던질 정도의 이해나 고뇌까지 가지 못했던 것 같다. 그래서 반드시 이해해야 했는 prompt에 대한 인식이 없었던 것 같다. 그래서 다양한 idea를 던질 수가 없었고, 다른 캠퍼 분들이 idea를 던지고 나서야 prompt가 이해를 해야 하는 영역이란 것 자체를 인식했던 것 같다. 다음에는 base-line 코드에서 사소한 것 하나까지 질문을 던지는 시간을 가지고 싶다.

Huggingface에 있는 library 내부를 건드려보았다.

이전에는 모델을 가져와서 쓰고 hyperparameter를 조정하는 방법만 사용했어서, 이번에는 모델 내부를 건드려 보고 싶다는 생각으로 huggingface에서 구현 되어있는 코드 내부를 보게 되었다. 처음으로 논문과 코드를 같이 봤는데 이해가 훨씬 더 빨랐고 어떻게 작동되는 지 직관적으로 확인할 수 있어서 좋았다. 잘 구현되어 있는 코드와 항상 같이 봐야겠다는 생각을 했다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

모델의 구조를 완벽하게 이해는 못 하더라도 모델 자체의 입출력과 작동방식을 짚고 넘어 가야겠다는 생각이 들었다.

Embedding layer를 추가하는 방법을 알기까지 너무 오래 걸렸다. bert의 input과 embedding 방식만 알아도 쉽게 해결되는 그림이었는데, 모델의 구조와 작동 방식을 짚지 않아서 오래 걸렸던 것 같다. 모델을 세부적으로 공부하고 적용을 해야 겠다는 생각을 했다.

문제 정의를 완벽하게 못 했다.

무턱대고 성능을 올리는 게 중요한 게 아니라, 어떤 문제인 지 확인하고 문제를 더욱 구체화해서 진행을 했어야 했는데 문제를 잘 정의하지도 어떻게 문제를 파악해야 할 지도 잘 몰랐던 것 같다.

참고할만한 방법조차 몰랐다.

어느 정도의 기반을 가지고 시작해야 했는데 그 기반이 없이 시작하니, 문제를 어디서 찾아야 할 지도 어떻게 찾아야 할지도 몰랐던 것 같다. 심지어 레퍼런스를 많이 찾았지만 정리를 하지 않아 휘발성이 강해서 더 아쉬웠다.

팀원도 있고 멘토님도 계시는데 너무 혼자서 해보려고 했던 것 같다.

팀원들에게 피해를 끼치지 않기 위해서 내가 맡은 일에 대해서 책임감을 가지는 건 좋지만 너무 길어지면 팀원들에게나 나에게나 손해인 것 같다. 앞으로 문제가 늘어진다 싶으면 팀원들과 멘토님들에게 방법적으로 어떻게 할 지, 너무 밑바닥이라도 일정 기간이 지나면 구체적으로 물어 봐야할 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

하루 단위로 내가 했던 것들을 기록해서 나를 객관적으로 확인을 하고 싶다.

찾았던 지식의 근원을 다 기록해야겠다는 생각을 했다.

코드를 한 줄 단위로 왜 작성했는 지에 대한 고뇌를 더 깊게 하는 것을 시도해보고 싶다.

아는 범주에서 다 확인을 했다면 최신 논문을 읽고 적용하는 것을 시도해보고 싶다.

최휘민

a. 성능 향상을 위한 노력

- 저번 대회에서 주로 모델링과 앙상블을 했었기 때문에 이번 대회에서는 이전에 주로 하지 않았던 데이터 EDA, 전처리 부분을 하고싶어서 주로 데이터를 다루는 역할을 하였고 저번 대회와 달리 최대한 논문과 합리적인 이유를 기반으로 행동하였습니다.
- 대회가 시작하고 SOTA 모델을 찾기 위해 한국어 데이터를 사용한 transformer 사전학습 모델 성능비교 논문을 참고하였고 또한 KLUE 데이터 github에 있는 transformer 성능 비교를 참고하여 팀원들과 어떤 모델이 좋을지 논의하였습니다.
- EDA를 진행하여 결측치, 데이터 편향, 데이터길이, 중복데이터, unknown 토큰, 데이터 조합 등을 분석하여 팀원과 공유하고 성능 향상을 어떻게 해야 할지 논의하였습니다.
- 중복데이터를 처리하는 전처리를 진행하였습니다.
- Label 편향을 처리하기 위한 no_relation의 downsampling과 부족한 label에 대한 Easy Data Augmentation 을 시도하였습니다.
- 좀더 합리적인 downsampling 방법을 시도하고 일반화의 성능을 높일 수 있게 subject, object의 word 단어 조합에 따른 no_relation의 downsampling을 시도하였습니다. 모델이 문맥을 이해나는 것이 아닌 단어의 조합으로 label 판단하는 가능성을 제거하는 것으로 합리적인 이유라고 생각되어 시도하였습니다.
- 학습된 모델로 나온 결과를 분석하여 어느 부분에서 잘 예측을 하고 못하였는지 분석하여 직접 해당 데이터 들을 보면서 팀원과 정보를 공유하였습니다.
- 결과 분석을 통하여 한자 데이터에 뜻과 음을 붙여주는 기능을 구현하였습니다.

b. 한계점과 아쉬운 점 그리고 해결 방안

- EDA와 결과 분석을 기반으로 시도했던 데이터 처리들이 모두 성능향상으로 이루어 지지 않아서 아쉬웠습니다. 논문과 합리적인 이유로 시도하였지만 Test 데이터에서 성능이 떨어졌고 Train과 Test 데이터의 데이터 분포 불균형으로 결론을 내렸는데 불균형이여도 좀더 다양한 방법의 시도를 해볼 수 있을 것이라 생각하여 아쉽습니다.
- 성능이 효과 있는지 없었는지 좀더 학습을 많이 해보고 판단을 해야 한다고 생각하지만 시간적 여유로 그러지 못하였습니다. 다음 대회에서 gpu 스케줄러 도입해 좀더 효과적인 학습이 필요하다고 생각합니다.
- 너무 세세한 부분을 테스트해가면 코딩을 하여 기능 구현에 시간을 너무 오래 걸렸고 그래서 중요한 정보 공유도 몇개는 늦었다고 생각하고 잘 정리하여 다음에 빠르게 활용할 수 있게 만들겠습니다.
- 결과 예측에서 구분을 잘 못하던 label에 대하여 사람도 구분하지 못하는 label 이라 생각하고 그냥 넘어갔는데 좀더 많은 시도를 그래도 해봤어야 했다고 생각합니다.
- 멘토님이 생성모델을 사용한 데이터 증강방법을 예기해 주셨는데 시도하지 못하여 아쉽습니다.

c. 다음 프로젝트에서

- gpu 스케줄러 도입해 좀더 많은 실험과 검증을 해봐야 한다고 생각합니다.

- 좀더 최신 논문을 기반하여 방법론을 탐색해보아야 한다고 생각합니다.
- 가능하면 생성모델을 도입해보고 싶습니다.

황윤기

a. 성능향상을 위한 노력

이번 Task를 수행하면서 비슷한 데이터셋에서 같은 Task를 수행한 모델의 참고를 먼저 시작한 것 같다. Baseline 코드를 수행하고, 이해하면서 좋은 성능을 낸 모델이 어떤 차이점이 있는지 분석하는 시간을 먼저 가졌다. 그리고 몇가지 변경할 사항을 미리 정해두고 학습을 시작했다. Encoding 모델, 모델 구조, 데이터 전처리 구조, Loss함수 변경 등으로 제한하였다.

지표비교를 위해 Baseline 코드의 학습과, 데이터 구조 변경을 시행하여 성능 향상을 조사했다. 또한 Encoding 모델 변경을 통해서 성능 향상폭을 비교하고, 좋은 데이터 구조 변경 방법을 조사했다. 여러가지를 한번에 변경하지 않고, Baseline코드를 기준으로 각각을 변경하여 좋은 성능을 내는 방법들을 조합해 나가는 방향으로 진행하도록 실험을 했다.

b. 한계점과 아쉬운 점

해당 방법은 지표를 개선하는데 효과적으로 작용했다. 하지만 어느 수준이 넘어가면서는, 모델의 성능이 올라가지 않았다. 모델의 성능 한계를 느낀 것이다. 그럴 때 어떤 방식으로 모델 구조를 개선하고, 데이터 구조를 변경해야 할지 조금 더 좋은 아이디어가 나오지 않은 것 같다. 비슷한 모델 구조가 아닌 다른 모델 구조를 설계한 논문에서 참고하여 구성도 해보았지만, 성능이 계속해서 그대로 정체됐을 때 나만의 새로운 효과적인 방법을 설계해봤어야 하는데, 논문에 사로잡혀 그런 시도가 부족했던 것 같다.

일례로, Entity에 대한 Contexts를 도입한 것도, T5모델을 사용하여 관계 예측을 한 모델(SuRE)에서 참고하여 도입한 것인데, Contexts를 도입할 때 영어로 관계를 설명해준 것이다. 하지만 레퍼런스에 억메이지 말고, 한글로 Entity에 대한 설명을 넣어주면 어땠을까하는 아쉬움이 남는다.

c. 다음 프로젝트 시도

SOTA 레퍼런스를 참고한 것은 다음에도 똑같이 진행할 것 같다. 하지만 레퍼런스와는 별개로 독창적인 방법을 논리적으로 제시하여, 우리만의 개선사항을 적용해보면 더 좋을 것 같다.