문장 내 개체간 관계 추출 분석 보고서

1. 서론

1.1 과제 개요 및 목표

본 과제는 문장 내 두 엔티티(entity) 간의 관계를 자동으로 추론하는 Relation Extraction(RE) 태스크를 다룬다. 관계 추출은 문장 수준의 비정형 텍스트로부터 구조적 정보(예: 인물-소속 조직 등)를 도출하는 핵심 NLP 기술로, 지식 그래프 구축, 질의응답 강화, 문서 요약, 추천 시스템 등에 폭넓게 활용될 수 있다. 본 분석 보고서의 목표는 다음과 같다.

- **EDA 결과 기반 전략 수립**: 사전적인 데이터 분석(EDA)을 통해 라벨 분포, 문장 길이, 엔티티 타입 등의 특징을 파악하고, 이를 모델 설계 및 하이퍼파라미터 설정, Loss 함수 선택 등에 반영한다.
- **단계적 접근을 통한 성능 개선**: 기본 RoBERTa 기반 파인튜닝에서 시작하여, EDA로 도출한 문제점(예: 불균형, 희소 클래스 식별 난이도)들을 하나씩 해결해 나가는 과정을 체계적으로 정리한다. 단순 Cross Entropy Loss에서 Focal Loss 적용, 엔티티 마커 삽입에서 엔티티 스팬 풀링 및 Attention-based Pooling으로 확장, 더 나아가 Ensemble 기법 도입을 통해 성능 개선을 이끌어내는 전 과정을 단계별로 분석하고 문서화한다.
- **데이터 불균형 문제 해결**: 실제 수집된 텍스트 데이터에서는 `no_relation` 클래스가 과도하게 많고, 특정 관계 클래스를 대표하는 샘플이 극히 적은 불균형 상황이 흔하다. 본 분석 보고서에서는 이러한 불균형 하에서도 희소 클래스를 제대로 인식할 수 있는 모델을 개발하는 것을 주요 목표로 삼는다.

최종적으로, 본 분석 보고서는 EDA를 토대로 단계적 개선을 거친 RE 모델을 제안하고, 그 과정에서 얻은 경험을 정리하여 향후 연구 방향을 제시하는 것을 목표로 한다.

1.2 데이터셋 특성 및 문제 정의

데이터셋은 한국어 문장과 해당 문장 내에서 주어진 엔티티 쌍(subject entity, object entity), 그리고 그 사이의 관계 라벨로 구성된다. 주요 특성은 다음과 같다.

	id	sentence	subject_entity	object_entity	label	source
0	0	〈Something〉는 조지 해리슨이 쓰고 비틀즈가 1969년 앨범 《Abbey R...	{'word': '비틀즈', 'start_idx': 24, 'end_idx': 26...	{'word': '조지 해리슨', 'start_idx': 13, 'end_idx':...	no_relation	wikipedia
1	1	호남이 기반인 바른미래당·대안신당·민주평화당이 우여곡절 끝에 합당해 민생당(가칭)으...	{'word': '민주평화당', 'start_idx': 19, 'end_idx': ...	{'word': '대안신당', 'start_idx': 14, 'end_idx': 1...	no_relation	wikitree
2	2	K리그2에서 성적 1위를 달리고 있는 광주FC는 지난 26일 한국프로축구연맹으로부터...	{'word': '광주FC', 'start_idx': 21, 'end_idx': 2...	{'word': '한국프로축구연맹', 'start_idx': 34, 'end_idx...	org:member_of	wikitree
3	3	균일가 생활용품점 (주)아성다이소(대표 박정부)는 코로나19 바이러스로 어려움을 겪...	{'word': '아성다이소', 'start_idx': 13, 'end_idx': ...	{'word': '박정부', 'start_idx': 22, 'end_idx': 24...	org:top_members/employees	wikitree
4	4	1967년 프로 야구 드래프트 1순위로 요미우리 자이언츠에게 입단하면서 등번호는 8...	{'word': '요미우리 자이언츠', 'start_idx': 22, 'end_id...	{'word': '1967', 'start_idx': 0, 'end_idx': 3,...	no_relation	wikipedia

데이터 규모 및 구성

- 문장 수 :
  - 학습 데이터 : 32,470 건
  - 테스트 데이터 : 7,765 건
- 관계 클래스 라벨 수: 30개

```
'no_relation',
'org:alternate_names',
'org:dissolved',
'org:founded',
'org:founded_by',
'org:member_of',
'org:members',
'org:number_of_employees/members',
'org:place_of_headquarters',
'org:political/religious_affiliation',
'org:product',
'org:top_members/employees',
'per:alternate_names',
'per:children',
'per:colleagues',
'per:date_of_birth',
'per:date_of_death',
'per:employee_of',
'per:origin',
'per:other_family',
'per:parents',
'per:place_of_birth',
'per:place_of_death',
'per:place_of_residence',
'per:product',
'per:religion',
'per:schools_attended',
'per:siblings',
'per:spouse',
'per:title'
```

- no_relation 클래스 비율: 29.36%

불균형한 라벨 분포

`no_relation` 클래스가 다수이며, 몇몇 관계 클래스는 매우 적은 비중을 차지한다. 예를 들어, 가장 빈도가 높은 `no_relation` 클래스는 29.36%, 그 다음으로 빈도가 높은 `org:top_members/employees` 클래스는 13.19%의 비중을 차지한다. 희소한 클래스(예: per:place_of_death)는 0.12%만을 차지한다. 이는 모델이 특정 클래스에 편향되거나 희소 클래스를 제대로 학습하지 못하는 문제를 유발한다.

문장 특성 및 엔티티 타입

문장 길이는 평균적으로 100자 내외로 분포하며, 400자를 넘는 긴 문장도 존재한다. 엔티티 타입은 `subject_type` 의 경우 인물(PER), 조직(ORG) 두가지에 대한 데이터만 존재하고, `object_type` 의 경우 인물(PER), 조직(ORG), 장소(LOC) 등 다양하게 존재한다. 특정 관계는 특정 타입 조합에서만 주로 발생한다. 이로 인해 엔티티 타입 정보 활용, 엔티티 마커 삽입 등의 전략이 유용할 것으로 예상된다.

이와 같은 데이터셋 특성은 모델 설계 시 다음과 같은 문제 정의를 명확히 한다. 즉, 다양한 엔티티-관계 패턴을 안정적으로 파악하면서도 `no_relation`에 치우치지 않고 희소 관계를 인식할 수 있는 모델을 구축해야 한다는 것이다.

1.3 평가 지표 및 과제 조건

본 과제에서는 모델 성능을 측정하기 위해 다음과 같은 평가 지표를 사용한다.

[평가 지표]

- `no_relation` **제외 Micro-F1**:

`no_relation` 클래스를 제외한 나머지 관계 클래스에 대해서만 Micro-F1 스코어를 계산한다. 이 지표를 활용하면 모델이 단순히 모든 것을 `no_relation`으로 예측하는 전략을 피하고, 실제로 관계가 존재하는 클래스를 제대로 식별하는 능력을 평가할 수 있다. Micro-F1는 모든 클래스의 정답 수의 합을 기반으로 평가하므로, 클래스별 불균형을 감쇄하는 효과를 가져와 희소 클래스 식별 능력을 향상시키는 전략 효과를 잘 반영한다.

- **AUPRC (Area Under the Precision Recall Curve, 모든 클래스 포함)**:

보조 지표로 AUPRC(Precision-Recall 곡선 하의 면적)을 사용한다. AUPRC는 각 클래스별 정확도, 재현율 관계를 종합적으로 살펴볼 수 있으며, 희소 클래스의 low recall 상황에서 precision 변화를 파악하는 데 도움이 된다. 이 경우에는 모든 클래스를 포함하므로, `no_relation` 의 성능도 함께 평가할 수 있기 때문에 모델 전반적 확률 분포의 품질을 진단하는 데 유용하다.

[과제 조건]

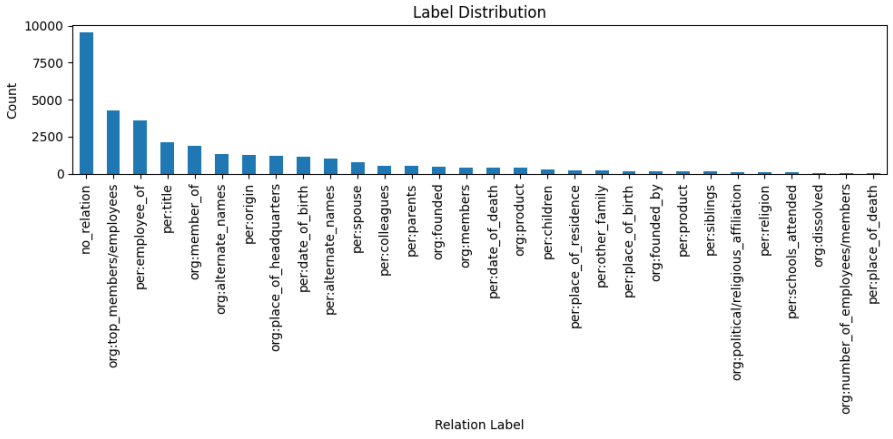
- 학습용 데이터(train.csv)는 총 32,470개 문장, 평가용 데이터(test_data.csv)는 총 7,765개 문장으로 구성되며, test_data.csv는 라벨 정보가 100으로 블라인드 처리됨.
- `dict_label_to_num.pkl`과 `dict_num_to_label.pkl`을 통해 총 30개 클래스의 문자 라벨-숫자 라벨 매핑을 반드시 준수해야 함.
- 외부 데이터 사용은 금지하며, train.csv만 활용 가능하지만 학습 데이터 기반의 데이터 증강은 허용됨.
  - test_data.csv를 분석하거나 레이블링하는 행위는 금지하며, 오직 모델 예측을 통해 submission.csv를 제출해야 함.
- submission.csv는 `id, pred_label, probs` 컬럼을 포함해야 하며, 이를 토대로 Public/Private 리더보드를 통해 성능을 평가함.

2. 데이터 탐색(EDA)

본 장에서는 주어진 train.csv 데이터를 중심으로 라벨 분포, 문장 길이, 엔티티 타입 등을 분석한다. 이를 통해 데이터의 특성과 잠재적 문제점을 파악하고, 이후 모델링(학습 전략, Loss 함수 선택, 하이퍼파라미터 설정 등)에 반영하기 위한 기반을 마련한다.

2.1 라벨 분포 분석 및 불균형 정도 파악

관계 추출 문제에서 라벨 분포는 모델 성능 및 학습 전략 결정에 중요한 영향을 미친다. 본 과제의 train.csv 데이터를 대상으로 라벨 빈도를 조사한 결과, 전체 30개 클래스 중 일부 클래스에 데이터가 편중되어 있으며, 특히 `no_relation` 클래스 비중이 상당히 높은 것으로 나타났다. 이를 통해 불균형 문제의 심각성을 확인할 수 있다.



라벨 분포 시각화 결과

주요 포인트

- `no_relation` 클래스: 전체 데이터의 약 30%로서, 단일 클래스 중 가장 빈도가 높다.
- 상위 3개 클래스: 전체 데이터의 절반 이상을 차지, 극도로 편중된 분포 확인.
- 희소 클래스(하위 10여 개 클래스): 하위 10개 클래스 수의 합이 전체의 3%밖에 되지 않음. 학습 시 모델이 해당 클래스를 거의 인식하지 못할 가능성 존재.

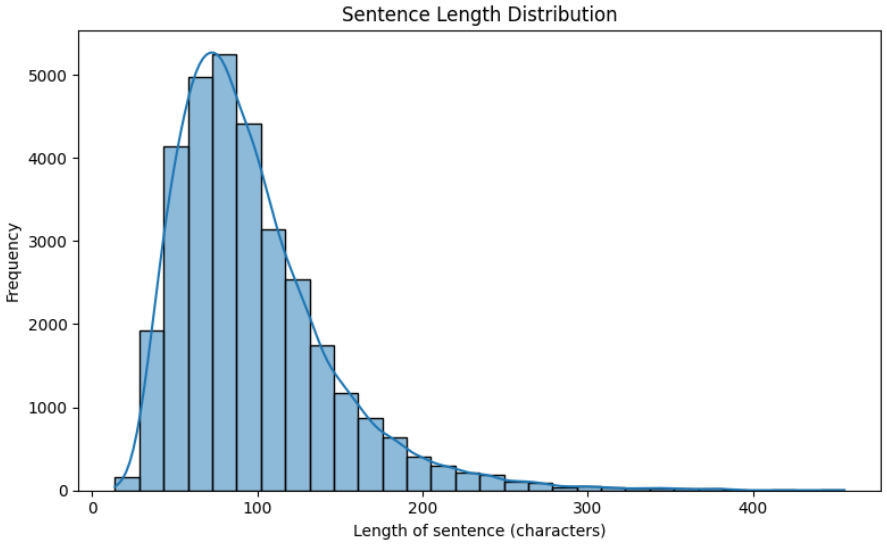
EDA 결과 활용 방안

- 불균형 해소 전략(예: Focal Loss, Label Smoothing)의 필요성을 확인.
- 희소 클래스 식별력 강화를 위해 데이터 증강과 같은 기법 검토.

2.2 문장 길이 및 엔티티 타입 분포 분석

문장 길이와 엔티티 타입 특성은 모델 입력 처리 전략(max_length 설정)과 엔티티 마커 디자인에 직접적인 영향을 미친다. 문장 길이가 지나치게 길면 정보 손실을 최소화하기 위해 max_length를 크게 잡아야 한다. 또한 엔티티 타입 분포를 파악하면 특정 타입 조합에서 주로 발생하는 관계를 모델이 쉽게 학습하도록 타입 정보를 마커에 명시하는 전략을 검토할 수 있다.

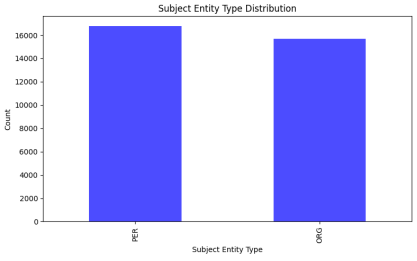
문장 길이 분석



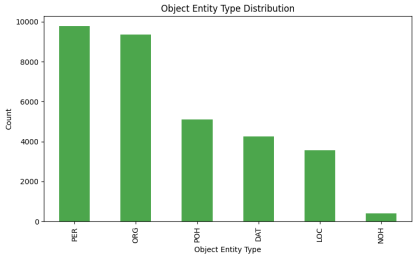
문장 길이 분포 시각화 결과

- 평균 문장 길이 : 100자 내외, 최대 400자 이상의 긴 문장도 존재
- 길이 분포 히스토그램을 통해 중간값(약 87자) 및 상위 25% 구간(약 120자) 확인
- 긴 문장(128자 초과)의 비중이 전체의 약 20%이므로, 이를 반영하여 max_length를 128~160 구간 내에서 설정하는 것이 정보 손실을 최소화할 수 있을 것

엔티티 타입 분포

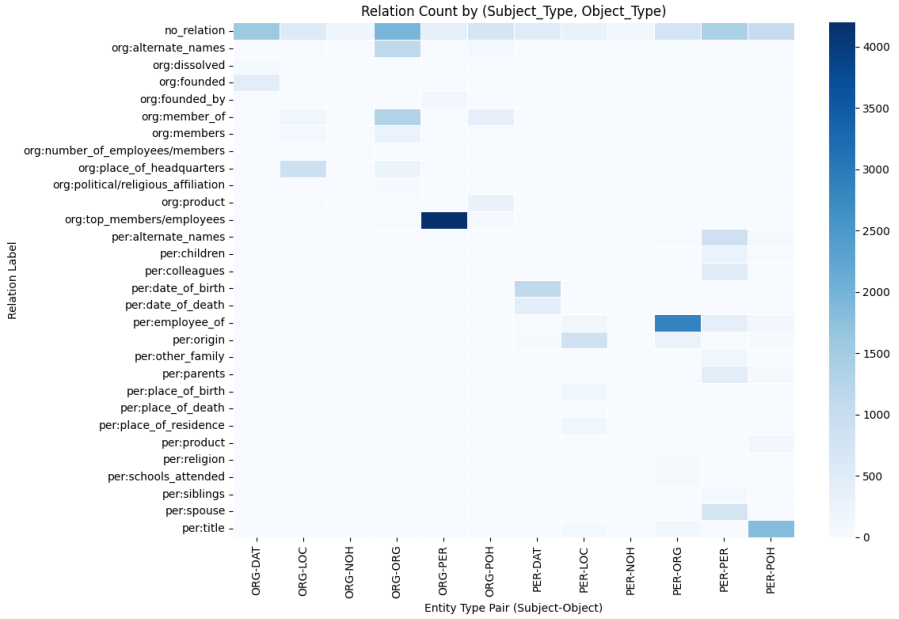


subject entity 타입 분포



object entity 타입 분포

- `subject_entity` 에는 인물(PER), 조직(ORG) 두가지 엔티티만 존재
- `object_entity` 에는 인물(PER), 조직(ORG), LOC(장소), NOH(수량), DAT(날짜), POH(기타)의 6가지 엔티티가 존재



특정 타입 조합에서 발생하는 특정 관계의 수 matrix

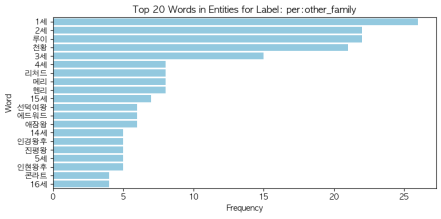
- 특정 관계(org:top_members/employees)는 특정 타입 조합(예: ORG, PER)에서 주로 발생한다는 패턴 파악 가능.
- 이로 인해 엔티티 마커뿐만 아니라 엔티티 타입 정보(E1-ORG, E2-PER)를 마커에 명시하는 전략이 관계 식별에 도움이 될 수 있음을 시사.

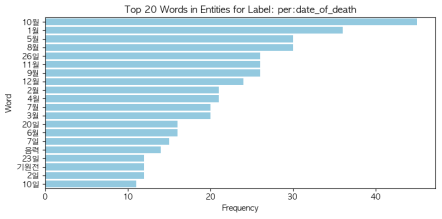
EDA 결과 활용 방안

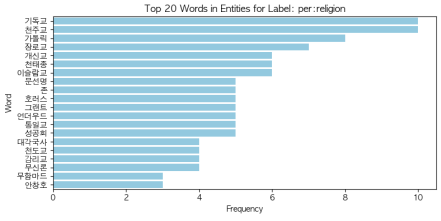
- 문장 길이 분석 결과를 바탕으로 max_length를 128~160으로 설정하는 실험을 고려할 수 있다.
- 엔티티 타입 분포 및 조합 패턴을 파악한 결과, 특정 관계는 특정 타입 패턴에서 집중적으로 나타난다. 이를 활용하기 위해 [E1], [/E1], [E2], [/E2] 같은 엔티티 마커에 타입 정보를 포함(E1-ORG, E2-PER)하여 모델이 해당 엔티티의 성격(조직, 인물 등)을 명확히 알도록 한다. 이로써 모델은 “ORG-PER 조합이면 `org:top_members/employees` 관계가 나올 확률이 높다”와 같은 유형별 패턴을 빠르게 학습할 수 있다.

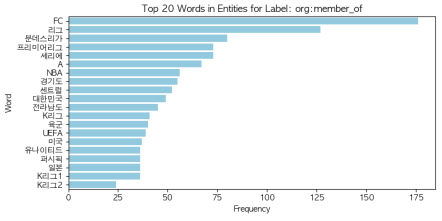
2.3 엔티티 내부 단어(토큰) 분석

본 항목에서는 엔티티 타입이나 마커([E1], [E2] 등)만으로는 충분하지 않을 수 있다는 가정 하에, 엔티티 내부에 등장하는 단어들이 실제 관계 판단에 어떤 영향을 주는지 확인하고자 했다. 특히 희소 클래스(예: per:other_family, per:date_of_death 등)를 중심으로, 엔티티 텍스트가 어떤 핵심 단어를 포함하고 있는지 통계·시각화 기법을 통해 살펴봄으로써, 단순 마커 대신 엔티티 내부 토큰의 중요성을 EDA 차원에서 파악하고자 한다.









- `per:other_family` 에서는 1세, 2세 등의 단어, `date_of_death` 에는 월(month)와 관련된 단어, `per:religion` 에는 종교 관련 단어, `org:member_of` 에는 스포츠팀관련 단어 등이 자주 출현한다.
- 이를 통해서, 특정 그룹에 해당하는 단어들이 포함될 때 모델이 관계 예측을 보다 잘 할 수 있을 것으로 예상할 수 있다.

2.4 EDA 결과를 통한 초기 모델링 방향성 수립

앞선 EDA 결과를 바탕으로 본 과제의 모델링 전략에 대한 초기 방향성을 구체화할 수 있다.

불균형 해소 방안(Focal Loss, Label Smoothing) 고려 근거

EDA 결과, `no_relation` 클래스의 비중이 매우 높고 일부 희소 클래스는 데이터가 극히 적은 수준이다. 이로 인해 모델이 쉽게 `no_relation` 에 치우쳐 모든 관계를 "없음"으로 예측하는 경향이 생길 수 있다.

이렇게 불균형 문제의 심각성이 확인되었으므로, 단순 Cross-Entropy Loss 대신 Focal Loss, Label Smoothing 등을 적용하는 방안을 우선 고려한다. Focal Loss는 희소 클래스를 더 잘 학습하게 하며, Label Smoothing은 모델이 특정 클래스에 과도하게 확신하는 문제를 완화한다. 이를 통해 희소 클래스 식별률을 높이고 `no_relation` 제외 Micro-F1를 개선하고자 한다.

엔티티 타입 반영 및 스팬 표기 필요성 확인

엔티티 타입 분석 결과, 특정 관계가 특정 타입 조합에서 집중적으로 발생하는 패턴을 확인했다. 이를 모델에 직접적으로 반영하기 위해 [E1], [E2] 마커에 엔티티 타입을 표기하는 전략을 도입한다([E1-ORG], [E2-PER] 등). 이는 모델이 "해당 스팬은 조직, 다른 스팬은 인물"임을 명확히 알게 해 관계 분류 시 검색 공간을 줄이고, 학습을 효율화한다.

또한, 엔티티 내부 토큰 정보가 특정 클래스에 유의미하다는 EDA 결과를 기반으로, 단순히 [CLS] 벡터에 의존하지 않고 엔티티 스팬 구간의 hidden state를 풀링하거나, Attention 기반 가중합을 통해 엔티티 관련 토큰에 집중하는 방식을 고려한다.

하이퍼파라미터 설정 시 문장 길이 반영

문장 길이 분석을 바탕으로 max_length를 128~160 범위로 설정하여 정보 손실을 최소화한다.

3. 기본 접근 방식 및 한계

관계 추출 태스크에 적용되는 기본적 접근은 사전학습 언어모델(Pre-trained Language Model, PLM)인 BERT, RoBERTa, 혹은 한국어 특화 모델(KoELECTRA, KLUE-RoBERTa 등)을 기반으로 파인튜닝(fine-tuning)하는 방식이다. 이 방식은 별도의 특수한 구조 없이 [CLS] 토큰 임베딩을 활용하거나, 단순히 문장 전체 임베딩을 이용해 관계를 분류하는 단순하고 직관적인 방법이지만, 불균형한 라벨 분포나 엔티티 정보 활용 측면에서 한계를 가진다.

3.1 기본 BERT/RoBERTa 기반 Fine-tuning

기본 접근법

기본 접근법은 다음과 같이 이루어진다.

1. **사전학습 모델 로드**:

KLUE-RoBERTa, KoELECTRA, KoBERT 등 한국어 처리에 최적화된 모델을 불러온다.

2. **문장 입력 및 토크나이징**:

문장과 함께 subject entity, object entity 위치 정보를 주석으로 확보하되, 초기에는 별도 마커 없이 단순히 "[CLS] 문장 [SEP]" 형태로 모델에 입력한다. 모델의 마지막 층에서 [CLS] 토큰 임베딩을 추출한다.

3. **관계 분류 헤드 추가**:

[CLS] 토큰 임베딩 위에 1~2개의 Fully-Connected(FC) Layer를 올려 30개 클래스 중 하나를 예측하는 분류기를 구성한다.

이러한 접근은 구현이 간단하고, 대규모 사전학습 모델이 이미 문장 이해 능력을 갖추고 있기 때문에 비교적 빠르게 baseline 성능을 확보할 수 있다. 그러나 다음과 같은 한계가 존재한다.

한계점

- [CLS] 벡터만 사용하면 엔티티 주변에 있는 관계 결정적 단서를 충분히 반영하기 어렵다.
- 희소 클래스나 불균형 라벨 문제에 직접 대응하는 기법이 없어, 모델이 `no_relation` 에 과도하게 편향될 가능성이 크다.

- 엔티티 타입 정보나 스팬 정보가 명시적으로 반영되지 않아, 특정 관계가 특정 엔티티 타입 조합에서 주로 발생하는 패턴을 모델이 효율적으로 학습하기 어렵다.

3.2 Entity Marker 도입 배경 및 기대 효과

EDA 결과, 특정 관계가 특정 타입 조합에서 빈번하고, 희소 클래스 식별에는 엔티티 정보에 대한 집중이 중요함을 확인했다. 기본 파인튜닝 방식에서는 엔티티의 위치나 타입 정보가 모델에 직접적으로 주어지지 않는다. 따라서 [E1], [E2] 같은 엔티티 마커를 문장 내 엔티티 앞뒤에 삽입함으로써 모델이 "이 부분이 바로 엔티티다"를 명확히 알 수 있게 하는 전략을 도입할 수 있다.

엔티티 마커 도입 기대 효과:

- 모델이 문장 내 수많은 토큰 중 어떤 구간이 중요한 엔티티인지 명확히 파악 가능.
- 엔티티 마커로 엔티티 스팬을 표시하면, 엔티티 관련 토큰 임베딩을 별도 풀링하거나 Attention할 수 있어 엔티티 중심의 표현을 강화한다.
- 나아가 [E1-ORG], [E2-PER]처럼 엔티티 타입까지 마커에 반영하면, 모델은 타입 정보를 즉각적으로 인식해 특정 관계가 주로 ORG-PER 조합에서 나오는 패턴 등을 쉽게 학습할 수 있다.

EDA를 통해 도출된 "엔티티 중심 정보 반영 필요성"은 엔티티 마커 도입의 합리적 근거가 된다. 단순 [CLS] 기반 접근에 엔티티 마커를 추가함으로써, 모델은 엔티티 주변 정보에 더 예민하게 반응할 수 있고, 이는 희소 클래스 식별력 향상 및 불균형 문제 개선에도 간접적으로 기여할 수 있다.

3.3 불균형 라벨 분포에 따른 문제점

EDA에서 가장 크게 부각된 문제는 라벨 불균형이며, 특히 `no_relation` 클래스가 과도하게 많은 점이다. 기본 BERT/RoBERTa 파인튜닝만으로는 불균형 대응이 충분치 않다. 이 경우 모델이 쉽게 `no_relation`만 예측하는 편향이 생기며, 희소 클래스를 거의 식별하지 못하는 문제가 발생한다.

문제점

- 단순 Cross-Entropy Loss와 균일한 학습 전략은 다수 클래스에 과도한 가중치를 실어 희소 관계를 놓치게 만든다.
- EDA 결과 희소 클래스를 인식하는 것이 모델 개선의 핵심이라는 점을 고려할 때, 기본 접근법은 이 문제를 직접적으로 해결할 방안이 없다.

EDA 반영 방안

- 이후 단계에서 Focal Loss, Label Smoothing 등 불균형 문제를 직접 해결하는 기법을 도입할 필요성이 있다.

4. 데이터 불균형 문제 해결을 위한 전략

4.1 Focal Loss 도입 및 이유

배경 및 필요성

- EDA 결과, 라벨 분포가 극도로 불균형하여 `no_relation` 클래스가 과도하게 많다는 점을 발견했다.
- 모델이 `no_relation` 만 예측해도 정확도가 일정 수준 유지되는 편향 발생 → 희소 클래스에 대한 Recall이 급격히 떨어짐.
- 일반 Cross Entropy Loss로는 이러한 문제를 직접적으로 해결하기 어려우며, 희소 클래스가 무시되는 경향이 심해진다. 희소 클래스의 손실 기여도가 작아 모델이 해당 클래스를 무시해도 전체 손실을 크게 낮출 수 있기 때문이다.

Focal Loss 원리

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

- **Focal Loss**는 Lin et al. (2017)에서 제안된 손실 함수로, 쉽거나 이미 맞출 수 있는 샘플에 대해서는 손실 기여도를 줄이고, 어려운(오답) 샘플일수록 가중치를 높여 학습하게 유도한다.
- $(1 - p_t)^{\gamma}$ 항을 도입해 모델이 틀리거나 불확실한 샘플(특히 소수 클래스)에 집중하도록 한다. α를 통해 특정 클래스에 추가 가중치 부여도 가능.
    - 모델이 쉽게 정답을 맞출 수 있는 샘플(다수 클래스)에 대해서는 $(1 - p_t)^{\gamma}$ 항이 매우 작게 작동해 손실이 줄어들고, 반대로 잘 틀리는(확률값이 낮은) 희소 클래스 샘플에는 큰 손실이 부여된다.

4.2 Label Smoothing 적용 방안

Label Smoothing

- 개념 : 모델이 특정 클래스에 확률 1.0(100% 확신)을 쉽게 부여하지 못하도록, 라벨 벡터를 소량 평탄화하는 기법. → 각 클래스 확률을 조금씩 분산시킴.
  - ex) one-hot vector [0,0,1,0] 대신 0.1 수준의 smoothing을 적용해 [0.03, 0.03, 0.91, 0.03] 형태로 바꾼 후 Cross Entropy 계산
- 효과:
  - Overconfidence(과도한 확률 할당)로 인한 오버피팅 완화
  - 희소 클래스를 전혀 예측하지 않거나, 반대로 `no_relation` 으로 모든 것을 예측하는 편향을 다소 줄여줌
  - Focal Loss와 함께 적용 시, 어려운 샘플(희소 클래스)에 대한 가중을 높이면서도, 모델의 극단적 확률 분배를 완화할 수 있다.

적용방안

- Label Smoothing 정도 (예: 0.05 ~ 0.2, 일반적으로 사용되는 값의 범위)를 직접 validation set에 테스트해보고, 성능 확인하는 방법
- 라벨 분포 불균형 정도를 계량화하여 smoothing 정도를 결정하는 방법
  - 라벨 분포의 엔트로피나 지니계수를 활용하는 방법 (함수 구현)
    1. 라벨별 빈도를 입력받아 전체 엔트로피와 불균형 정도를 계산.
    2. 엔트로피가 낮을수록(즉 특정 클래스 편향이 심할수록) label smoothing을 크게 설정.
    3. 엔트로피 기반으로 smoothing을 0.05~0.2 범위 내에서 동적으로 결정.

```
import math

def compute_label_smoothing_factor(label_counts, base_min=0.05, base_max=0.2):
    """
    라벨 빈도(label_counts)를 받아 label smoothing 파라미터를 동적으로 결정.
    - label_counts: {label: count} 형태의 dict
    - base_min: 최소 smoothing 값
    - base_max: 최대 smoothing 값

    1) 전체 라벨 수 N = sum(count)
    2) 각 라벨 비율 p_i = count_i / N
    3) 엔트로피 H = -sum(p_i * log(p_i))
    4) 최대 엔트로피 = log(K) (K는 클래스 수)
    5) 정규화 엔트로피 = H / log(K)
       - 정규화 엔트로피가 1에 가까울수록 분포가 고르게 퍼져 있고, 0에 가까울수록 불균형 심함.
    6) smoothing = base_max - (base_max - base_min)*정규화엔트로피
       - 정규화 엔트로피가 낮을수록(불균형 클수록) smoothing값은 base_max에 가까워짐.
    """
    counts = list(label_counts.values())
    total = sum(counts)
    p = [c/total for c in counts]

    # 엔트로피 계산
    entropy = 0.0
    for pi in p:
        if pi > 0:
            entropy -= pi * math.log(pi)

    K = len(counts)
    max_entropy = math.log(K)
    normalized_entropy = entropy / max_entropy
```

```
    # normalized_entropy가 1일 때 smoothing = base_min,
    # normalized_entropy가 0일 때 smoothing = base_max로 설정
    smoothing = base_max - (base_max - base_min) * normalized_entropy
    return smoothing
```

위 함수는 분포가 균등할수록 smoothing을 낮게, 불균형할수록 smoothing을 높게 잡는 방향으로 구현되어 있음.

5. 엔티티 스팬 기반 개선 방안

5.1 엔티티 마커 단순 삽입 방식의 한계

배경

- 앞서 3.2에서 언급한 대로, 엔티티 마커([E1], [E2] 등)을 문장에 삽입하면 모델이 엔티티 위치와 타입 정보를 좀 더 명확히 알 수 있어 관계 추출 성능이 개선됨
- 하지만 단순 마커만 추가하는 것으로는 엔티티 주변 세밀한 정보를 충분히 확인하지 못할 가능성이 남아 있다.

한계 요인

- [CLS]에 여전히 의존 : 일반적인 파인튜닝 구조에서는 여전히 문장 전체 정보를 [CLS] 임베딩(혹은 문장 마지막 토큰)을 통해 요약 후, 이를 분류기에 전달한다. [E1], [E2] 마커가 엔티티 위치와 타입을 알리기는 하지만, 최종적으로 [CLS] 한 점에 집중된 정보에서 엔티티 주변 단서가 희석될 위험이 크다.
- **엔티티 내부 표현 부족 :** 마커는 엔티티 '시작'과 '끝'을 알려주지만, 엔티티 구간 내 각 토큰(예: 인물, 장소 등)에 대한 세밀한 정보를 별도로 추출하지 않으면, 긴 엔티티나 복합 엔티티 상황에서 정보 손실이 발생할 수 있다.
  - 엔티티가 "비틀즈"처럼 간단한 단어 하나로 구성된다면 마커만으로도 어느 정도 효과가 있을 수 있다.
  - 그러나 엔티티가 "대한민국 서울특별시 용산구"처럼 복합·장문의 표현일 경우, 마커는 스팬의 시작과 끝만 표시할 뿐, 스팬 내 각 단어가 중요한지 여부를 구체적으로 반영하지 못한다.

	id	sentence	subject_entity	object_entity	label	source
35	35	로버트 조지 (바비) 실(Robert George (Bobby)(Bobby) Sea...	{'word': '흑표당', 'start_idx': 99, 'end_idx': 10...	{'word': '휴이 뉴튼', 'start_idx': 89, 'end_idx': ...	org:founded_by	wikipedia
234	234	커널 샌더스(Colonel Sanders)로 널리 알려져 있는 튈랜드 데이비드 샌더...	{'word': 'KFC', 'start_idx': 78, 'end_idx': 80...	{'word': '커널 샌더스', 'start_idx': 0, 'end_idx': ...	org:founded_by	wikipedia
459	459	1992년에는 에두아르드 리모노프와 알렉산드르 두긴을 중심으로 국민볼셰비키당이 창당...	{'word': '국민볼셰비키당', 'start_idx': 35, 'end_idx':...	{'word': '에두아르드 리모노프', 'start_idx': 8, 'end_id...	org:founded_by	wikipedia
616	616	함연지 씨는 오뚜기 창업주인 함태호 명예회장의 손녀로, 뮤지컬 배우로 활동하고 있다.	{'word': '오뚜기', 'start_idx': 7, 'end_idx': 9, ...	{'word': '함태호', 'start_idx': 16, 'end_idx': 18...	org:founded_by	wikitree
679	679	마이크로 센터는 1979년 35,000달러의 투자금과 함께 라디오섁의 판매원이었...	{'word': '마이크로 센터', 'start_idx': 0, 'end_idx':...	{'word': '빌 배인', 'start_idx': 55, 'end_idx': 5...	org:founded_by	wikipedia
...	...	...	...	...	...	...
30408	30408	삼성그룹 분가 과정에서 1995년 삼성그룹 창업주였던 고 이병철 회장의 2남인 고 ...	{'word': '삼성그룹', 'start_idx': 19, 'end_idx': 2...	{'word': '이병철', 'start_idx': 32, 'end_idx': 34...	org:founded_by	wikipedia
30474	30474	현대그룹의 사업확장이 거둔 성공은 현대 그룹에 탄탄한 자금력을 가져다 주었음은 물론...	{'word': '현대 그룹', 'start_idx': 19, 'end_idx': ...	{'word': '정주영', 'start_idx': 96, 'end_idx': 98...	org:founded_by	wikipedia
30739	30739	2013년 11월, 안철수는 신당 창당을 목표로 국민과 함께하는 새정치 추진 위원회(...	{'word': '새정치 추진위원회', 'start_idx': 36, 'end_id...	{'word': '안철수', 'start_idx': 11, 'end_idx': 13...	org:founded_by	wikipedia
31456	31456	카카오 창업자인 김범수 카카오 이사회 의장이 코로나19 극복을 위해 20억원 상당의...	{'word': '카카오', 'start_idx': 0, 'end_idx': 2, ...	{'word': '김범수', 'start_idx': 9, 'end_idx': 11,...	org:founded_by	wikitree
31845	31845	피자헛은 1958년 미국 캔자스주에서 프랭크와 댄 카이니 형제에 의해 작은 점포를 ...	{'word': '피자헛', 'start_idx': 0, 'end_idx': 2, ...	{'word': '프랭크와 댄 카이니 형제', 'start_idx': 21, 'en...	org:founded_by	wikipedia

	id	sentence	subject_entity	object_entity	label	source
91	91	백한성(白漢成, 水原白人, 1899년 6월 15일 조선 충청도 공주 출생 ~ 197...	{'word': '백한성', 'start_idx': 0, 'end_idx': 2, ...	{'word': '조선 충청도 공주', 'start_idx': 28, 'end_id...	per:place_of_birth	wikipedia
774	774	코치시 출탄(1952년 5월 30일 ~ 2016년 11월 6일)은 형가리 태생의 피...	{'word': '코치시 출탄', 'start_idx': 0, 'end_idx': ...	{'word': '형가리', 'start_idx': 37, 'end_idx': 39...	per:place_of_birth	wikipedia
1105	1105	윤호중은 경기도 장단군 출신으로, 배재고등보통학교에서 미술 교사였던 조각가 김복진에...	{'word': '윤호중', 'start_idx': 0, 'end_idx': 2, ...	{'word': '경기도', 'start_idx': 5, 'end_idx': 7, ...	per:place_of_birth	wikipedia
1110	1110	토미 메이컴은 북아일랜드 카운티 아마의 키디에서 1932년에 태어났다.	{'word': '토미 메이컴', 'start_idx': 0, 'end_idx': ...	{'word': '북아일랜드 카운티 아마', 'start_idx': 8, 'end_...	per:place_of_birth	wikipedia
1300	1300	빌리 오르반은 독일 카이저슬라우테른에서 헝가리인 어버지와 폴란드인 어머니 사이에서 ...	{'word': '빌리 오르반', 'start_idx': 0, 'end_idx': ...	{'word': '카이저슬라우테른', 'start_idx': 11, 'end_idx...	per:place_of_birth	wikipedia
...	...	...	...	...	...	...
31936	31936	본관은 상주(尙州)이고 호(號)는 산남(山南)이며 영어 이름은 토마스 도(Thoma...	{'word': '토마스 도', 'start_idx': 35, 'end_idx': ...	{'word': '상주(尙州)', 'start_idx': 4, 'end_idx': ...	per:place_of_birth	wikipedia
32025	32025	루자인 알하틀릴은 2015년 12월에 사우디아라비아 역사상 최초로 여성이 선거권을 ...	{'word': '루자인 알하틀릴', 'start_idx': 0, 'end_idx':...	{'word': '사우디아라비아', 'start_idx': 21, 'end_idx':...	per:place_of_birth	wikipedia
32162	32162	1946년 평안북도 진남포 (현재의 남포특별시)라고 알려져 있지만 공식적으로 평안남...	{'word': '하용조', 'start_idx': 67, 'end_idx': 69...	{'word': '평안남도 강서군', 'start_idx': 43, 'end_idx...	per:place_of_birth	wikipedia
32204	32204	툰루 압둘 라만은 1903년 2월 8일, 말레이시아의 알로르스타르에서 태어났다.	{'word': '툰루 압둘 라만', 'start_idx': 0, 'end_idx':...	{'word': '말레이시아', 'start_idx': 23, 'end_idx': ...	per:place_of_birth	wikipedia
32404	32404	소냐 엘리지는 1965년 뉴질랜드의 오클랜드에서 달마티아에서 이민을 온 크로아티아계...	{'word': '소냐 엘리치', 'start_idx': 0, 'end_idx': ...	{'word': '뉴질랜드', 'start_idx': 14, 'end_idx': 1...	per:place_of_birth	wikipedia

  - 희소 클래스 중에는 특정 세부 토큰이 결정적 역할을 하는 관계(예: org:founded_by에서 창립자 이름, per:place of birth에서 특정 지역명)가 많으므로, 단순 마커 삽입만으로는 이 정보를 풍부하게 살리지 못할 것이다.

EDA로부터의 인사이트 적용

- 엔티티 타입 조합(ORG-PER 등)에 따른 특정 관계가 추출된다는 패턴이 뚜렷하게 나타날 경우, 단순 엔티티 마커 뿐만 아니라 타입도 포함시키는게 유용할 수 있을 것이다.
- 엔티티 내부 토큰 정보도 중요한 단서가 됨을 EDA에서 확인했다.
- 단순 마커만으로는 "이 엔티티가 ORG라는 것"까진 쉽게 표현 가능하나, 스팬 내 토큰이 주는 세부 정보를 효과적으로 반영하기에는 부족하다.

결론

- 스팬 구간 자체를 별도의 임베딩으로 추출해 "엔티티가 포함하는 단어들"을 통합적으로 고려해야, 관계에 결정적 단서를 놓치지 않는다.

- 이를 해결하기 위해 5.2에서 소개할 엔티티 스팬 풀링 기법을 도입한다.

5.2 엔티티 스팬 풀링(Entity Span Pooling)의 구현 및 기대 효과

개념

- 엔티티 스팬 풀링(Entity Span Pooling): 엔티티 마커를 통해 `[E1] … [/E1]`, `[E2] … [/E2]` 구간을 명시했음에도, 단순히 [CLS] 임베딩만 사용하는 구조로는 엔티티 내부 토큰들의 가치를 고르게 반영하기 어렵다. 이를 보완하기 위해 엔티티 스팬 내부 임베딩을 직접 풀링(aggregating)하여 하나의 벡터로 만들고, 그 벡터를 최종 분류에 활용하는 방식을 **엔티티 스팬 풀링**이라 부른다.
  - 스팬 : `[E1]부터 [/E1] 사이`(또는 `[E2]부터 [/E2] 사이`)의 토큰들을 한 덩어리(스팬)로 묶는다.
  - 풀링 : 해당 스팬에 속하는 모든 토큰의 hidden state(Transformer 마지막 레이어의 임베딩)를 모아 평균(Mean), 최대(Max), 혹은 Attention 기반 가중합으로 단일 벡터를 생성한다.
- 이를 통해 엔티티 스팬의 세부 토큰 정보를 집약해 모델이 관계 판별에 필요한 엔티티 자체의 표현을 명확히 얻을 수 있다.

구현 방식

- 평균 풀링(Mean Pooling) : 엔티티 스팬 내 모든 토큰 임베딩을 단순 평균하여 하나의 벡터로 생성
  - 구현이 간단하고 계산 비용이 적으나 엔티티가 긴 경우 단어마다 중요도가 다를 수 있다는 점이 반영되지 않음
- 최대 풀링(Max Pooling) : 스팬 내 각 차원별 임베딩의 최댓값을 취해 하나의 벡터로 생성
  - 가장 두드러진 특징을 뽑아낼 수 있으나, 평균 풀링과 마찬가지로 토큰별 중요도를 차별화하기 어려움
- Attention Pooling : 각 토큰 임베딩에 대해 학습 가능한 가중치를 할당해, 중요한 토큰에 더 큰 비중을 주어 가중합
  - 토큰별 중요도를 반영할 수 있어, 엔티티 내 핵심단어에 집중할 수 있다.
- 구현 코드

```
class AttentionPooling(nn.Module):
    def __init__(self, hidden_size: int):
        super().__init__()
        self.w = nn.Parameter(torch.randn(hidden_size))

    def forward(self, span_hidden_states: torch.Tensor) -> torch.Tensor:
        # span_hidden_states: [span_length, hidden_size]
        # return: [hidden_size]

        # attention score : dot product between span_hidden_states(token vectors) and w
        # [span_length, hidden_size] · [hidden_size] -> [span_length]
        scores = torch.matmul(span_hidden_states, self.w)

        # softmax to get attention weights
        attn_weights = torch.softmax(scores, dim=0) # [span_length]

        # weighted sum of span_hidden_states
        # (span_length, ) x (span_length, hidden_size) -> sum after broadcasting
        pooled = torch.sum(attn_weights.unsqueeze(-1) * span_hidden_states, dim=0)
        return pooled

class RelationClassifier(nn.Module):
    def __init__(self, model_name: str, num_labels: int,
                 dropout: float = 0.1, tokenizer_len: int = None,
                 use_span_pooling: bool = False,
                 use_attention_pooling: bool = False):
        super().__init__()

        ...

        if self.use_span_pooling:
```

```
            # due to span pooling, hidden size is doubled : [E1_vec; E2_vec] concat -> hid
den_size * 2
            self.classifier = nn.Linear(self.model.config.hidden_size * 2, num_labels)
        else:
            self.classifier = nn.Linear(self.model.config.hidden_size, num_labels)

        if self.use_attention_pooling:
            self.e1_attention_pooling = AttentionPooling(self.model.config.hidden_size)
            self.e2_attention_pooling = AttentionPooling(self.model.config.hidden_size)

           ...

    def forward(self, input_ids: torch.Tensor, attention_mask: torch.Tensor,
                e1_start_idx: torch.Tensor = None, e1_end_idx: torch.Tensor = None,
                e2_start_idx: torch.Tensor = None, e2_end_idx: torch.Tensor = None) -> tor
ch.Tensor:
        outputs = self.model(input_ids=input_ids, attention_mask=attention_mask)
        hidden_states = outputs.last_hidden_state # [batch_size, seq_len, hidden_size]

        if self.use_span_pooling:
            # span pooling
            # E1 pooling
            e1_pooled = []
            for i, hs in enumerate(hidden_states): # hs : [seq_len, hidden_size]
                start = e1_start_idx[i].item()
                end = e1_end_idx[i].item()
                # to handle case where area is not correct
                if end >= start:
                    span_hs = hs[start:end+1] # [span_length, hidden_size]
                    if not self.use_attention_pooling:
                        span_vec = span_hs.mean(dim=0)
                    else:
                        span_vec = self.e1_attention_pooling(span_hs)
                else:
                    # fallback: CLS or avg but here use CLS
                    span_vec = hs[0]
                e1_pooled.append(span_vec)
            e1_pooled = torch.stack(e1_pooled, dim=0) # [batch_size, hidden_size]

            # E2 pooling
            e2_pooled = []
            for i, hs in enumerate(hidden_states):
                start = e2_start_idx[i].item()
                end = e2_end_idx[i].item()
                # to handle case where area is not correct
                if end >= start:
                    span_hs = hs[start:end+1] # [span_length, hidden_size]
                    if not self.use_attention_pooling:
                        span_vec = span_hs.mean(dim=0)
                    else:
                        span_vec = self.e2_attention_pooling(span_hs)
                else:
                    # fallback: CLS or avg but here use CLS
                    span_vec = hs[0]
                e2_pooled.append(span_vec)
            e2_pooled = torch.stack(e2_pooled, dim=0) # [batch_size, hidden_size]

            # concat
```

```
            concat_vec = torch.cat([e1_pooled, e2_pooled], dim=-1) # [batch_size, hidden_s
ize * 2]
            concat_vec = self.dropout(concat_vec)
            logits = self.classifier(concat_vec) # [batch_size, num_labels]

            ...

        return logits
```

관계 판단 과정

- E1 스팬 벡터와 E2 스팬 벡터를 각각 구한 뒤, 두 벡터를 단순 concat하거나, element-wise 연산 등을 통해 통합 벡터 생성.
- 최종적으로 분류 헤드(FC layer 등)에 입력해 30개 관계 중 하나를 예측한다.
- 이 과정에서, 엔티티 타입 정보나 추가적인 마커 임베딩도 함께 concat해 모델이 "이 벡터는 ORG-PER 조합, 이 단어들은 엔티티 본문"임을 명확히 인지하도록 할 수 있음.

기대 효과

- 엔티티 중심 정보 강화: 단순 [CLS]에 분산되었던 엔티티 정보를 별도의 스팬 임베딩으로 추출함으로써, 모델이 "두 엔티티 각각이 어떤 내부 토큰 정보를 갖고 있는지" 명확히 파악 가능.
- 긴 문장에도 대응 가능 : 문장이 길어지면 [CLS] 임베딩에 더 다양한 정보가 압축되어 엔티티 부분이 희석될 위험이 높다. 스팬 풀링은 엔티티 구간만을 떼어 임베딩을 생성하므로, 긴 문장 여부와 무관하게 엔티티 중심 표현을 잃지 않는다.
- 희소 클래스 식별 개선: 엔티티가 길거나 복잡한 표현(예: "대한민국 서울특별시" 같은 지명)을 가질 때, 평균/최대/Attention 풀링이 엔티티 내부 단서를 더 잘 포착해 희소 클래스 예측에 기여.

5.3 Attention 기반 Span Pooling으로의 확장

배경 및 필요성

- 엔티티 스팬 내 토큰들이 모두 동일한 중요도를 갖는 것은 아니므로, 평균/최대 풀링은 한계를 가진다.
- 예: "대한민국 서울특별시 용산구" 중 "서울특별시"가 관계 결정에 더 중요한 경우, 이를 모델이 학습하도록 가중치를 부여해야 함.

Attention Pooling 원리

$\alpha_i = \frac{\exp(h_i \cdot w)}{\sum_j \exp(h_j \cdot w)}$

- **학습 가능한 파라미터**(예: 벡터 w)를 사용해 스팬 내 각 토큰 임베딩 h_i와 점곱($h_i \cdot w$)을 구해 스칼라 점수를 계산한 뒤, softmax로 정규화해 가중치로 사용.
- 최종 스팬 임베딩은 $\sum_i \alpha_i h_i$형태의 가중합으로 얻는다.
- 이를 통해 토큰별 중요도를 모델이 학습으로 결정하게 하여, 실제 관계판별에 더 중요한 단어나 서브토큰에 집중하게 만든다.

세부 구현 내용 (코드는 5.2 참고)

- **별도 Attention 모듈**: 엔티티 스팬 내 토큰 임베딩을 입력으로 받아, 점곱(혹은 추가 feed-forward) 후 소프트맥스 가중치로 가중합을 구하는 미니 모듈.
- **E1, E2 각각 적용**: E1 스팬 벡터, E2 스팬 벡터를 각각 Attention Pooling으로 얻은 뒤, concat 등을 통해 최종 벡터 생성.

6. Ensemble 기법 도입

6.1 다양한 한국어 PLM 병렬 학습

다양한 모델 간 특성 차이 활용

- KoELECTRA, KLUE-RoBERTa, KoBERT 등 한국어 PLM마다 사전학습 데이터, 토크나이저, 모델 구조 (Generator/Discriminator, BPE vs. SentencePiece 등)가 달라 미묘한 예측 편향을 갖는다.

- 예컨대 KoELECTRA는 discriminator 방식으로 pre-training해 문장 이해력이 특정 도메인에서 강점을 보일 수 있고, KLUE-RoBERTa는 KLUE 벤치마크에 최적화된 특성이 있을 수 있다.
- 각 모델이 독립적인 파인튜닝 과정을 거쳐 최적화된 가중치를 얻도록 한다.

모델별 커스텀 설정

- Focal Loss(γ, α)나 Label Smoothing, LR 스케줄러(linear/cosine/polynomial) 등을 모델별로 다르게 적용해볼 수 있다.
- 어떤 모델은 ($\gamma = 1.0, \alpha = 0.25$), 다른 모델은 ($\gamma = 2.0, \alpha = 0.5$) 등 여러 조합을 탐색하며 validation set 성능을 모니터링한다.
- 엔티티 마커, 스팬 풀링 등도 모델별로 조금씩 다른 하이퍼파라미터나 풀링 방식을 시도해, 독립적 성능 극대화를 도모한다.

6.2 Voting / Averaging 방식의 앙상블 구현

Voting 앙상블

- Hard Voting : 각 모델이 예측한 관계 라벨을 표로 취합해, 최빈값(가장 많이 나온 라벨)을 최종 예측으로 결정한다.
- Soft Voting : 각 모델이 산출한 관계 클래스별 확률 분포를 평균 내거나 합산 후 argmax를 구한다. 희소 클래스 확률이 여러 모델에서 조금씩만 높게 나와도 합산으로는 상당히 커질 수 있어 유리하다.

Logit Averaging

- 모델별 출력 로짓(logits)을 합산 or 평균한 뒤 softmax를 취해 최종 확률 분포로 결정.
- soft voting과 유사하지만, 확률 공간에서 더 정교히 결합할 수 있고, 모델별 가중치를 부여해 특정 모델이 더 높은 영향력을 갖게 조정할 수도 있다.

Weight 조절

- 모델별 성능(micro f1 등)에 따라, 좀 더 성능이 좋은 모델의 가중치를 크게, 미흡한 모델의 가중치를 작게 설정하는 방법(Weighted Soft Voting)
  - $p_{\text{final}} = w_1 \cdot p_1 + w_2 \cdot p_2 + \ldots + w_n \cdot p_n, \ \sum w_i = 1.$
- 이런 방식으로 희소 클래스를 더 잘 맞추는 모델에 가중치를 높여, `no_relation` 제외 micro-f1 개선을 도모한다.

6.3 앙상블 기법 적용 시 이점 및 고려 사항

이점

- 편향 상쇄 효과 : 모델 A가 희소 클래스를 놓칠 때 모델 B가 잡아줄 수 있고, 모델 C가 `no_relation` 위주로 편향되더라도 다른 모델이 균형을 맞출 수 있음
- 희소 클래스 Recall 개선: 특정 모델 하나만으론 포착이 어려웠던 희소 클래스 사례를, 여러 모델이 조금씩 확률을 부여해 결과적으로 확률합이 큰 값이 될 수 있다.(Soft Voting)

고려 사항

- 추론 비용 : n개의 모델 각각 추론해야 하므로, 실시간 혹은 대규모 배치 처리에서 시간이 n배로 증가한다.
- 학습 자원 증가 : 각 모델 파인튜닝에 필요한 GPU, 데이터 처리 시간이 증가
- 모델 관리 복잡도 : 다양한 모델 체크포인트를 모두 관리하고, 통합 추론 코드를 작성해야 한다.

적용 전략

- 희소 클래스 성능 모니터링 : `no_relation` 제외 Micro-F1, 희소 클래스별 F1 등 세밀한 지표를 확인하여, 어떤 모델 조합이 시너지 효과가 가장 큰지 탐색한다.
- Weight Tuning : 모델별 validation 데이터에서의 성능에 비례해 앙상블 가중치를 설정하거나, Grid Search 등을 이용해 weight를 자동 튜닝해볼 수 있다.

7. 실험 설정

7.1 실험 환경

개발 및 실행 환경

- Google Colab을 주 플랫폼으로 사용하였으며, 전체적인 코드 구성과 편집은 Cursor IDE를 통해 진행
- 코드 형상관리는 GitHub을 활용해 버전 관리를 수행

Python 패키지 관리 (Poetry)

- Pyenv로 Python `3.10.13`을 설치 후, Poetry를 이용해 의존성을 관리
- 사용 패키지 리스트 (pyproject.toml)
  - 주요 패키지 버전
    - `torch` (2.5.1)
    - `transformers` (4.47.1)
    - `scikit-learn` (1.6.0), `numpy` (2.2.0), `pandas` (2.2.3)

```
[tool.poetry]
name = "upstage-assignment"
version = "0.1.0"
description = ""
authors = ["sora kim <icon_o_clast@naver.com>"]
readme = "README.md"

[tool.poetry.dependencies]
python = "^3.10"
requests = "^2.32.3"
pandas = "^2.2.3"
tqdm = "^4.67.1"
torch = "^2.5.1"
transformers = "^4.47.1"
scikit-learn = "^1.6.0"
numpy = "^2.2.0"
matplotlib = "^3.10.0"
seaborn = "^0.13.2"
optuna = "^4.1.0"
optuna-dashboard = "^0.17.0"
accelerate = "^1.2.1"
safetensors = "^0.4.5"


[tool.poetry.group.dev.dependencies]
ipykernel = "^6.29.5"

[build-system]
requires = ["poetry-core"]
build-backend = "poetry.core.masonry.api"
```

- 사용된 모델 리스트
  - KoELECTRA(monologg/koelectra-base-v3-discriminator)
  - KLUE-RoBERTa(klue/roberta-base or klue/roberta-large)
  - DeBERTa(team-lucid/deberta-v3-base-korean)

7.2 하이퍼파라미터 탐색 범위 및 결정 과정

EDA를 통해 불균형 문제와 문장 길이 분포, 희소 클래스 존재 등을 확인한 결과에 기반해, 다음과 같은 하이퍼파라미터를 집중 탐색했다.

max_length

- 문장 길이가 평균 100자 내외, 400자 이상도 관찰됨
- [128, 160, 200] 값을 실험

batch_size

- [16, 24, 32] 값을 실험

learning_rate

- [2e-5, 3e-5, 5e-5] 값을 실험
- 검증 성능에 따라 3e-5를 최종 채택한 사례가 많았음

learning rate scheduler

- Linear vs. Cosine vs. Polynomial vs. constant로 비교
- Linear는 워밍업 후 linearly decay하는 전형적 스케줄, Cosine은 후반 학습에서 조금 더 천천히 학습률을 낮추는 형태. Polynomial은 학습 과정에서 학습률(learning rate)을 다항식(polynomial) 형태로 점진적으로 감소시키는 스케줄링 방식

Focal Loss 파라미터 (γ, α)

- $\gamma \in \{0.5, 1.0, 1.5, 2.0\}, \alpha \in \{0.25, 0.5, 0.75, 1.0\}$등 그리드 탐색
- validation set에서 `no_relation` 제외 Micro-F1을 지표로 삼아 최종 결정

Label Smoothing

- 0.05 ~ 0.2 범위가 일반적으로 많이 사용되는 범위
- 라벨 분포 불균형 정도를 계량화하여 smoothing 정도를 결정하는 방법을 이용 (4.2 참고)

  → 최종 0.09로 결정

엔티티 마커와 스팬 풀링 옵션

- `[E1], [E2]` 단순 엔티티 마커 vs. `[E1-ORG], [E2-PER]` 타입 포함 엔티티 마커
- Mean Pooling vs. Attention Pooling

결정 과정

1. 특정 모델을 baseline(CE Loss, linear 스케줄러, max_length=128)로 1epoch 학습 후 validation 성능 기록
2. max_length, lr, lr scheduler, batch_size, label_smoothing, focal loss 파라미터를 바탕으로 `Optuna` 라이브러리를 이용하여 각 조합의 validation 성능 확인
3. 스펙이 확정된 뒤, 다른 모델에도 유사 범위를 적용해 각자 best hyperparam 찾음
4. 최종 적합된 모델들을 평가 분석

7.3 학습/검증/테스트 데이터 분리 전략

- 원본 `train.csv` 를 train : valid = 8 : 2로 분할
- Stratified Split : 라벨 불균형이 심하므로, scikit-learn의 `train_test_split(..., stratify=labels)` 옵션을 활용해 각 라벨이 train/valid에 유사 비율로 분포하도록 함

```
train, valid = train_test_split(train_data, test_size=0.2, stratify=train_data["label"])

train.reset_index(drop=True, inplace=True)
valid.reset_index(drop=True, inplace=True)
```

```
train.to_csv("data/train_data.csv", index=False)
valid.to_csv("data/valid_data.csv", index=False)
```

8. 실험 결과 및 분석

8.1 RoBERTa + Entity Marker + Basic CE Loss (baseline) 결과

실험 설정

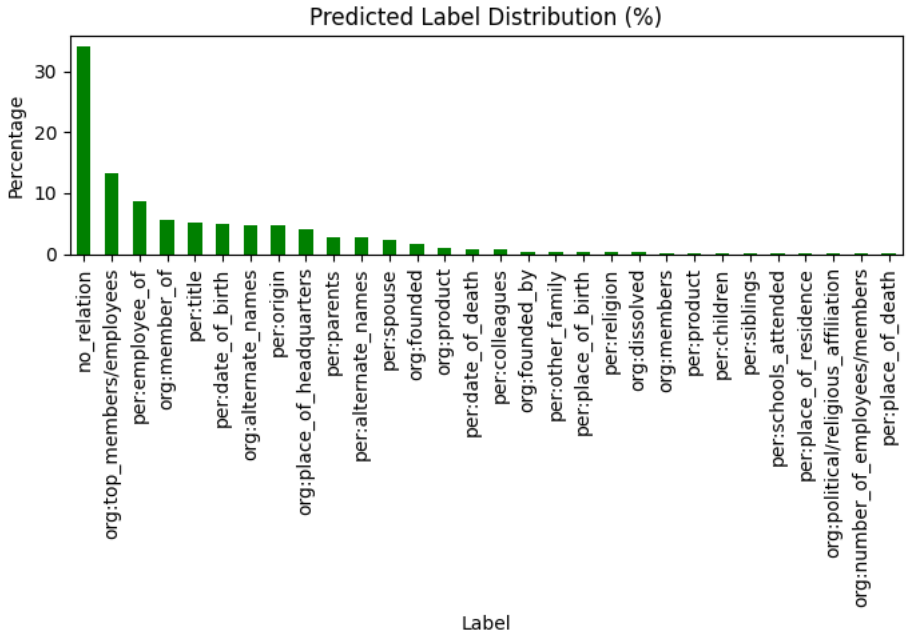
- 모델 : klue/roberta-base
- 손실함수 : Cross Entropy
- 기타 하이퍼파라미터 : max_length=128, batch_size=16, learning_rate=3e-5, linear scheduler
- 엔티티 마커 사용 : 타입 미포함, 스팬 풀링 미적용(단순 [CLS] 활용)

성능 결과

*validation set 기준

	Micro F1(no relation excl.)	AUPRC
엔티티 마커 사용X	0.5942	0.4824
엔티티 마커 사용O	0.8288	0.7757

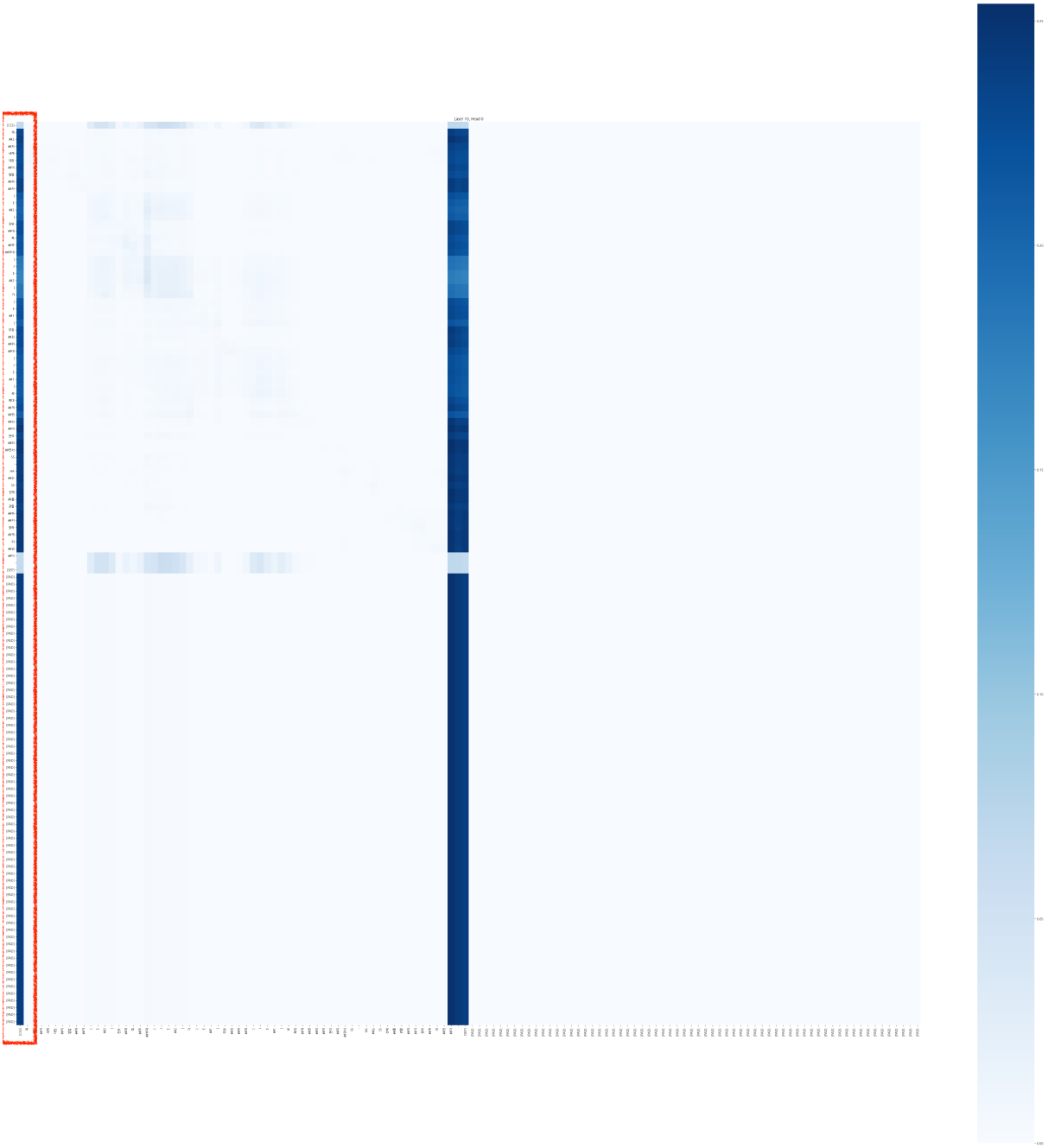
- 하위 클래스 5개 (희소 클래스)의 평균 recall이 34.44%



- 모델이 `no_relation`에 치우쳐 예측하는 경향을 보임 (실제 validation set의 `no_relation` 의 비중은 29.36%이나 예측결과는 34.08%)
- 엔티티 마커 사용만으로도 성능이 크게 오르는 것을 확인(micro-f1: 23%p, auprc: 29%p)

분석 결과

- 데이터 불균형 상황에서 단순 CE Loss로는 희소 클래스를 제대로 인식하지 못함



- 엔티티 마커만으로도 엄청난 성능 향상이 있는 것으로 보였지만, [CLS]에 의존해 스팬 내부 정보를 활용하지 않아 엔티티 집중 학습이 제한되는 것으로 보임.
- 이는 Focal Loss, 스팬 풀링 등을 도입해 `no_relation` 편향을 완화할 필요가 있음을 보여준다.

8.2 Focal Loss 적용 후 성능 변화

실험 설정

- 나머지 모델, 파라미터 설정 8.1과 동일 (klue/roberta-base, max_length=128, batch_size=16, learning_rate=3e-5, linear scheduler, 엔티티 마커 사용)
- loss를 Focal Loss로 변경 ($\gamma = 1.0, \alpha = 0.25$)
- label smoothing (0.1) 함께 사용

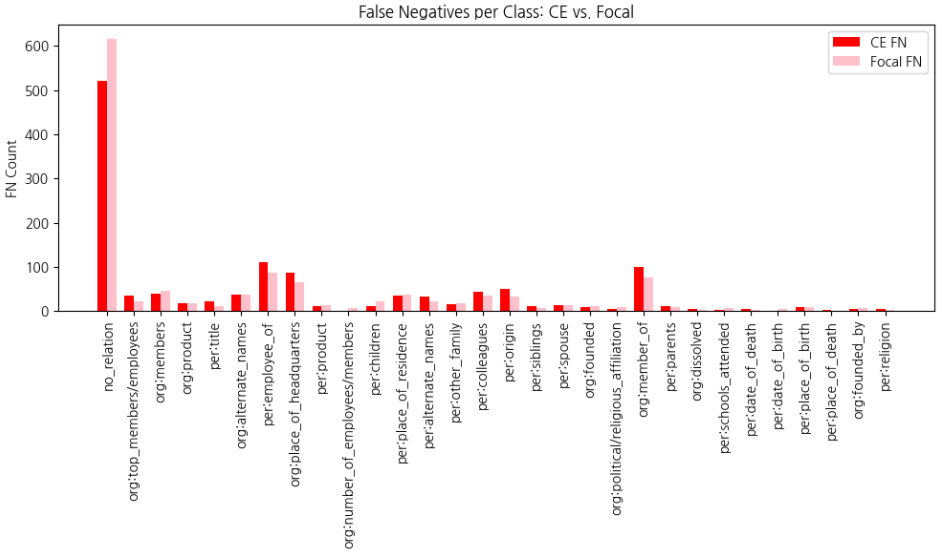
성능 결과

*validation set 기준

	Micro F1(no relation excl.)	AUPRC
Cross Entropy Loss	0.8288	0.7757
Focal Loss	0.8327	0.7890

- Micro F1이 0.4%p 상승, AUPRC 1.3%p 상승

분석 결과



- Focal Loss가 "어려운 샘플(희소 클래스)"에 더 큰 손실을 부여해, 모델이 이를 놓치지 않도록 유도하는 효과가 validation set에서 확인됨 → 희소 클래스의 FN이 CE를 적용한 모델보다 적은 경향을 확인

8.3 엔티티 스팬 풀링 적용 전후 비교 결과

실험 설정

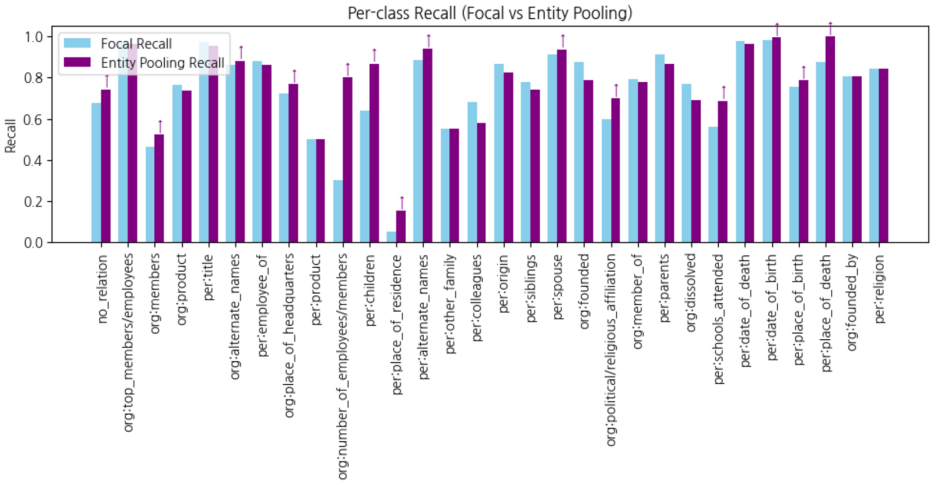
- 엔티티 마커 사용 및 타입 사용
- 스팬 풀링 사용, mean pooling 사용 (5.2 참고)
- Focal Loss ($\gamma = 1.0, \alpha = 0.25$) 및 label smoothing (0.1) 유지
- 나머지 파라미터 이전 설정과 동일 (klue/roberta-base, max_length=128, batch_size=16, learning_rate=3e-5, linear scheduler, 엔티티 마커 사용)

성능 결과

*validation set 기준

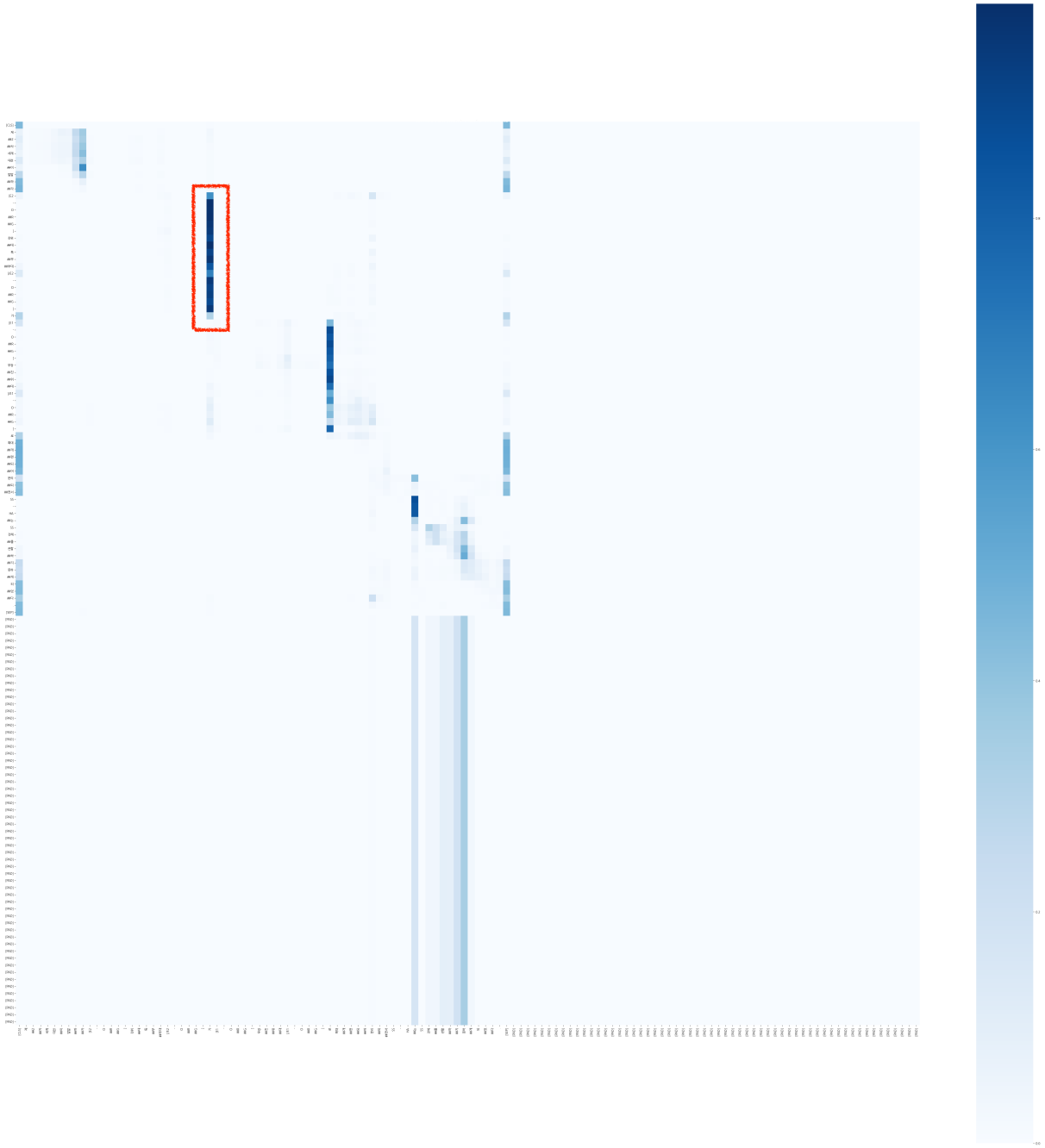
	Micro F1(no relation excl.)	AUPRC
스팬 풀링 미적용	0.8327	0.7890
스팬 풀링 적용	0.8505	0.7972

- Micro F1이 약 2%p 상승, AUPRC 약 1%p 상승



- 희소 클래스 하위 5개(per:religion, per:schools_attended, org:dissolved, org: number_of_employees/members, per:place_of_death)에 대한 recall이 평균 13.46%p 상승

분석 결과



- 스팬 풀링을 적용함으로써, [CLS] 의존에서 벗어나 엔티티 스팬 임베딩을 직접 뽑아 관계 결정에 활용한 결과, 희소 클래스 식별력이 크게 높아졌다. → `[E1]` 토큰에 집중한 경향을 확인 가능
- 다만, Mean Pooling이라 토큰별 중요도를 구분하지 않아 일부 긴 엔티티에서 여전히 정보가 희석될 가능성이 남는다.

8.4 Attention-based Pooling 추가 도입 결과

실험 설정

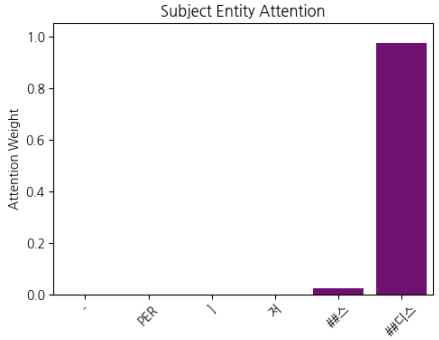
- 엔티티 마커 및 타입 사용
- 스팬 풀링 사용, attention pooling 사용
- Focal Loss ($\gamma = 1.0, \alpha = 0.25$) 및 label smoothing (0.1) 유지
- 나머지 파라미터 이전 설정과 동일 (klue/roberta-base, max_length=128, batch_size=16, learning_rate=3e-5, linear scheduler, 엔티티 마커 사용)

성능 결과

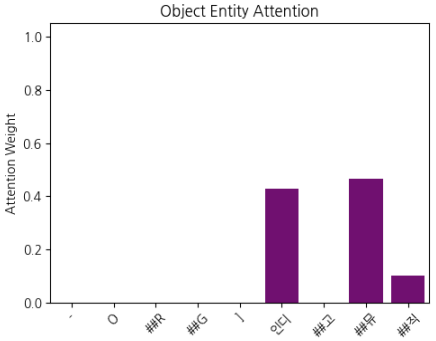
*validation set 기준

	Micro F1(no relation excl.)	AUPRC
attention pooling 미적용	0.8505	0.7972
attention pooling 적용(with linear scheduler)	0.8472	0.7909
attention pooling 적용(with polynomial scheduler, 학습률 1.5e-05)	0.8513	0.8050

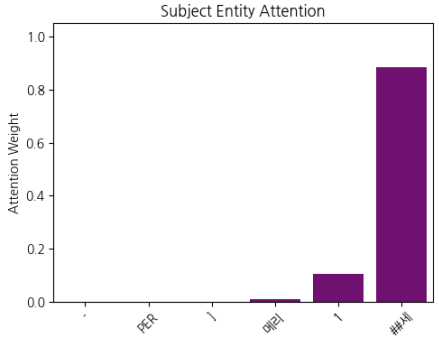
- Attention Pooling 도입 시 파라미터와 학습 복잡도가 커져, 수렴 속도 문제로 성능이 떨어지는 것을 확인
- 이를 보완하기 위해 Scheduler를 Linear → Polynomial으로 변경, 학습률 조절
- Polynomial Scheduler를 적용한 Attention Pooling 모델은 기존 대비 Micro F1 0.05%p 상승, AUPRC 1% 상승



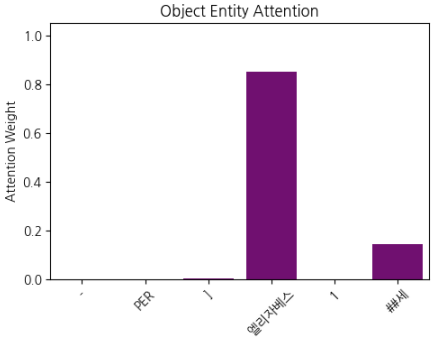
per:employ_of



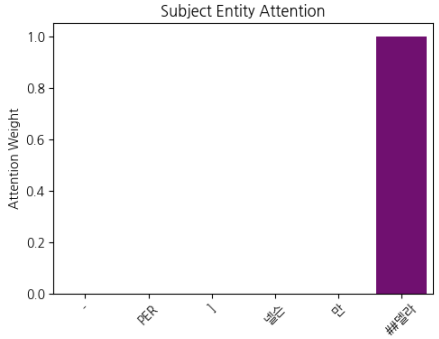
per:employ_of



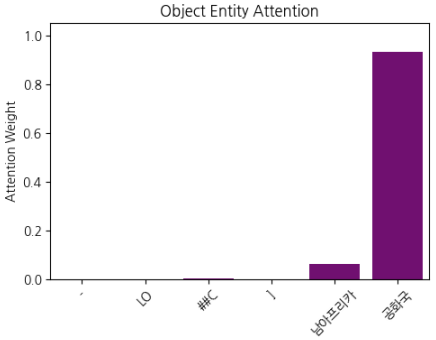
per:other_family



per:other_family



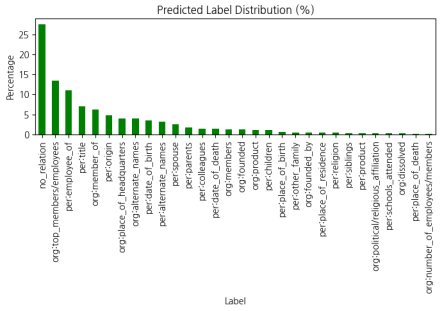
per:origin



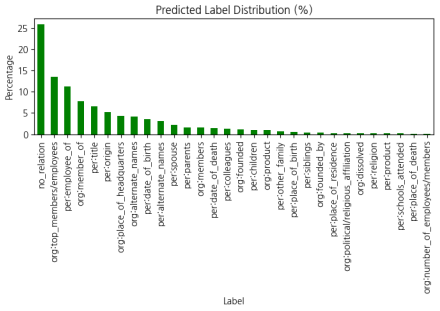
per:origin

- 모델이 엔티티 내부 토큰 중 핵심 키워드에 집중해 예측하는 경향을 보임

분석 결과



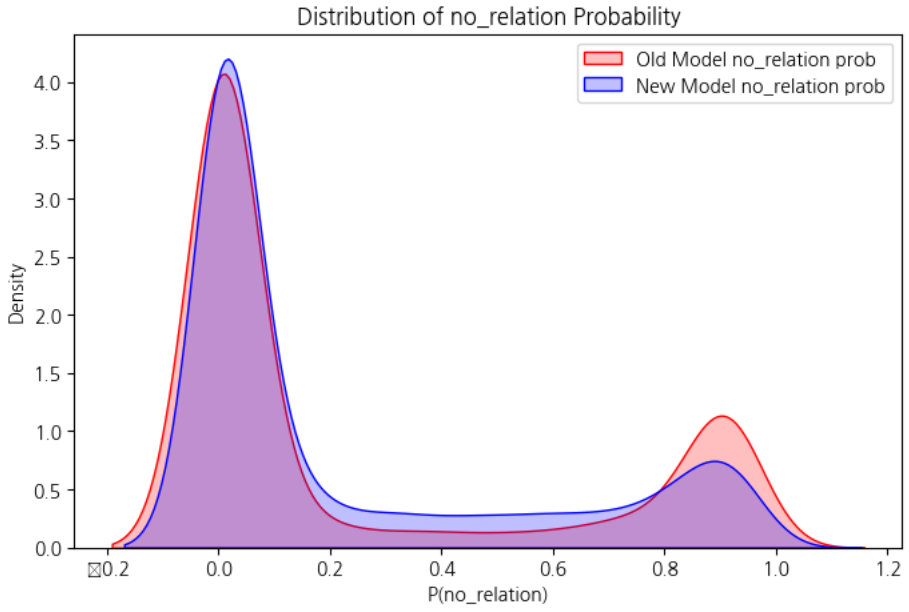
attention pooling 미적용 모델



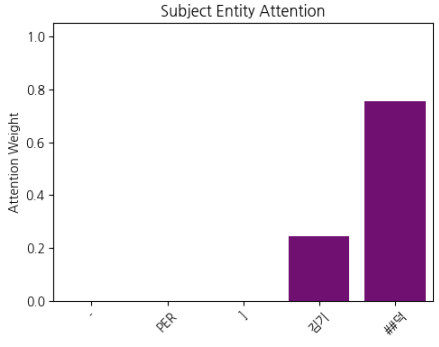
attention pooling 적용 모델

`no_relation` : 27.64%

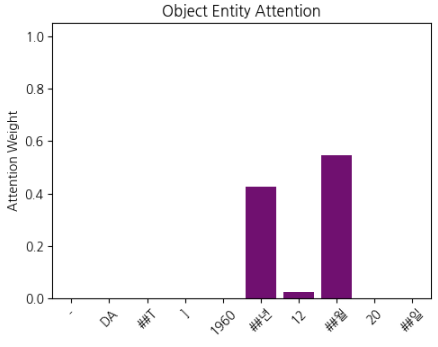
`no_relation` : 25.85%



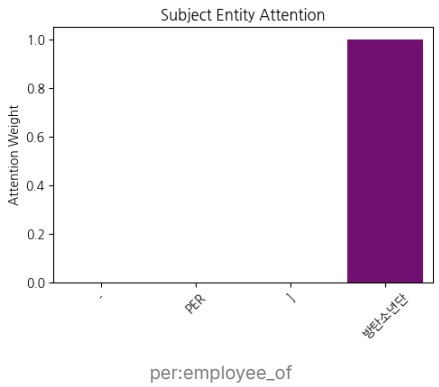
- `no_relation` 편향도 더 낮아졌고, 희소 클래스를 아예 무시하던 예전과 달리, 모델이 적절히 확률을 분산시키는 형태를 관찰.



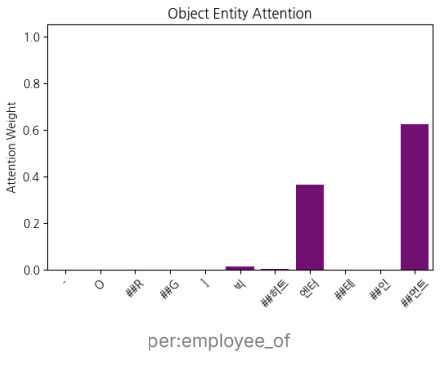
per:date_of_birth



per:date_of_birth



per:employee_of



per:employee_of

- 엔티티 구간이 길거나 여러 표현이 섞인 경우(예: 긴 기관명), 실제로 Attention이 “핵심 토큰”에 가중을 집중시키는 사례가 관찰됨

8.5 앙상블 기법 적용 시 성능 변화 및 비교

앙상블 대상 모델

- KLUE-RoBERTa(klue/roberta-large)
- KoELECTRA(monologg/koelectra-base-v3-discriminator)
- DeBERTa(team-lucid/deberta-v3-base-korean)

앙상블 과정

1. 각 모델(KoELECTRA, KLUE-RoBERTa, DeBERTa)을 `optuna` 를 이용하여 모델의 하이퍼파라미터 최적화하고, Micro-F1 기준으로 가장 성능이 좋은 모델을 선택
2. 최적화된 모델 3가지를 가지고 soft voting, logit averaging, weighted logit averaging을 모두 시도
3. 이 때 weighted logit averaging에서 weight는 각 모델의 f1 score가 전체 f1 score의 sum에서 차지하는 비중으로 구함

앙상블 대상 모델별 최적화 하이퍼 파라미터

- KLUE-RoBERTa-large

```
batch_size : 32
num_epochs : 3
learning_rate : 2.80026895611933e-05
max_length : 200
dropout : 0.0
scheduler : linear
focal_loss : false
label_smoothing : 0.15000000000000002
span_pooling : true
attention_pooling : false
entity_markers : true
entity_types : false
```

- KoELECTRA

```
batch_size : 16
num_epochs : 6
learning_rate : 2.8262768586139738e-05
max_length : 128
dropout : 0.0
scheduler : linear
focal_loss : false
label_smoothing : 0.2
```

```
span_pooling : false
entity_markers : true
entity_types : true
```

- DeBERTa

```
batch_size : 8
num_epochs : 8
learning_rate : 2.9361705292170236e-05
max_length : 128
dropout : 0.2
scheduler : linear
focal_loss : true
label_smoothing : 0.05
alpha : 0.25
gamma : 2.0
span_pooling : false
entity_markers : true
entity_types : false
```

성능 결과

*테스트 데이터 기준

	Micro F1(no relation excl.)	AUPRC
soft voting	69.5315	73.0182
logit averaging	69.4768	73.4883
weighted logit averaging (DeBERTa : 0.1, KoELECTRA : 0.2, RoBERTa : 0.7)	71.7391	75.7794

9. 논의 및 한계점

9.1 불균형 문제에 대한 해결 효과 분석

Focal Loss 등을 통한 개선 확인

- 8장 결과에서, Focal Loss를 적용했을 때 희소 클래스(예: org:founded_by, per:origin) FN 수를 줄인 것을 확인.
- 이는 EDA(2.1)에서 발견된 `no_relation` 편향과 희소 클래스 무시 문제를 완화하기 위한 주요 전략이 작동했음을 의미한다.

잔여 편향 해결 필요성

- 아직 일부 class(`org:place_of_residence` , `per:product`)에 대해서는 Recall이 여전히 낮은 경우가 있어, 편향이 완전히 해소되지는 않음.
- 이는 Focal Loss+Label Smoothing만으로도 한계가 있음을 시사하며, 데이터 증강이나 앙상블(6장)로 조금 더 보완이 필요하다는 점을 재차 확인.

9.2 엔티티 스팬 풀링 및 앙상블 기법 도입 결과 분석

엔티티 스팬 풀링 성과

- EDA 상 문제였던 “희소 클래스”에서도 Recall이 높아진 사례가 다수 관찰되었다(8.3).
- 또한, Attention Pooling에서 보다 높은 성능을 얻을 수 있었는데, 스팬 내 핵심 키워드를 평균 혹은 Attention Pooling(8.4)으로 강조하면서, 적은 수의 샘플만으로도 관계 분별력을 높인 것으로 해석된다.
- Attention Pooling(8.4)으로 확장 시 추가 파라미터, 학습률 튜닝이 필요하다는 점을 제외하면, 구현 난이도 대비 성능 향상에서 얻는 이점이 더 큼.

앙상블 기법 적용 결과

- 8.5에서, RoBERTa 모델과 KoELECTRA 기반 모델 등 서로 다른 특성을 지닌 모델을 학습 후 앙상블.
- Weighted Logit Averaging 방식으로 합쳤을 때, 단일 모델보다 0.5%p정도 Micro-F1이 추가로 향상.
- 개별 모델 간 성능 격차가 큰 경우, 단순 앙상블(특히 동일 가중치)은 오히려 Micro-F1 및 AUPRC 하락을 초래할 수 있음을 확인.

9.3 추후 개선 방향

Span-based 모델(LUKE, SpanBERT) 한국어 파인튜닝

LUKE

- LUKE는 엔티티 어웨어(Span-based) Pre-training을 거친 후, **영어** 관계 추출 등에 우수한 성능을 보이는 것으로 알려져 있음.
- 한국어 적용을 위해서는 "LUKE의 한국어 버전(koLUKE)"가 필요하거나, 원본 LUKE를 **한국어 코퍼스**로 추가 파인튜닝(도메인 어댑테이션)할 수 있음.

SpanBERT

- 본 과제에서는 RoBERTa 기반 스팬 풀링을 직접 구현했으나, SpanBERT처럼 스팬 예측에 특화된 언어모델을 사용하면 희소 클래스나 엔티티 간 관계 이해력이 더 높아질 잠재력이 있음.

적용 방안

- **영어 LUKE/SpanBERT를 한국어 코퍼스에 도메인 어댑테이션**: 대규모 한국어 텍스트(위키 등)로 "additional pre-training" 후 Relation Extraction에 파인튜닝.

기대 효과

- 스팬 기반 Pre-training으로 엔티티와 관계를 구조적으로 학습 → 기존 RoBERTa 대비 **한층 높은 Relation Extraction 성능** 기대.

데이터 증강

EDA & AEDA

- EDA(Easy Data Augmentation): Synonym Replacement, Random Insertion, Random Swap, Random Deletion 등으로 문장을 변형해 희소 클래스 데이터 수를 늘릴 수 있다.
- **AEDA(An Easier Data Augmentation)**: 기존 EDA보다 단순화된 문장 부호 삽입·변형 방식 등을 활용해, 원문 흐름을 크게 깨뜨리지 않고 문장 변형을 할 수 있다.

Honorific Transformation

- 한국어 문장에서 존댓말("합니다")을 반말("한다") 등으로 변환하거나, 호칭어("님", "씨") 등을 추가/제거하는 방식으로 어체를 다양화.

Masking & Infilling

- 특정 핵심 단어(예: 엔티티 내부 단서) 또는 보조 단어를 [MASK] 처리하고, 사전학습모델(PLM)로 그 빈칸을 추론·채우게 하여 새로운 문장을 생성.
- 희소 클래스 문장 중 일부 키워드를 마스킹 후, 유사하지만 미묘하게 다른 단어로 치환하면 더 많은 변형 데이터를 확보 가능.

기대 효과

- 희소 클래스(예: org:founded_by, per:colleagues 등)에 대한 데이터가 상대적으로 크게 늘어나고, 다양한 표현(존댓말/반말, 문장 부호 삽입, 마스킹 복원)으로 모델이 일반화 능력을 높일 것으로 예상.
- 엔티티 내부 토큰도 번갈아 채워 넣음으로써, 모델이 다양한 키워드 변형을 학습하게끔 유도.

10. 결론

10.1 주요 성과 정리

불균형 문제 해결의 실질적 개선

- 본 과제의 주요 목표였던 “라벨 불균형으로 인한 희소 클래스 식별력 저하” 문제를 Focal Loss, Label Smoothing 등을 통해 크게 완화함
- EDA 단계에서 제기된 `no_relation` 편중 현상과 희소 클래스의 Recall 저하 문제가, 학습 후 `no_relation` 제외 Micro F1이 초반 약 59%에서 최대 83%(엔티티 마커 및 loss 변경) 이상으로 개선되는 성과로 이어짐.

엔티티 스팬 풀링 도입

- 단순 [CLS] 임베딩 방식에서 벗어나, 엔티티 구간 전체의 토큰 임베딩을 pooling(Mean/Attention)하여 약 2%p 성능 상승을 실현

앙상블 기법 적용으로 최종 점수 향상

- 서로 다른 백본을 최적화 후 weight logit averaging 등의 앙상블을 통해 단일 모델 대비 0.5%p 추가 상승

하이퍼파라미터 최적화

- Focal Loss 파라미터(α, γ)나 Label Smoothing 정도(0.05~0.2), Scheduler(Linear/Cosine/Polynomial 등) 등 다양한 범위를 `Optuna`로 탐색했다.
- 그 과정에서, 최적 조합 시 Micro-F1이 소폭 상승했으며, 지나치게 공격적인 γ 설정 시 학습이 불안정해질 수 있다는 점을 발견해 파라미터 중요성을 재확인하였다.

10.2 향후 연구 방향 및 추가 실험 계획

Span-based 사전학습 모델(LUKE, SpanBERT) 한국어화

- 9.3에서 논의한 바와 같이, LUKE나 SpanBERT 같은 **스팬 기반 Pre-training** 모델을 한국어 도메인에 맞춰 Fine-tuning하거나, 추가 Pre-training(도메인 어댑테이션)을 수행.
- 본 과제에서 직접 스팬 풀링을 구현한 것보다 더 근본적으로 “스팬 인식”을 pre-trained한 모델을 사용하면 관계 추출 정확도가 한층 높아질 가능성이 크다.

데이터 증강

- 9.3에서 논의한 바와 같이, 문장 내 단어 삽입·치환·삭제(Easy Data Augmentation, AEDA)에서 나아가 존댓말/반말 전환(Honorific Transformation), 특정 단어 Mask 후 PLM으로 복원(Masking & Infilling) 등의 방법이 있다. 이들을 한 파이프라인에 결합하여, 단일 문장이라도 여러 스타일·어조·단어 조합이 생성되도록 한다. 이를 통해 기존 희소 라벨에 속한 문장도 다양한 변형 버전을 얻어, 모델이 여러 표현 패턴을 학습할 기회를 극대화할 예정
- 위 증강 과정을 희소 라벨 샘플(예: org:founded_by, per:colleagues) 위주로 집중 적용해, 극소량의 원본 데이터를 여러 방식으로 변형·확장할 것. 단순 어휘 치환뿐 아니라, 예의체/평서체 변환이나 핵심 토큰 일부 Mask & Infilling을 함께 수행함으로써, 고유한 문맥은 유지하되 표현이 달라진 새로운 학습 예제를 확보해 모델 편향을 완화하고 일반화 성능을 높이고자 함.

10.3 참고 논문

1. **Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P.** (2017).

Focal Loss for Dense Object Detection. (https://arxiv.org/abs/1708.02002)

- 원래 객체 검출(Object Detection)을 위해 제안된 Focal Loss 개념을 소개한 논문으로, 불균형 데이터에서 γ와 α 파라미터를 이용해 쉽고 잦은 예제보다 어려운/희소 예제에 가중을 더 주도록 설계한 방법. 본 과제에서 희소 라벨(Focal Loss)을 적용할 때 큰 참조가 됨.

2. **Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., & Levy, O.** (2020).

SpanBERT: Improving Pre-training by Representing and Predicting Spans. (https://arxiv.org/abs/1907.10529)

- 스팬 단위 마스킹(Span Masking)과 예측을 통해, BERT보다 스팬 중심 학습을 심화한 사전학습 모델을 제안. 관계 추출(RE) 등 엔티티 간 스팬을 활용하는 태스크에서 효과가 큰 것으로 보고되었으며, 본 분석 보고서에서 스팬 풀링 아이디어 확장 및 한국어 SpanBERT 적용 가능성을 검토할 때 참고.

3. **Yamada, I., Asai, A., Shindo, H., Takeda, H., & Matsumoto, Y.** (2020).

LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention.
(
https://arxiv.org/abs/2010.01057)

- LUKE는 엔티티 어웨어(Entity-aware) 사전학습을 통해, 특히 Named Entity Recognition(NER)과 Relation Extraction(RE)에서 우수한 성능을 보이는 모델. 본 분석 보고서의 스팬 풀링(5.2, 5.3, 8.3, 8.4) 접근과 유사하며, 한국어 버전이 있다면 관계 추출 성능을 크게 끌어올릴 잠재력이 있다고 판단.

4. **Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K.** (2019).

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

(https://arxiv.org/abs/1810.04805)

- BERT는 본 분석 보고서에서 사용된 RoBERTa, KoELECTRA 등 여러 한국어 PLM의 기반 아이디어. Transformer 기반 양방향 사전학습으로 NLP 전반의 성능을 끌어올린 핵심 연구.

5. **Park, S., et al.** (2021).

KLUE: Korean Language Understanding Evaluation. (https://arxiv.org/abs/2105.09680)

- 한국어 전용 벤치마크 KLUE에서 RoBERTa, KoELECTRA 등 한국어 PLM들이 제시되었고, Relation Extraction 태스크("KLUE-RE")도 포함. 본 과제에서 모델 선택(klue/roberta-base)과 성능 비교 지점으로 참고.

6. **Zhou, et al.** (2022).

An Improved Baseline for Sentence-level Relation Extraction. (https://arxiv.org/abs/2102.01373)

- Entity Marker가 Relation Extraction Task에서 주요한 역할을 한다고 주장한 연구

11. Appendix(결과 정리)

*검증 데이터 기준

- base model : klue/roberta-base

Model Setting	Micro-F1 (no_relation excl.)	AUPRC
(A) Model Only	0.5942	0.4824
(B) Entity Marker	0.8288	0.7757
(C) Entity Marker + Focal Loss	0.8327	0.7890
(D) Entity Marker + Focal Loss + Span Pooling(mean)	0.8505	0.7972
(E) Entity Marker + Focal Loss + Span Pooling(attention)	0.8513	0.8050

*검증 데이터 기준

- 앙상블 대상 모델

Model	Optimized Parameter	Micro-F1 (no_relation excl.)	AUPRC
(A) deberta-korean-base (optimized)	batch_size : 8 num_epochs : 8 learning_rate : 2.9361705292170236e-05 max_length : 128 dropout : 0.2 scheduler : linear focal_loss : true label_smoothing : 0.05 alpha : 0.25 gamma : 2.0 span_pooling : false entity_markers : true entity_types : false	0.8191	0.7431
(B) koelectra-v3-discriminator (optimized)	batch_size : 16 num_epochs : 6 learning_rate : 2.8262768586139738e-05 max_length : 128	0.8389	0.7648

Model	Optimized Parameter	Micro-F1 (no_relation excl.)	AUPRC
	dropout : 0.0 scheduler : linear focal_loss : false label_smoothing : 0.2 span_pooling : false entity_markers : true entity_types : true		
(C) roberta-large(optimized)	batch_size : 32 num_epochs : 3 learning_rate : 2.80026895611933e-05 max_length : 200 dropout : 0.0 scheduler : linear focal_loss : false label_smoothing : 0.15000000000000002 span_pooling : true attention_pooling : false entity_markers : true entity_types : false	0.8593	0.8235

*테스트 데이터 기준

Model	Micro-F1 (no_relation excl.)	AUPRC
(D) (A) + (B) + (C) soft voting	69.5315	73.0182
(E) (A) + (B) + (C) logit averaging	69.4768	73.4883
(F) (A) + (B) + (C) weighted logit averaging ((A) : 0.1, (B) : 0.2, (C): 0.7)	71.7391	75.7794

*테스트 데이터 기준

Model Variant	Loss Function	Label Smoothing	LR Scheduler	Entity Marker(Type included)	Span Pooling	Attention Pooling	Micro-F1 (no_relat excl.)
RoBERTa base	Focal Loss	0.1	Linear	Yes (With Type)	Yes	No	65.4346
RoBERTa large	Focal Loss	0.1	Linear	Yes (With Type)	Yes	No	67.3219
RoBERTa large	Focal Loss (Optimized)	0.09	Linear	Yes (With Type)	Yes	No	68.7285
RoBERTa large	Focal Loss (Optimized)	0.09	Cosine	Yes (With Type)	Yes	No	71.1004