**최종점수 : 80.7% (16위)**

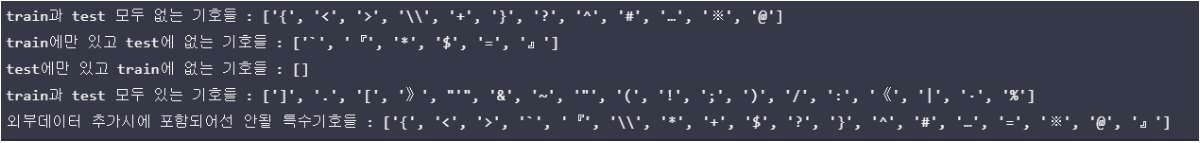
**시도해본 방법들**

**데이터 전처리**

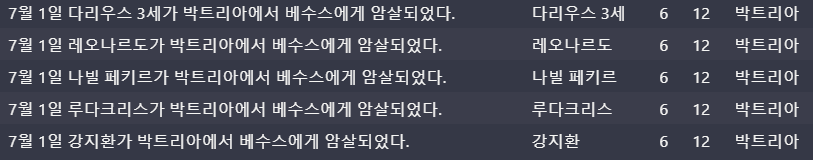
* 특수기호 전처리
  + test set에는 존재하지 않는데 train에는 존재하는 특수기호들이 존재하였고 train 중 이러한 문장들이 차지하는 비율이 매우 극소수였기 때문에
  + 이들을 각각 합리적이라고 생각하는 기준(제 주관)에 따라 제거하거나 적당한 값으로 바꾸어주었습니다.

[ \* , 『 , 』 , =, $ ] 와 같은 기호들

* + - 별표는 확인결과 아이디 마스킹, 혹은 특정 그룹명에 쓰여있는 것을 발견하여 제거하여도 무방하다고 판단하여 제거하였습니다.
    - 『 혹은 』를 포함하는 sentence 확인결과



* Minority 부분 데이터 증강
  + KFold 학습을 진행하기 위해 Minority Class 부분이 K개 이상 필요하여 증강을 시도한 정도에 그쳤습니다.
  + 단순 Resampling 대신 이용하기 위한 목적이 가장 컸습니다.
  + 실제 적용시에도 두 개의 클래스에만 적용하였습니다.
  + Minority 부분에 대해서만 Pororo 번역 Augmentation 시도
  + 아래는 본래의 entity를 동일 카테고리의 단어로 치환하는 Augmentation (맨 윗문장이 원본문장)



이는 단순 paraphrasing일 뿐 그 의미가 유사하기 때문에 어떤 성능의 증대를 기대하고 시도한 방법은 아니었고, 성능의 변화가 그리 크지 않다는 것을 직접 느낄 수 있었습니다.

* 외부데이터 추가
  + 본래 학습데이터의 Max Token Length를 고려하여 필터링 후 데이터 추가
  + 본래 학습데이터의 Label 비율을 고려하여 Label별 데이터 추가

개인적으로 당연히 해야하는 전처리라고 생각하고 진행했던 처음부터 진행하였던 부분이었습니다.

KOELECTRA 모델을 활용하며 나머지 환경을 컨트롤하며 실험해본 결과 성능이 더 나아지는 결과가 나와서 끝까지 해당 데이터를 추가하여 분석을 진행하였습니다.

오히려 마지막에는 모델이 최고 성능을 저해하는 요소로 작용하지 않았나 다시금 생각해보게 되었습니다

**아쉽게 시도해보지 못한 방법론**

* Pororo 라이브러리 활용 NER 태깅을 통한 Sentence 변경후 추가
  + 데이터 전처리 부분의 코드에 작성은 잘 완료했지만,

**모델 활용**

* KoBERT
* Bert-Multilingual
* Koelectra
* XLM-Roberta

결론 : XLM-Roberta 모델의 성능이 가장 좋았습니다. 다양한 시도를 해보았지만 결국 가장 큰 모델이 성능이 높다는 점이 약간 아쉬운 부분이었기도 합니다.

**하이퍼파라미터**

* max\_lr : 1e-5
* max\_lr : 5e-7
* Customized Cosine Scheduler with Warmup
* Batch size : 32
* Label Smoothing Factor : 0.1

위처럼 적당한 lr과 배치사이즈를 빠르게 찾은 후, HyperParmeter를 튜닝하기 위한 많은 노력은 들이지 않았습니다.

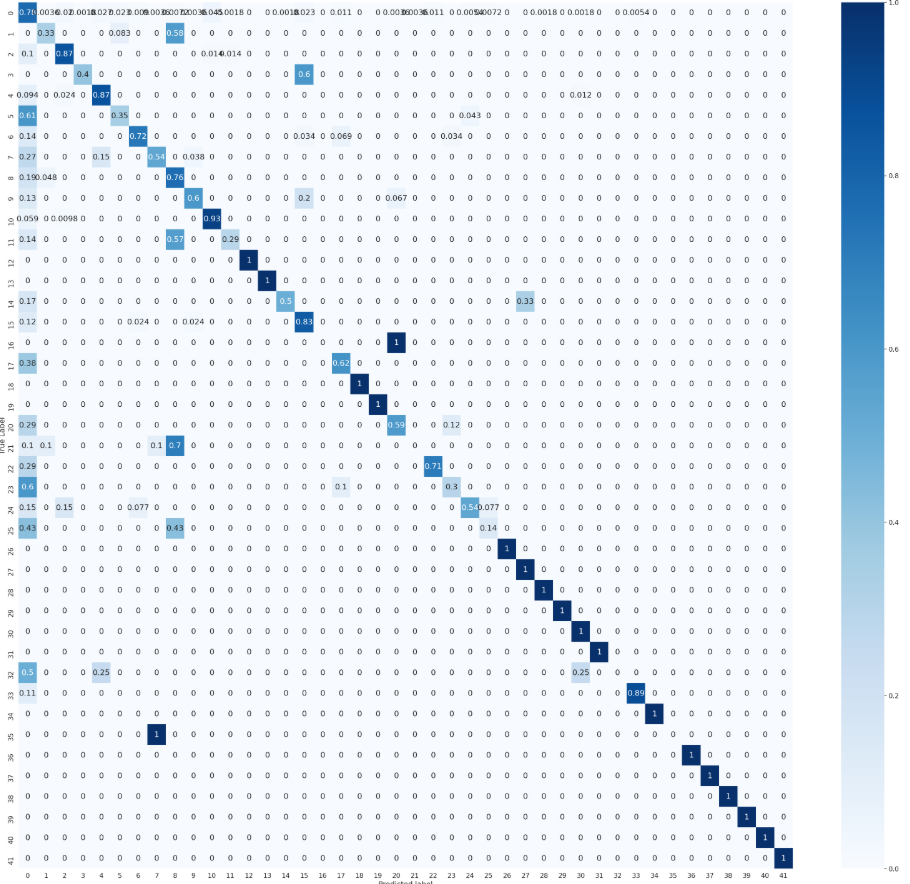
**검증방법 : 5-Fold Ensemble 활용**

| **Name** | **Value** |
| --- | --- |
| [0-th val\_acc\_score](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=0-th%20val_acc_score) | 0.7882960413080895 |
| [1-th val\_acc\_score](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=1-th%20val_acc_score) | 0.806368330464716 |
| [2-th val\_acc\_score](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=2-th%20val_acc_score) | 0.7960413080895009 |
| [3-th val\_acc\_score](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=3-th%20val_acc_score) | 0.817399 |
| [4-th val\_acc\_score](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=4-th%20val_acc_score) | 0.813953488372093 |
| [Result ACC : 5-fold val Total Average acc](https://app.neptune.ai/jh951229/Pstage-2-EntityRelationExtraction/e/PSTAG2-228/all?path=logs&attribute=Result%20ACC%20%3A%208-fold%20val%20Total%20Average%20acc) | 0.8042872279629776 |

5-Fold CV 를 통해 검증을 하였는데 실제 Public LB 점수와 매우 유사하게 나와서 검증 셋이 잘 분할되었음을 수차례 느낄 수 있었습니다.

**Confusion Matrix 활용**

Fold 별 모델이 학습되면서의 Valid Set에 대한 Prediction 결과를 Confusion Matrix로서 다음과 같이 저장하며 어느 클래스를 옳게, 혹은 틀리게 구별했는지 파악하려 노력하였습니다.



**LR Scheduler 활용**

다음과 같이 직접 구현한 CustomizedCosineScheduler with Warmup Restarts를 이용하여 학습하였고, 이를 통해 Learning Rate를 튜닝함에 있어 그리 큰 노력을 들이지 않았음에도 안정적으로 학습할 수 있었습니다.

