부스트캠프
AI Tech

[P2] Relation Extraction Competition Wrap-up Report

점수 및 순위

- Public and Private LB / 81.9% / 공동 1위

Final Submit의 Options

Inference and Ensemble

- 예측 시 Class 0에 가중치 추가
- Classifier가 다른 3개 Model의 Inference를 Hard Voting하여 개인 Submit 생성
- 정익효님, 권태양님의 Submit과 Hard Voting Ensemble을 통해 Final Submit 생성

Models

- XLM-RoBERTa-large + Sequence Classifier
- XLM-RoBERTa-large + LSTM Classifier
- XLM-RoBERTa-large + R-BERT Classifier

Data

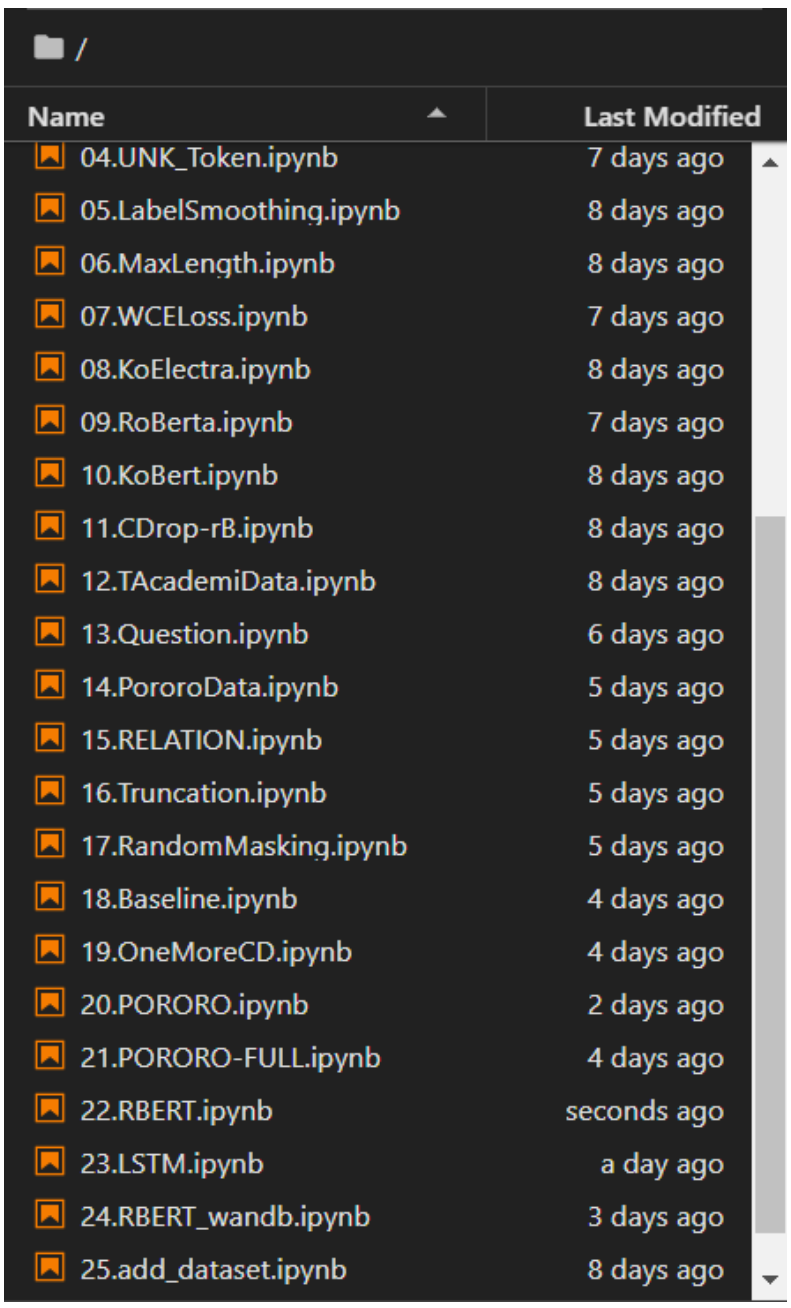
- Back Translation을 통한 Data Augmentation
- Random Masking을 통한 Data Augmentation
- Truncation을 통한 Text 전처리

Training

- AdamP Optimizer 사용
- Cosine Annealing LR Scheduler 사용
- Class Imbalance Data에 따라서, Focal Loss 사용
- 각 Model의 적합한 Hyper Parameter Search 및 사용
- Stratified K Fold로 Validation Set 구성 및 Model 평가
- Multi Sentence와 Plain Text를 구분자로 사용해 Model 학습

성능 향상을 위해 시도한 실험

다양한 Idea와 설정으로 실험을 수행했다.



Version별 Code 예시

내 제출

Submission ID	Phase	Description	Type	Created_at	Download
50	Finished	hard voting 2	File	2021-04-22 22:13	
49	Finished	one more	File	2021-04-22 21:59	
48	Finished	6 plus koelectra	File	2021-04-22 21:51	
47	Finished	hard voting	File	2021-04-22 20:45	
46	Finished	LSTM-RBERT-PORORO64	File	2021-04-22 17:47	
45	Finished	RBERT 3models (batch32), PORORO batch 64	File	2021-04-22 12:28	
44	Finished	RBERT 5models (batch32), PORORO batch 64	File	2021-04-22 12:23	
43	Finished	pororos and rbert-dr03	File	2021-04-22 08:09	
42	Finished	Ensemble, Matched PORORO Data, Truncatio...	File	2021-04-22 08:09	
41	Finished	RBERT PORORO64 softvoting	File	2021-04-22 07:55	

Version별 제출 예시 (총 50회)

Name (3 visualized)	State	Notes	User	Tags	Created	Runtime	Sweep	MODEL_NAME	__wrapped__
Question-large-ml100	finished	Add notes	hkl	Questic	6d ago	30m 50s	-	xlm-roberta-large	wandb.sdk.wandb_config.Config
roberta-large (special token </:	finished	Add notes	hkl	add-UN	7d ago	34m 47s	-	xlm-roberta-large	wandb.sdk.wandb_config.Config
roberta-large-150	finished	Add notes	hkl	add-UN	1w ago	27m 36s	-	xlm-roberta-large	wandb.sdk.wandb_config.Config
roberta-base-150	finished	Add notes	hkl	add-UN	1w ago	13m 16s	-	xlm-roberta-base	wandb.sdk.wandb_config.Config
roberta-base-maxlength200	finished	Add notes	hkl	maxlen	1w ago	5m 14s	-	xlm-roberta-base	wandb.sdk.wandb_config.Config
TAcademi	finished	Add notes	hkl	TAcade	1w ago	1h 25m 34	-	xlm-roberta-base	wandb.sdk.wandb_config.Config
TAcademi	finished	Add notes	hkl	TAcade	1w ago	41m 38s	-	xlm-roberta-base	wandb.sdk.wandb_config.Config
CDropout03-10epochs	finished	Add notes	hkl	10epoc	1w ago	18m 58s	-	xlm-roberta-base	wandb.sdk.wandb_config.Config
roberta-base	finished	Add notes	hkl	add-UN	1w ago	15m 10s	-	xlm-roberta-base	wandb.sdk.wandb_config.Config

실험 별 Logging 예시

실험 List

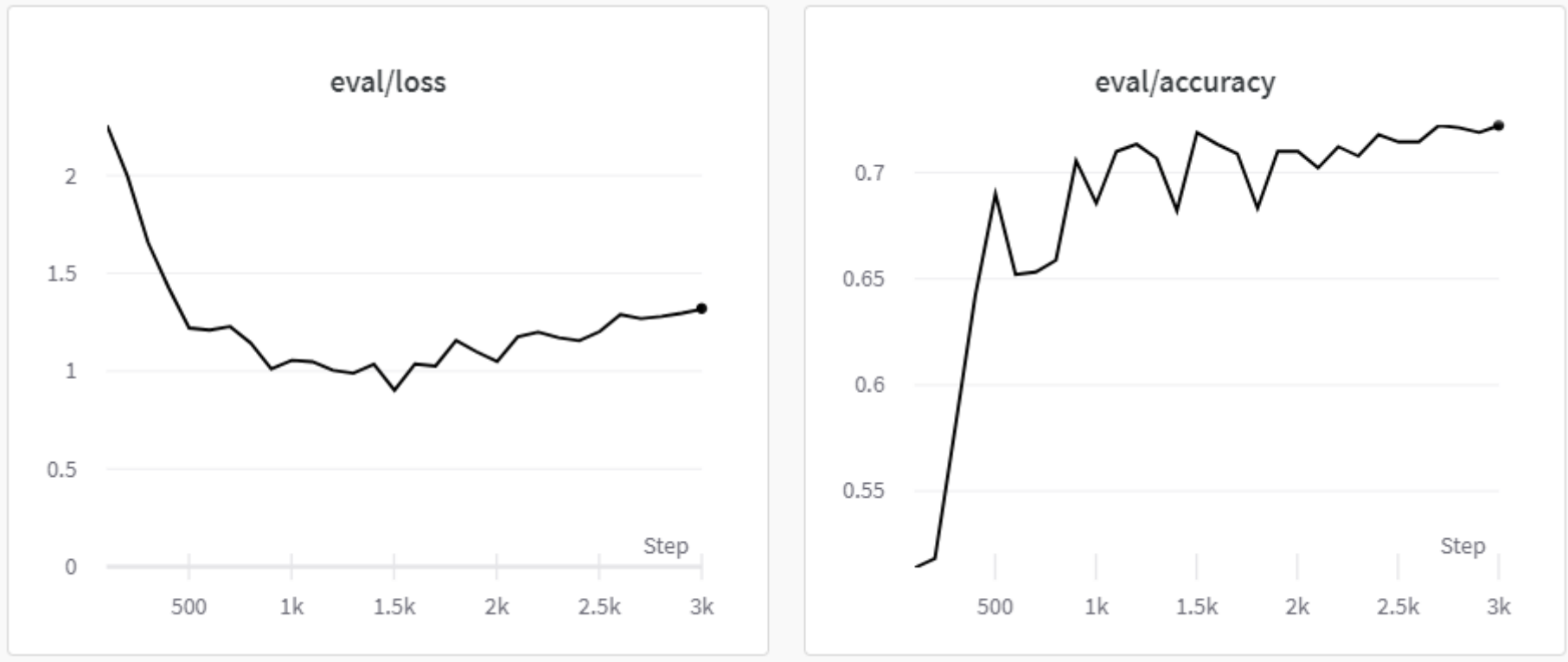
- Pre-trained Model 실험 (BERT Multi-lingual Model)
- Entity Token을 통한 Entity 강조 실험 (Plain Token, Special Token, Sbj and Obj 등)
- Entity 내 Unknown Token 제거 실험 (Tokenizer 별 Special Token 등)
- Label Smoothing Loss (Rate 0.1, 0.2 등)

- Weight Cross Entropy Loss 실험 (Data Count, Doubling 등)
- Backbone Model 실험 (KoBERT, KoElectra, XLM-RoBERTa 등)
- Tokenizer Max Length 실험 (100, 150, 200 등)
- Layer 별 Dropout Rate 실험 (Hidden Layer, Classifier Layer 등)
- 외부 Data 실험 (Kor-RE-Gold, Kor-RE-Silver, T-Academy 등)
- Question Type Input 실험 (Short Sentence, Long Sentence, Keyword 등)
- Back Translation Data 구축 및 실험 (PORORO Library)
- Truncation 실험 (Length 50, 100 등)
- Random Masking 실험 (Threads Hold Rate, In Evaluation Phase 등)
- Add Weight 실험 (Weight 0.1, 0.2 등)
- R-BERT Architecture 실험 (Dropout, Weight Decay, Activation Function)
- LSTM Classifier 실험 (Hyper-parameters, Output Layer 등)
- LR, Batch Size, Scheduler 등 실험 (Warm-up Step, T-Max 등)

실험의 근거와 의의

Pre-trained Model 실험

Baseline Code에서는 Scratch BERT Model을 활용하고 있어, Pre-trained Model의 성능을 확인했다.
Scratch로부터 학습 시킨 Model은 약 57%의 Accuracy Score를 보였으나, Pre-trained Model은 70%를 웃도는 성능을 보여주었다.
이에, Pre-trained Model을 기본 Baseline으로 구성하고 Competition을 진행했다.



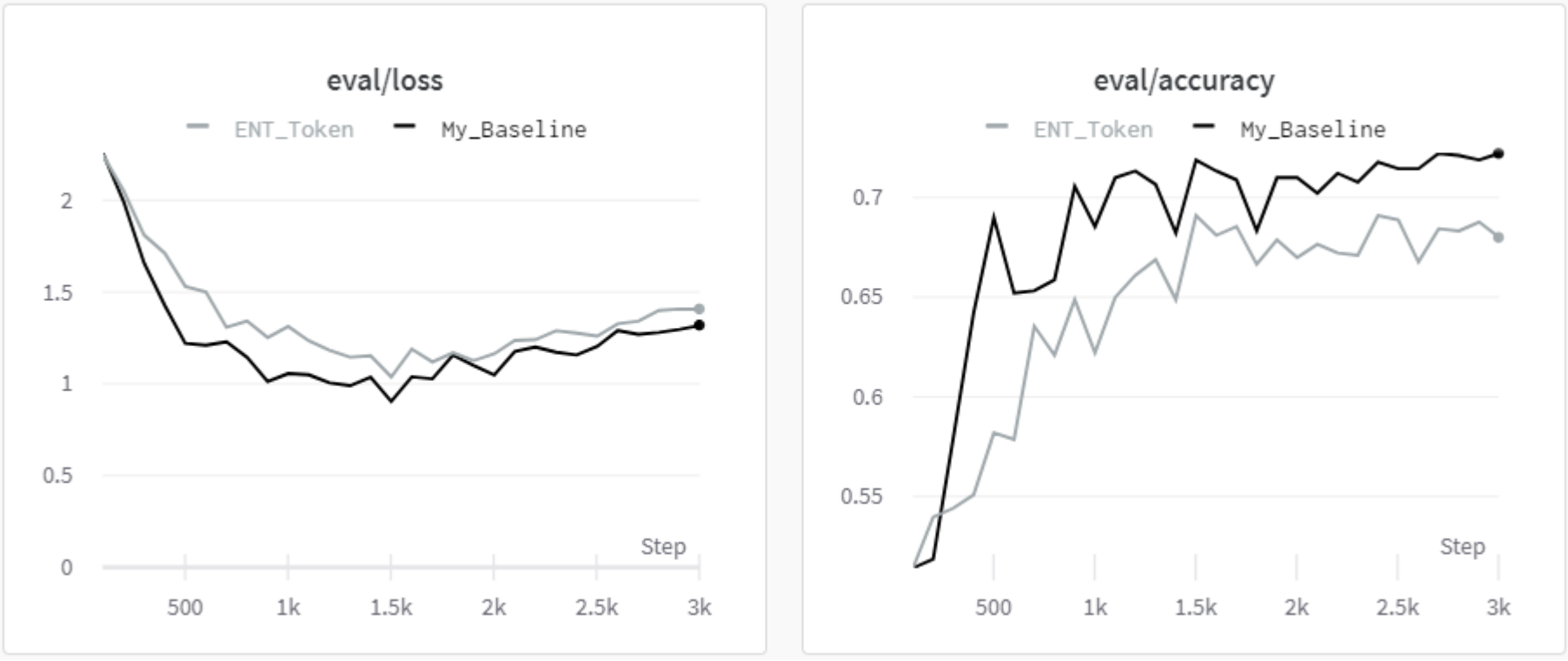
Pre-trained Model의 학습 Graph

Entity Token을 통한 Entity 강조 실험

관계 추출 Task가 기본적으로 단일 Token의 Embedding Vector만을 활용하기 때문에, 관계를 추출하고자 하는 Entity를 강조해줄 필요가 있다고 판단했다. 이에 아래와 같은 방법을 실험했다.

1. 기존 Sentence에 Special Token을 추가해 Entity를 감싸줌.
2. 기존 Sentence에 General Token을 추가해 Entity를 감싸줌.
3. Query의 역할을 하는 Sentence에 Token을 추가함.
4. 각 Entity를 구분 지을 수 있도록 Object와 Subject Token을 추가함.

결과적으로 위 실험을 통해 성능 향상시킬 수는 없었다.
단순히 Token을 추가하는 것만으로는 Model에게 적합한 Feature를 전달할 수 없다는 결론을 내렸다.



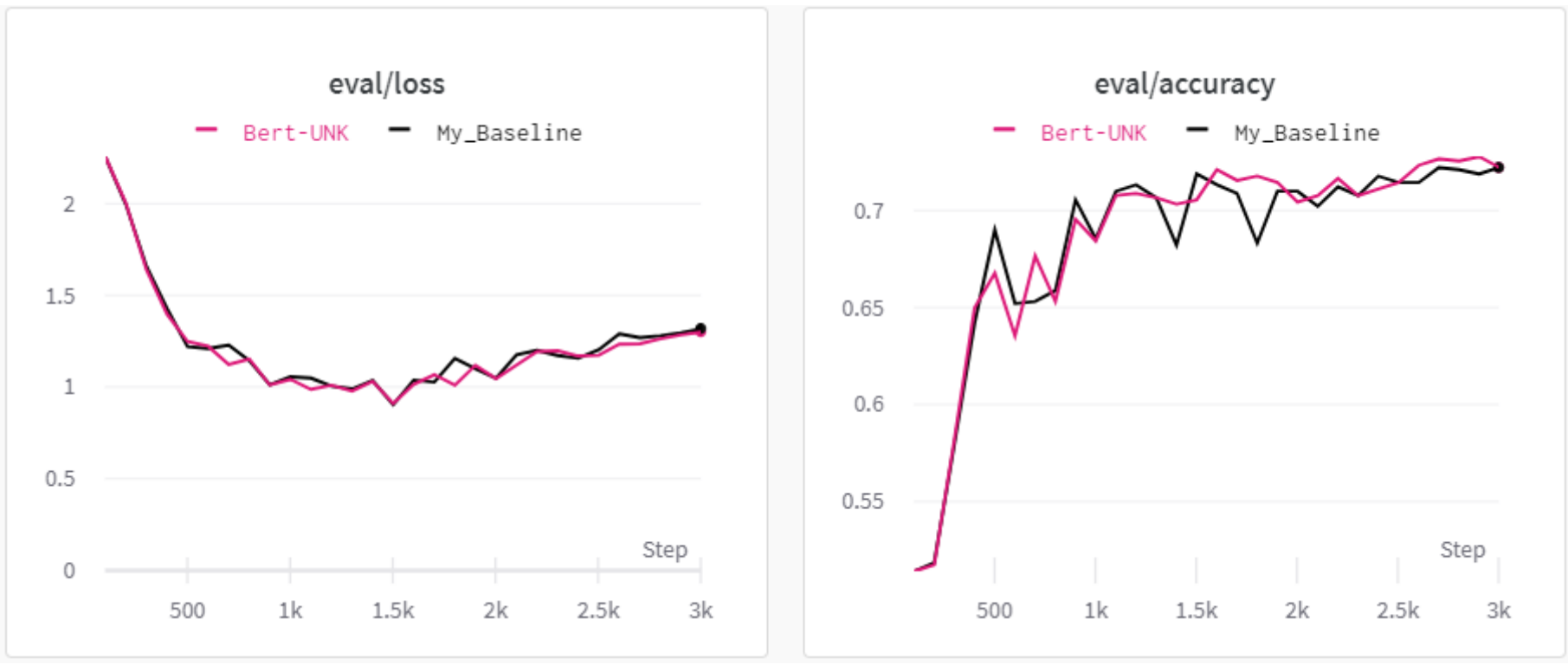
Entity Token 실험의 Graph

Competition간에는 시도하지 못했지만, Embedding Layer를 추가하거나 NER 정보를 Tagging하는 방법을 추가해볼 수 있었을 것같다.

Entity 내 Unknown Token 제거 실험

Entity 내에 Unknown Token이 있는 경우, 학습이 진행되어도 Model이 해당 Entity의 의미를 이해할 수 없을 것이라 생각했다. 이에, Entity 내에서 등장하는 Unknown Token을 미리 식별하고 Vocab에 추가하는 실험을 진행했다.

BERT와 같은 Model에서는 성능향상을 얻을 수 있었다. 약 100개에 가까운 Token이 식별되었고, 학습의 과정도 매우 매끄러워지는 효과를 얻었다.



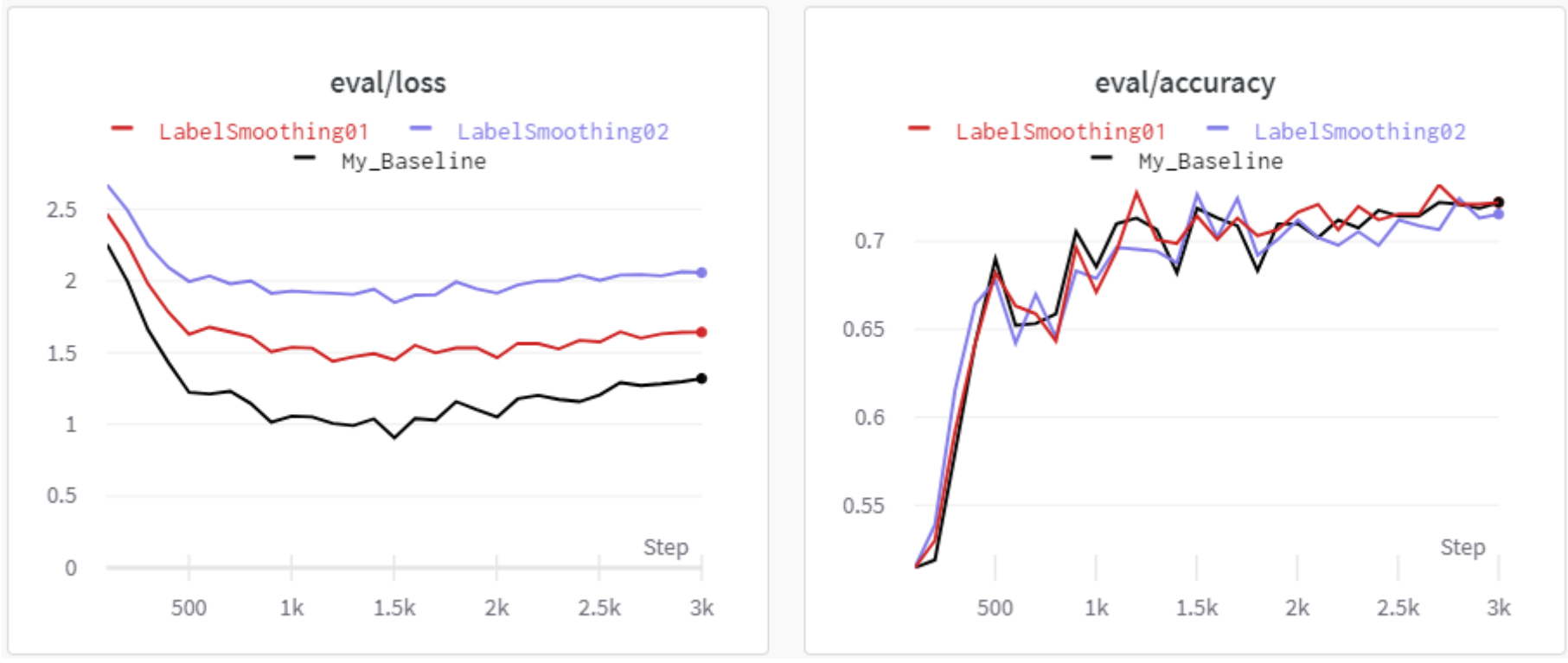
BERT 기준 학습 Graph

하지만, 최종적으로 사용한 Backbone Model에서는 이와같은 효과를 얻을 수 없었다. RoBERTa Tokenizer에서는 Unknown Token으로 식별되는 Token이 4개에 불과했으며, 이를 추가할 경우 오히려 학습의 그래프를 불안정하게 만들었다.
결론적으로 이 실험을 통해 최종 성능 향상을 얻을 수는 없었다.

Label Smoothing Loss 실험

Task에서 주어진 각 Class가 완전히 무관하지 않고, 유사한 의미를 갖는 Class가 많았기에 Label Smoothing Loss가 유의미할 것이라 판단했다.

Baseline에서 사용하는 Trainer 객체에 label_smoothing_factor를 활용해 이를 실험했고, 다음과 같은 결과를 얻을 수 있었다.

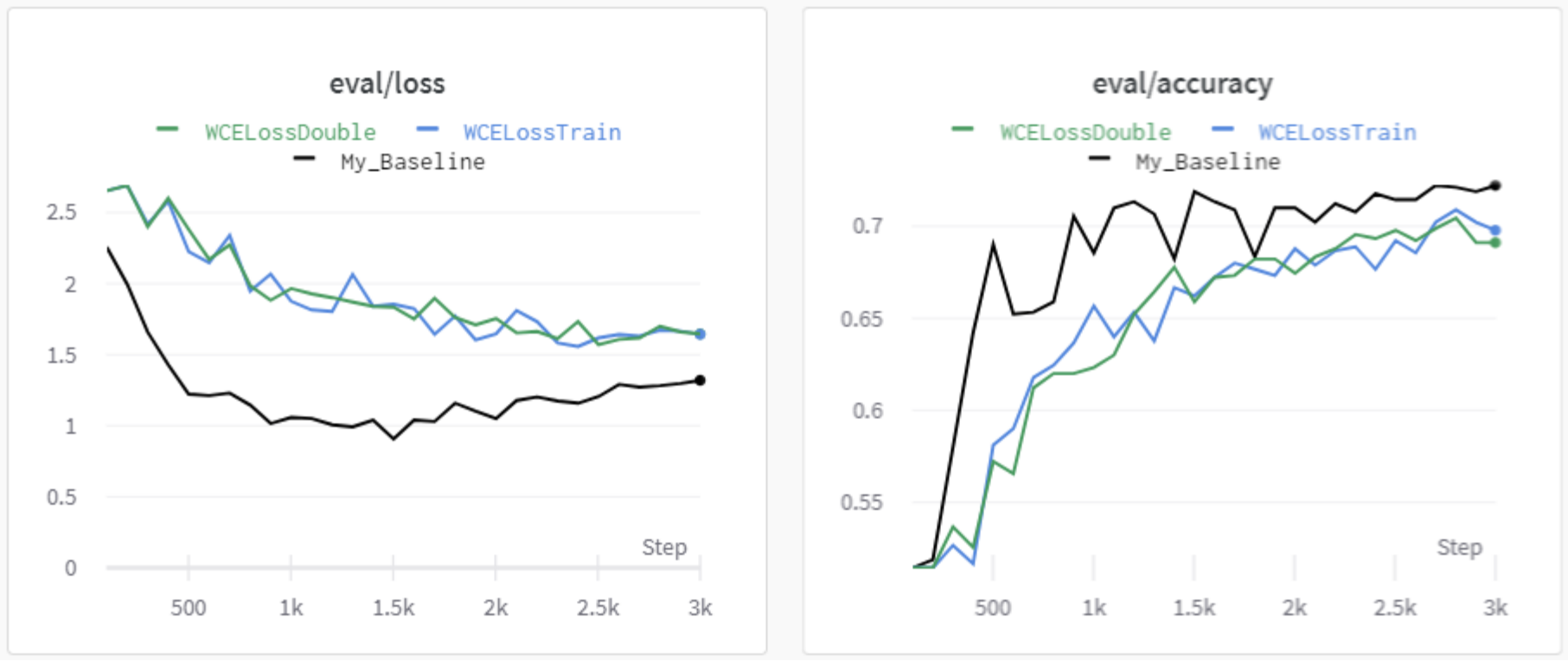


Label Smoothing Factor에 따른 학습 Graph

실제로 0.1 정도에서 높은 성능이 나올 것으로 예상 되었으나, 실제 제출 결과는 상이했다.
이를 통해서 당장의 성능 개선은 이루지 못했지만, Validation Set 구성에 문제가 있음을 확인할 수 있었다.

Weighted Cross Entropy Loss 실험

위 실험들을 통해서, Train Data와 Test Data의 Class 분포가 큰 차이가 없다는 생각이 들었다.
이에, 오히려 많은 갯수를 가진 Class에 가중치를 준 채로 학습시키는 것이 성능에 도움이 될 것 같다는 Idea가 생겼다.
Label의 분포에 따라 Weighted하는 Cross Entropy Loss를 구성해 실험을 진행했다.



Weighted Cross Entropy Loss를 활용한 학습 Graph

Competition 당시에는 위 Graph를 보고, 성능 향상에 도움이 되지 않을 것이라고 판단했다.
Wrap-up Report를 쓰는 지금에서는 성능이 수렴하는 곳을 확인해보지 않았던 것이 아쉽게 느껴진다.

Backbone Model 실험

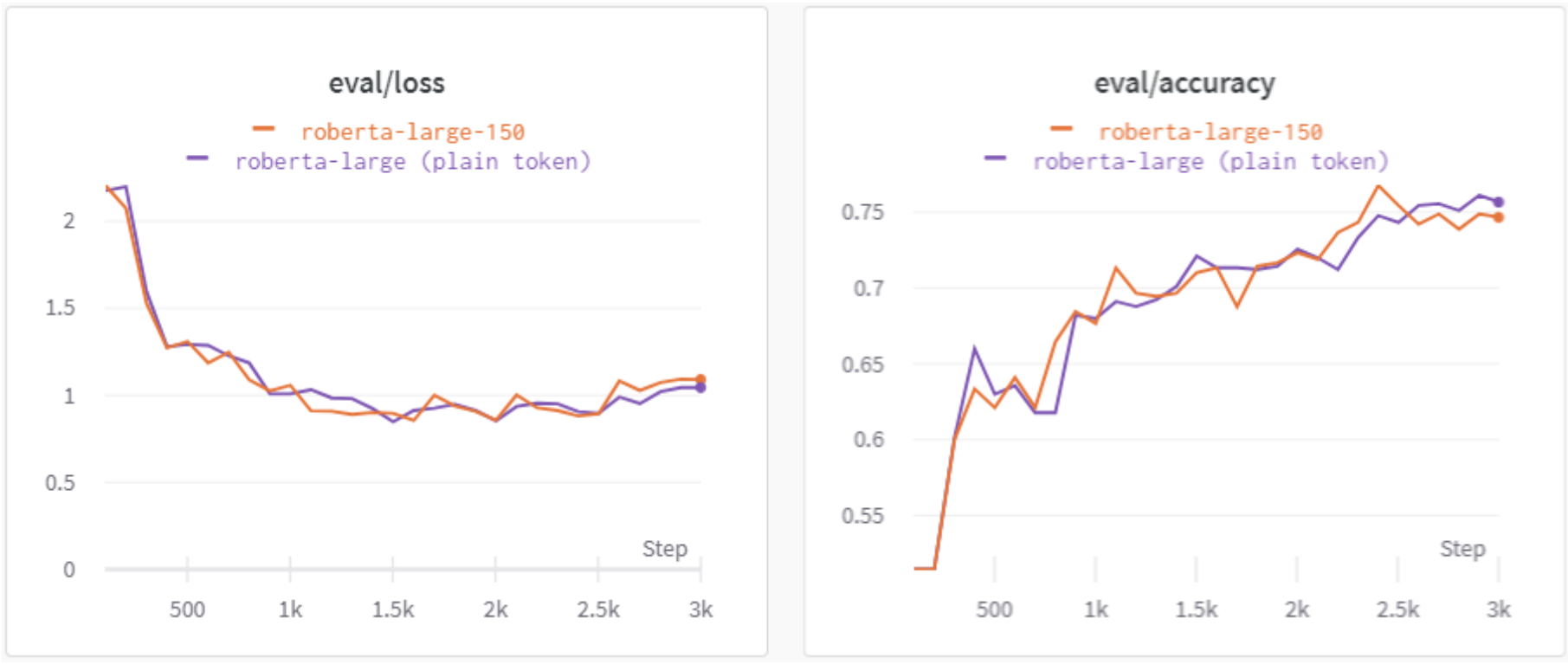
BERT 외의 다양한 Model의 성능을 확인하고자 했다. 실험 대상은 KoElectra, KoBERT, RoBERTa로 선정했다.

각 Model에 따른 성능 Graph

XLM-RoBERTa-large Model을 통해서, 77.1%의 성능을 달성할 수 있었다.
Model 변경 이후 Tokenizer의 변화에 따라 앞선 실험들을 다시 진행했다. 이 때, 위 실험에서 얻었던 결과와 상이한 결과를 얻었고, RoBERTa Model에 적합한 Option을 다시 찾아나서게 되었다.

Tokenizer Max Length 실험

EDA결과 Data의 90% 이상이 150 이상의 Length를 갖는 것으로 확인되었다. 기존에 사용하던 Max Length는 100이었으므로, 이에 대한 실험을 진행하게 되었다.



Max Length에 따른 학습 Graph

위 그래프 외에도 Max Length를 200으로 하여 실험해보았으나, 실질적은 성능향상을 얻을 수는 없었다.
문장이 잘리지 않은 채로 학습 되면 더 좋은 성능이 나올 것이라고 생각했으나, 결과는 반대였다.
실질적으로 Entity간의 관계를 확인하는 것에 있어서는, 문장 전체의 정보가 필요하지는 않다는 결론이 내려졌다.

Drop-out Rate 실험

비교적 적은 Data로 일반화 성능을 높이고자 다양한 Layer에 Dropout Rate를 변경해가며 실험을 진행했다.

각 Layer에 Dropout Rate를 변경한 학습 Graph

앞선 Competition에서는 늘 좋은 성능을 보였기에, 나름 만능이라고 생각했던 Option으로 여기고 있었다. 그래서 이번 실험 외에도 매우 수차례에 걸쳐서 계속 Dropout에 대한 실험을 진행했다. 결론적으로는 매번 성능 향상을 얻을 수 없었고, 오히려 시작이 부족해지는 경험을 했다.

외부 Data 실험

적은 Data를 보강할 방법을 찾던 중, 외부 Data를 정제한 토론 게시글을 찾을 수 있었다. 해당 Data를 활용해 실험을 진행해보았다.

외부 Data를 활용한 학습 Graph

기존의 9000개의 Data에 비해 압도적으로 많은, 23000개의 Data였기에 학습 자체에 매우 긴 시간이 소요되었다.
Data를 살펴보니 비슷한 형태의 Data가 매우 많은 것을 확인했고, 이 안에서도 Sampling이 필요하다는 것을 느꼈다. 다른 캠퍼들이 공유해준 Data도 마찬가지였다.
더 나아가서 캠퍼들이 수행한 Data 정제가 이번 Task에 적합하도록 정제되었다도 확신도 받을 수 없었다.

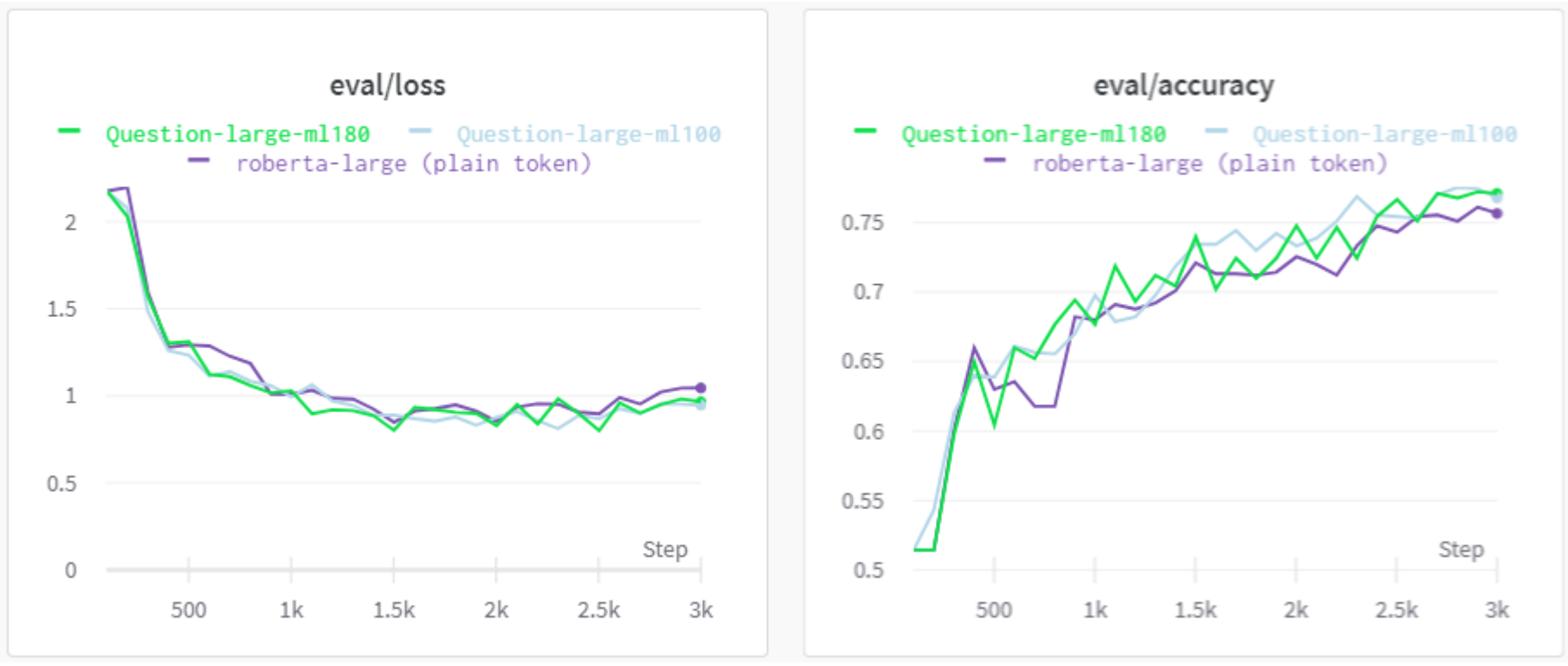
이에, Master께서 권유해주신 Kor-RE-Gold와 Silver Data를 직접 전처리하기 시작했다.
전처리 작업을 수행하다보니, 실제로 학습에 사용할 수 있는 Data가 현저히 적다는 것을 알게 되었다. 시간 대비 효율이 떨어지는 작업이라 생각해, 이 작업을 중단하게 되었다.

Question Type Input 실험

RoBERTa Model로 바꾼 이후 성능 개선에 큰 어려움을 겪고 있었다. Entity를 구분하던 구분자로 RoBERTa의 Special Token을 사용하니, 성능에 큰 하락이 있었기 때문이다.

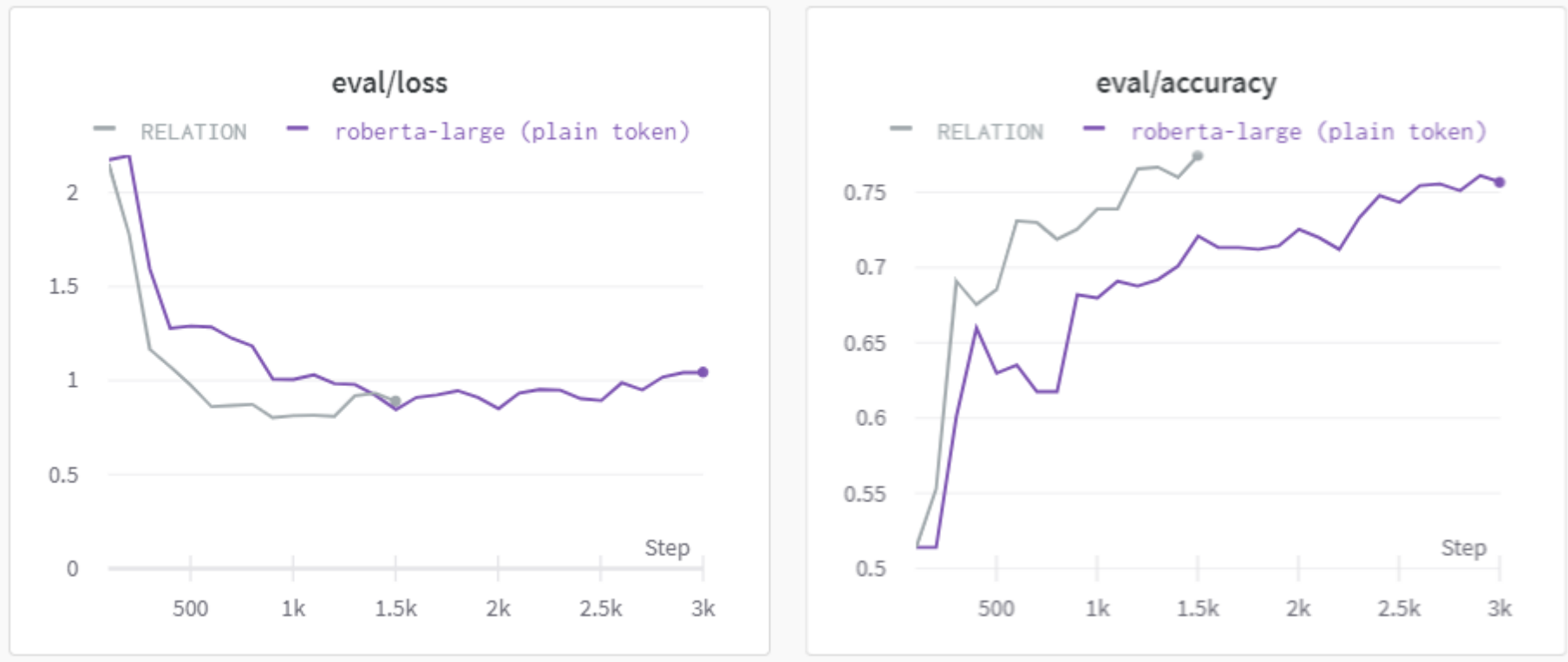
Entity간의 구분자로 `</s>`를 사용했을 때의 학습 Graph

이에, Query형태의 Input을 Question Type으로 변경해보는 실험을 진행했다. Input이 길어짐에 따라서 Max Length에 대한 실험도 동반되었다.



Question Type의 Input과 학습 Graph

이 실험을 통해 구분자로 Special Token을 사용할 때보다, Plain Text를 사용할 때 성능이 오른다는 것을 확인할 수 있었다.
또한, Input 길이가 길어짐에 따라 학습이 불안정해지는 것도 확인할 수 있었다.
이에, Query의 역할을 하는 Sentence를 최대한 간결하게 만들어서 학습시키는 방법을 강구하게 되었다.



구분자로 `RELATION`이라는 Plain Text를 활용한 학습 Graph

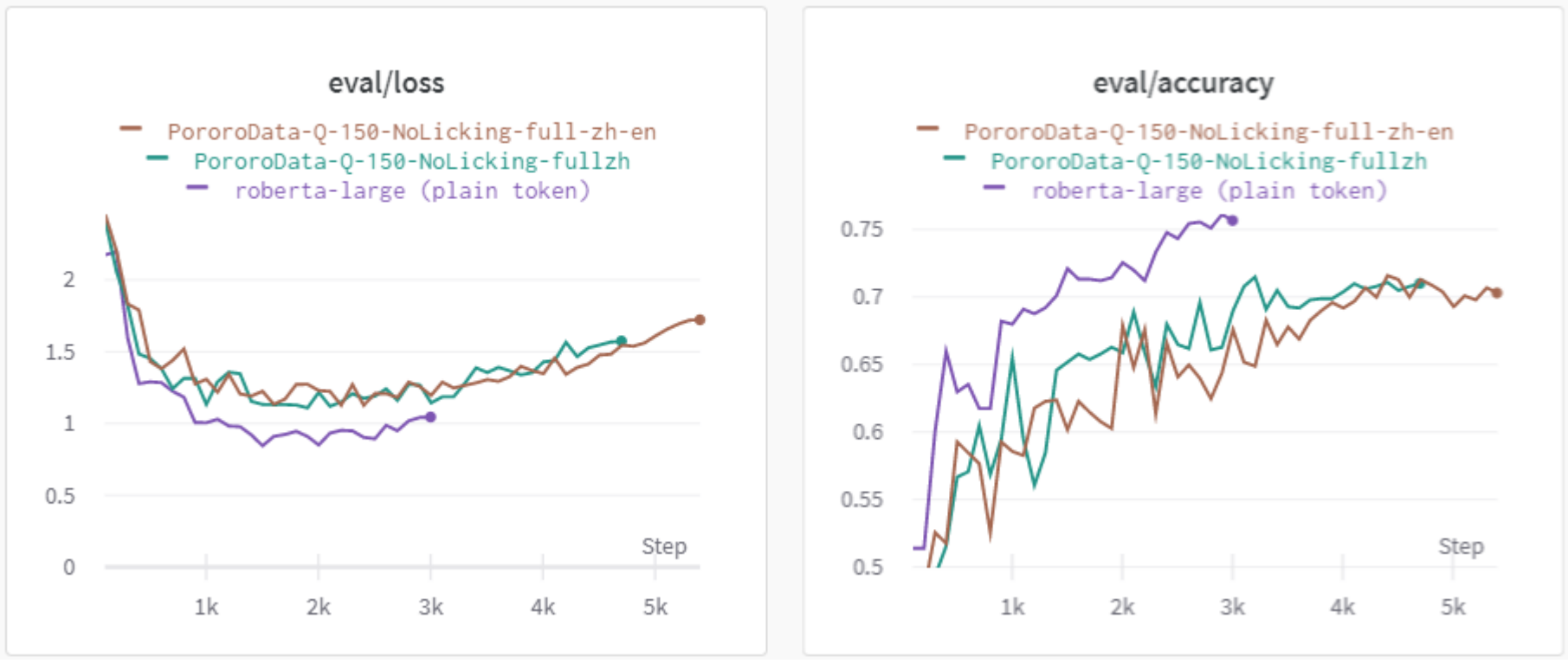
이 방법을 통해서 약 77.5%의 Score를 달성할 수 있었고, 이 Idea를 기본으로 채택하였다.

Back Translation Data 구축 및 실험

Task에 대한 신뢰성이 높은 Data를 어떻게 만들 수 있을까에 대해 고민하던 중, Back Translation에 대해 알게 되었다.
이를 활용하면 Text의 뉘앙스만 변경되기 때문에, Task에 적합한 Augmentation 방법이라고 생각했다.
PORORO Library를 통해 어떻게 하면 최대한 많은 양의 Data를 생성할 수 있을지 고민하고 실험했으며, 이를 실제 학습에 활용해보았다.

Back Translation Data를 구분없이 Validation Set으로 활용한 학습 Graph

위 방법으로 얻은 모든 Data를 추가하고, 구분 없이 Validation Set을 구성했더니 위와 같은 Graph를 얻었다. 하지만 실제 제출 결과는 위와 매우 상이했다.
즉, Back Translation된 Text를 통해 Data Licking이 일어났다고 판단할 수 있었다.
이에, Back Translation을 통해 추가 Data가 생성되지 않은 Text를 Validation Set으로 구성하여 학습을 진행해보았다.



Validation Set을 따로 구성한 학습 Graph

하지만, 위 방법으로도 성능 향상을 얻을 수는 없었다.
이 원인에 대해 생각해보자면, 동일한 Entity와 Label을 가진 Data를 여러번 학습했기 때문이라고 생각한다.

추후, 이 Data를 Data Augmentation처럼 활용하자는 Idea를 얻을수 있었다. 즉, 매 Epoch에서 동일 Index더라도 Back Translation 된 Data가 추출될 수 있도록 구성하는 것이다.

Data Augmentation 방식으로 Back Translation Data를 활용한 학습 Graph

이를 통해 79.5%의 Score를 달성할 수 있었고, 이를 기본 Idea로 활용하게 되었다.

Truncation 실험

Max Length 실험을 통해서, 이번 Task에 있어서는 전체 Sentence의 정보가 필요하지는 않다는 생각이 들었다.
이에, 각 Entity로부터 특정 거리만큼의 Token만을 활용하는 Idea를 적용해 실험해보았다.
실제로 일부 성능 향상을 얻을 수 있었지만, Back Translation Data는 Index 정보를 활용할 수 없어서 최종 Idea로 채택되지는 못했다.

Random Masking 실험

BERT Model의 Language Model Task와 유사하도록 구성한 Idea이다. Entity나 Special Token을 제외한 Token을 Random하게 `<mask>` Token으로 변경하는 것이다.
더 어려운 문제를 생성함으로써 Model이 더 잘 학습할 수 있으리라 예상했다.

Random Masking을 적용한 학습 Graph

실제로, 이는 성능 향상으로 이어졌고 이를 기본 Idea로 채택하여 활용했다.

Add Weight 실험

Class Imbalanced Data이기 때문에, 최종 Inference 시 특정 Class에 가중치를 추가하는 실험을 진행했다.
이는 학습단계에서는 활용하지 않고, 추론 단계에서만 사용했기에 Leader Board로 Score를 확인했다.

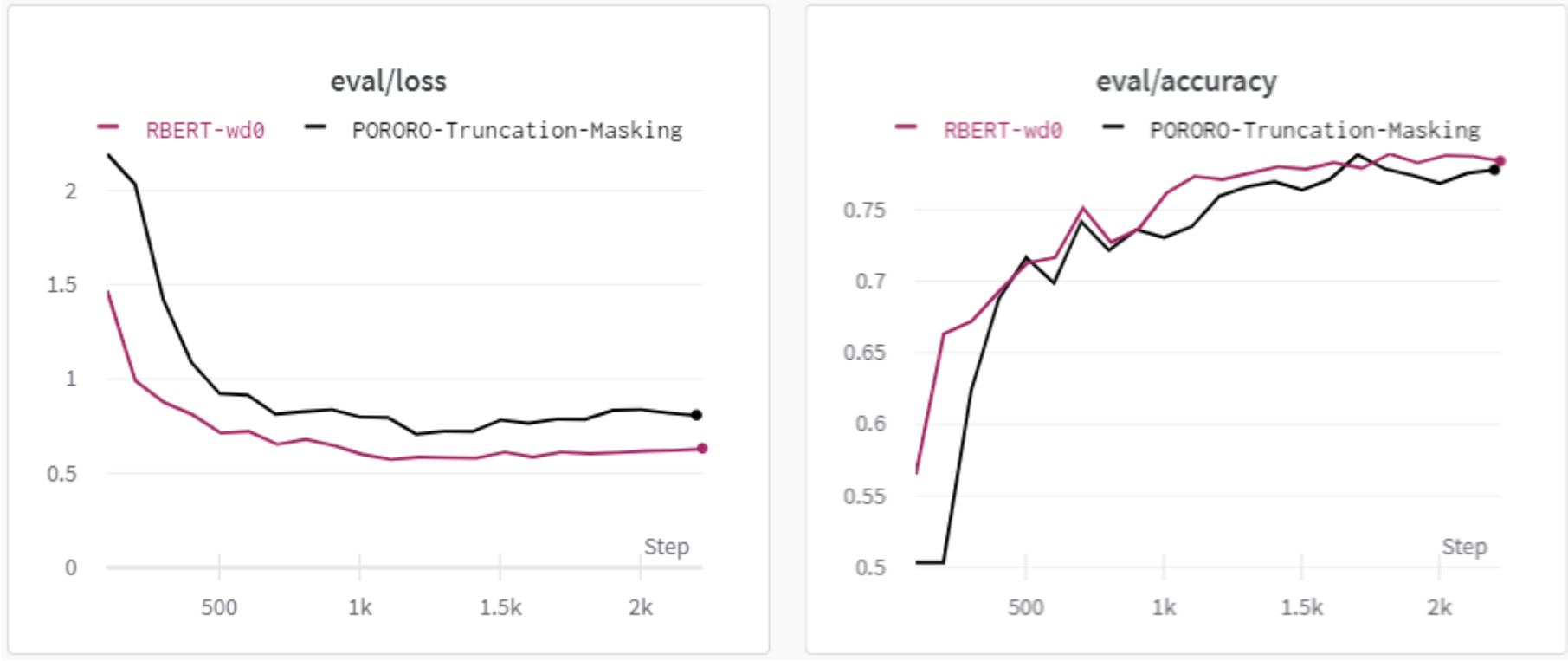
35	Matched PORORO Data, 5OOF, Truncation, Random Masking, Add weight 0.1	79.9000%
34	Matched PORORO Data, 5OOF, Truncation, Random Masking	79.6000%
36	Matched PORORO Data, 5OOF, Truncation, Random Masking, Add weight 0.2	79.6000%

Add Weight에 따른 성능 예시

Logit에 0.1의 가중치를 주었을 때 가장 높은 성능을 보였고, 이를 통해 79.9%의 성능을 달성할 수 있었다.

R-BERT Model 실험

전체 Text중에 한 개의 Token만을 사용하는 기존 Classifier를 개선할 수 있으리라 생각했다. Entity의 Embedding Vector를 직접 활용할 수 있는 R-BERT의 구조가 눈에 띄었고, 이를 실험해보았다.

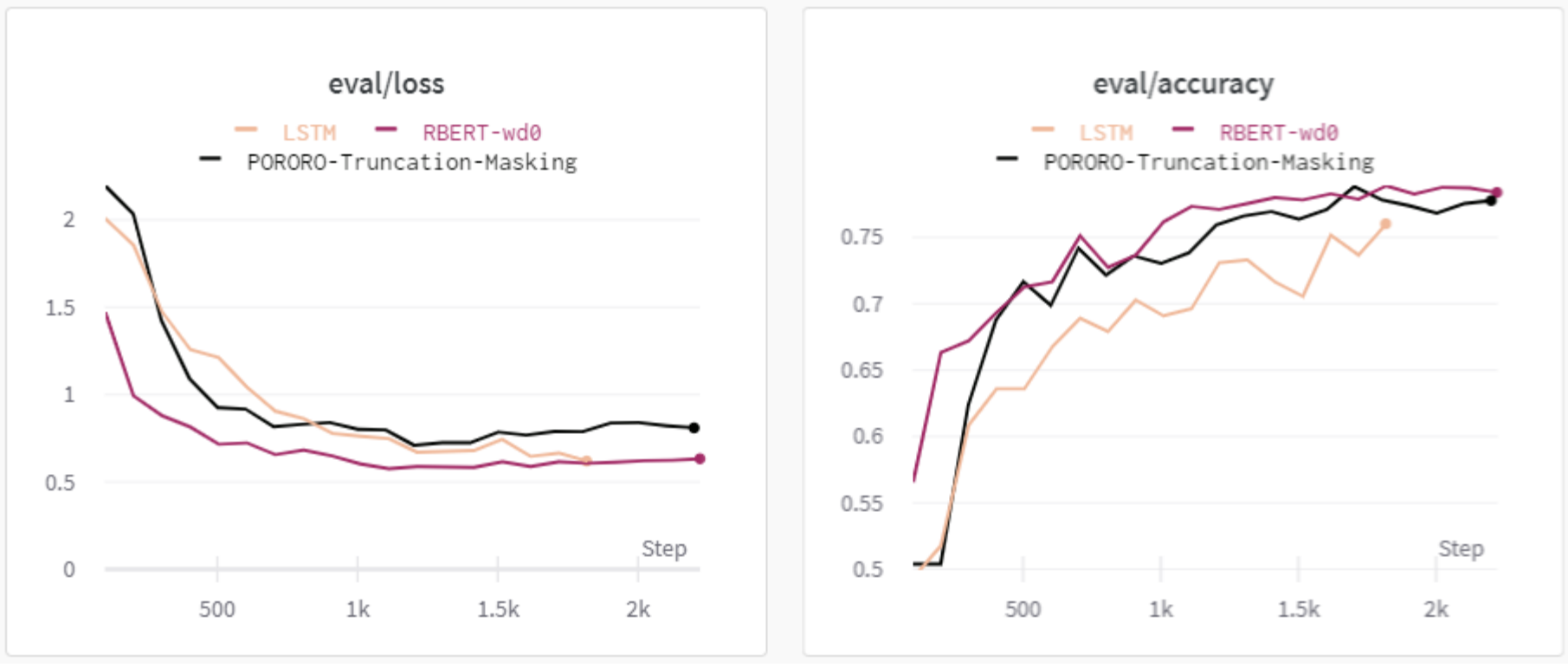


R-BERT 구조 Model의 학습 Graph

확실히 더 많은 Feature를 활용하는 만큼, 초반부터 안정적인 학습 Graph를 확인할 수 있었다.
기존의 Model과 함께 Ensemble하여 80.6%의 성능을 달성할 수 있었다.

LSTM Classifier 실험

최종 Submit을 위해 다양한 Architecture와의 Ensemble이 필요하다고 느껴졌고, 이에 LSTM을 Classifier로 붙인 Model을 실험했다.

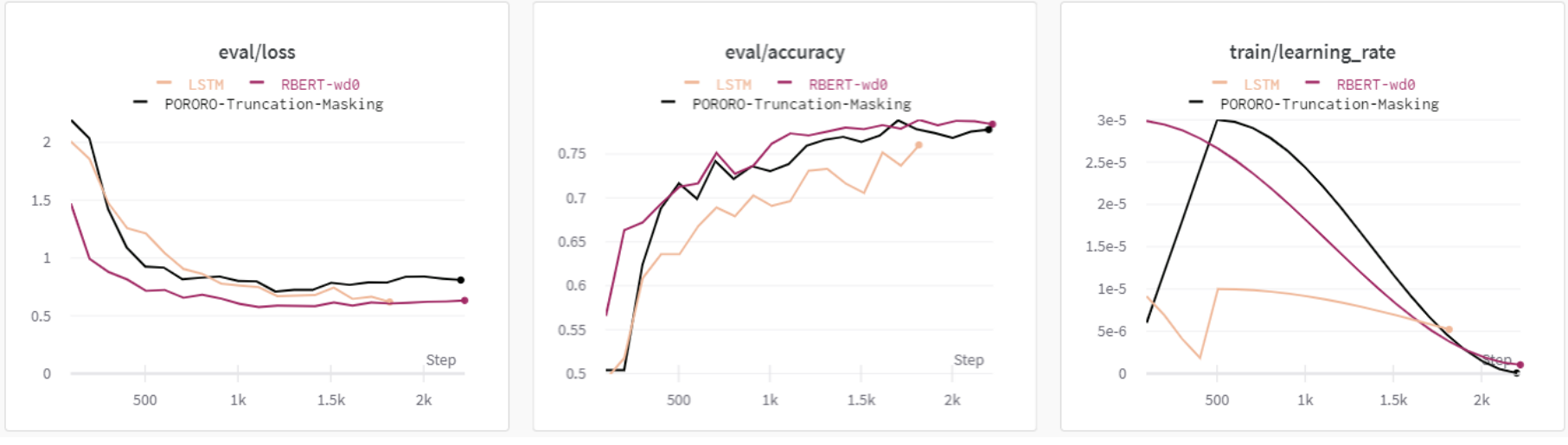


다른 Model과 LSTM Model의 학습 Graph

실제로 LSTM Model은 다른 Model에 비해서 성능은 떨어졌지만, 다른 Architecture인만큼 Ensemble에서는 매우 큰 효과가 있었다.
Soft Voting시에는 81.1%의 Score를, Hard Voting시에는 81.5%의 Score를 달성할 수 있었다.

LR, Batch Size, Scheduler 등 실험

최종적으로 선택된 Model들은 각각의 특성이 달라, 동일한 Hyper-parameter를 사용할 수 없었다. 이에, 각 Model별로 Hyper-parameter Search를 수행했다.



각 Model별로 사용된 LR과 Scheduling의 Graph

P2을 통해 얻은 교훈

개선해야 할 점

- Model Architecture를 직접 구현할 수 있는 실력을 길러야 하겠다.
- NER Tagging이나, Embedding Layer 추가 등의 실험을 진행하지 못했다.
- Generative Model을 활용해보지 못했다.
- 다양한 시도와 공부보다는 성능에만 연연했던 것 같다.

느낀 점

정말 좋은 사람들과 팀을 꾸렸기에 수행할 수 있던 Competition이었다.
무엇을 해도 성능 개선이 되지 않고, 뭘 해야할지 막막했던 순간에 이를 같이 고민해줄 사람이 곁에 있었다. 덕분에 끝까지 계속 나아갈 수 있었고, 결국 좋은 성적으로 마무리까지 할 수 있었다.
위에서 언급했던 모든 실험과 Idea가 모두 팀과 함께 만들어 낸 것이었다.
정말 마음 맞는 실력 좋은 사람들과 함께 할 수 있음에 감사했던 Competition이었다.

초반에는 정말 답답하고 막막했다. Stage의 설계 상 약 2주차부터 본격적으로 Competition에 참가하라고는 하셨지만, 눈앞의 Score를 못본 척 할 수는 없었다.
Task에 대한 이해도 부족하고, 도대체 뭘 어떻게 건드려야할지도 모르는 상황이 너무 힘들었다.
당장 Competition과 무관한 강의의 내용은 귀에 들어오지 않았고, 그러한 과제는 제 시간에 제출하지 못하기도 했다.
주어진 Baseline은 Hugging Face의 Library를 활용하는 형태로 되어있었고, 이를 이해하는데에도 오랜 시간이 걸렸다.
솔직히 힘들고 지쳤었다. 뭘 해낼 수 있으리란 생각도 들지 않았었다.
그래도 꾸준히 계속 뜯어보고, 공부하고, 강의를 듣다보니 조금씩 할 수 있는게 생겼다.
팀과 함께 내가 할 수 있는 것을 조금씩 해나가다보니, 점점 할 수 있는게 많아졌다.
이후에 또 엄청난 좌절을 겪기도 했다. 기존에 갱신했던 나의 Score를 넘기가 너무나도 어려웠다.
엄청 긴 시간을 들이고, 무조건 좋아질 것만 같았던 Idea들이 번번이 실패로 돌아갔었다. 무엇을 더 어떻게 해야할 지 전혀 갈피를 잡지 못했었다.
이 순간도 팀이 있어서 해결할 수 있었다. 내가 구상한 Idea보다 더 좋은 Idea가 공유되었고, 또 공유하는 과정에서 내 Idea도 개선되었다.
결국 폐기해야할 것 같았던 Idea도 최종 성능을 개선하는데 사용 될 수 있었다.

'인사가 만사'라더니, 정말 그랬다. 이후의 Stage에서도 이 팀과 함께 할 수 있음에 감사하다.

이번 Competition을 통해 얻은 큰 소득 중 하나는 Wandb이다.
이전에 Competition을 진행할 때에는, 별도의 실험관리 Tool 없이 진행했었다. 꼼꼼하게 정리를 하려고해도 매번 빈틈이 생겼다.
어떤 Option을 언제 적용 했는지 등은 정리 되더라도, 각 성능에 대해서 직관적으로 이해하기는 매우 어려웠다.
이번에는 Hugging Face를 공부하는 김에 아예 Wandb를 공부해 연동해봤다.
정말 정말 좋았다. 각 실험에 대한 관리부터, Graph를 통한 성능 비교까지 할 수 있었다. 왜 이제야 사용했나 싶을 정도로 좋았다.
이 덕분에 실험의 관리를 철저하게 할 수 있었고, 불필요한 시간을 단축시킬 수 있었다. 이번 성과의 주역이라고 생각해도 좋을 정도였다.
앞으로의 Competition에서도 이와 같은 Tool을 꼭 사용해야겠다고 느꼈다.