NLP 기초 프로젝트

-Semantic Text Similarity(STS)-

TEAM : 아기더키들**(NLP-09**조**)**

MEMBER : 곽희준**_T7308**

김민준**A_T7317**

김정은**_T7325**

박동혁**_T7335**

한동훈**_T7440**

[1. 프로젝트 개요**]**

프로젝트 주제	Semantic Text Similarity(STS): 두 텍스트가 얼마나 유사한지 판단하는 NLP Task
구현내용	두 문장을 입력으로 받았을 때 사람이 유사도(0.0~5.0)를 측정한 값을 예측하는 모델을 구현
개발 환경	**GPU**: Tesla V100 Linux server 4개 **Development Tool**: Git, Pandas, Hugging Face Transformers, Pytorch Lightning, WandB
협업 환경	**Github**: 코드 공유 **Slack**: 서버 구축 환경 설명 및 서버 사용 여부 정리 **Notion**: Idea Brainstorming 및 계획 정리 **Google Doc**: 실험 결과 공유 **Zoom**: 실시간 회의
프로젝트 구조	**BOOSTCAMP_PROJECT1-SEMANTIC_TEXT_SIMILARITY** ├──**data** # raw, preprocessed, augmented data + code └──**STS** ├──**baselines** # config.yaml (model hyperparameters) ├──**EDA** # .ipynb about EDA ├──**src** # model.py(+custom models), data_pipeline.py ├──**train.py** ├──**inference.py** # inference valid set ├──**ensemble.py** # model ensemble ├──**environment.yaml** # conda └──**wandb** # wandb sweep

[2. 프로젝트 팀 구성 및 역할**]**

대부분의 팀원들이 첫 NLP 도메인의 프로젝트인만큼 명확한 기준을 가지고 업무를 구분한 것보다 다양한 인사이트를 기르기 위해 데이터 전처리부터 모델 튜닝까지 End-to-End 로 경험하는 것을 목표로 하여 협업을 진행했다.

이름	역할
김민준	**EDA**, **Data Augmentation**(swap sentence, copy sentence, under-sampling), **Ensemble** (soft-voting, weight voting)
김정은	**EDA**(통계 분석, 문장데이터 분석), **Preprocessing**(특수/중복문자 제거), **Augmentation**(synonym replacement, swap sentence, under sampling, copy sentence), **Modeling**(large/small 모델 비교, hyperparameter tuning), **Ensemble** (bagging, boosting, soft voting, weight voting)
곽희준	**Env settings**(server 4대-git-conda를 통한 실험 환경 구성), **EDA**(token분석, preprocessing 분석), **Preprocessing**(띄어쓰기, 중복문구제거, 맞춤법 교정), **Augmentation**(copy, swap sentence), **Modeling**(pretrained 모델 비교분석, layer customizing, freezing, hyperparams tuning), **Postprocessing**(prediction outputs 0.2 단위 tuning)
박동혁	**EDA**(라벨 및 소스에 따른 전체 데이터 영향 분석, UNK토큰 탐색 및 제거), **Model Tuning**(KrELECTA- discriminator,sBERT), **Modeling**(custom : weightedMSELoss 함수), **Ensemble**(SoftVoting코드와 앙상블전처리 함수 연결)
한동훈	**EDA**, **Model tuning**(Head layer modification using cls and sep tokens) , **Data augmentation**, **Ensemble**(soft voting)

[3. 프로젝트 수행 절차 및 방법**]**

협업관련
● **Server**: 5명의 팀원이 4개의 서버를 공유하면서 작업 ● **Git Collaboration**: 각 팀원이 브랜치에서 독립적으로 작업 ● **Remote Repository**: 코드 관리를 하며 자유롭게 서버 이동 ● **Miniconda**: 환경 독립성과 이동성 확보(`environment.yaml` 활용)
프로젝트 주차 일정
● **1**주차: 개인 end-to-end 프로젝트 맛보기 ● **2**주차: 1주차 내용을 공유하고 협업하여 성능 높이기

[4. 프로젝트 수행 결과**]**

4.1 EDA

데이터에 대한 라벨의 분포와 토큰에 대한 분석을 진행하여 가설 설정을 통한 방향성을 결정하였다.

4.1.1 기본탐색

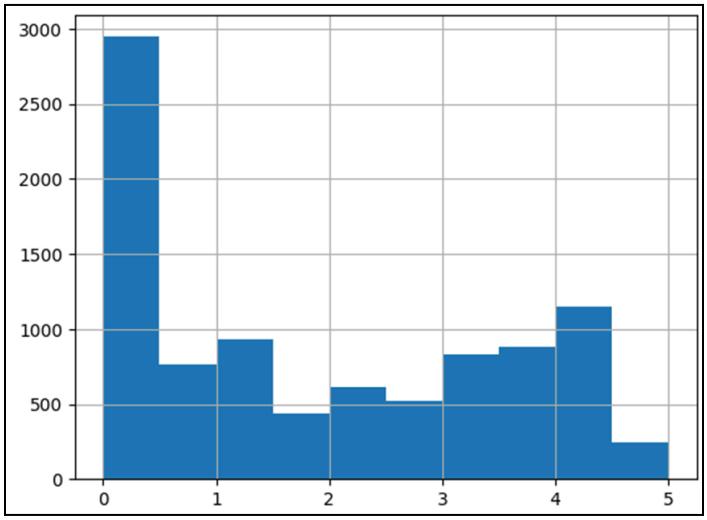
프로젝트에서 기본적으로 주어진 파일은 총 4개이다. 해당 내용은 다음과 같다. `train.csv` 파일은 9324개의 행을 가지고 있으며, `dev.csv` 파일은 550개의 행, `test.csv` 파일은 1100개의 행, `sample_submission.csv` 파일도 1100개의 행을 포함하고 있다. 특히, 다음의 사진과 같이 `train.csv`와 `dev.csv`의 label 컬럼은 0.0에서 5.0 사이의 실수 값으로 표현되어 있다.

id	source	sentence_1	sentence_2	label	binary-label
boostcamp-sts-v1-train-000	nsmc-sampled	스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요~	반전도 있고,사랑도 있고재미도있네요.	2.2	0.0
boostcamp-sts-v1-train-001	slack-rtt	앗 제가 접근권한이 없다고 뜹니다;;	오, 액세스 권한이 없다고 합니다.	4.2	1.0
boostcamp-sts-v1-train-002	petition-sampled	주택청약조건 변경해주세요.	주택청약 무주택기준 변경해주세요.	2.4	0.0
boostcamp-sts-v1-train-003	slack-sampled	입사후 처음 대면으로 만나 반가웠습니다.	화상으로만 보다가 리얼로 만나니 정말 반가웠습니다.	3.0	1.0
boostcamp-sts-v1-train-004	slack-sampled	뿌듯뿌듯 하네요!!	꼬옥 실제로 한번 뵈어요 뿌뿌뿌~!~!	0.0	0.0
boostcamp-sts-v1-train-005	nsmc-rtt	오마이가뜨지저스크라이스트퓻	오 마이 갓 지저스 스크론 이스트 팬	2.6	1.0
boostcamp-sts-v1-train-006	slack-rtt	전 암만 찍어도 까만 하늘.. ㅠㅠ	암만 찍어도 하늘은 까맣다.. ㅠㅠ	3.6	1.0
boostcamp-sts-v1-train-007	nsmc-sampled	이렇게 귀여운 쥐들은 처음이네요.ㅎㅎㅎ	이렇게 지겨운 공포영화는 처음..	0.6	0.0
boostcamp-sts-v1-train-008	petition-sampled	미세먼지 해결이 가장 시급한 문제입니다!	가장 시급한 것이 신생아실 관리입니다!!!	0.4	0.0
boostcamp-sts-v1-train-009	petition-sampled	크림하우스 환불조치해주세요.	크림하우스 환불조치할 수 있도록해주세여	4.2	1.0
boostcamp-sts-v1-train-010	slack-rtt	그 책부터 언능 꺼내봐야 겠어요!	책에서 꺼내야겠어요!	2.4	0.0

모든 파일의 데이터 품질을 확인해본 결과 결측치와 중복 행은 발견되지 않아고 이는 데이터 무결성이 잘 유지되어 있음을 보여준다. 조금더 깊게 문장들을 살펴본 결과, 사용될 문장들은 NSMC, slack, 국민청원과 같이 다양한 출처에서 크롤링되었으며 원본은 'sampled', 이를 영문 번역하여 다시 한글로 번역한 것은 'rtt'로 구분되어진다. 해당 출처들의 특성상 비격식체들이 많음이 보인다.

추가적으로 수치형 column인 label에서 데이터 불균형이 있는지가 의문이 들어 다음의 라벨EDA를 진행하였다.

4.1.2 라벨EDA (구간별 라벨 개수)

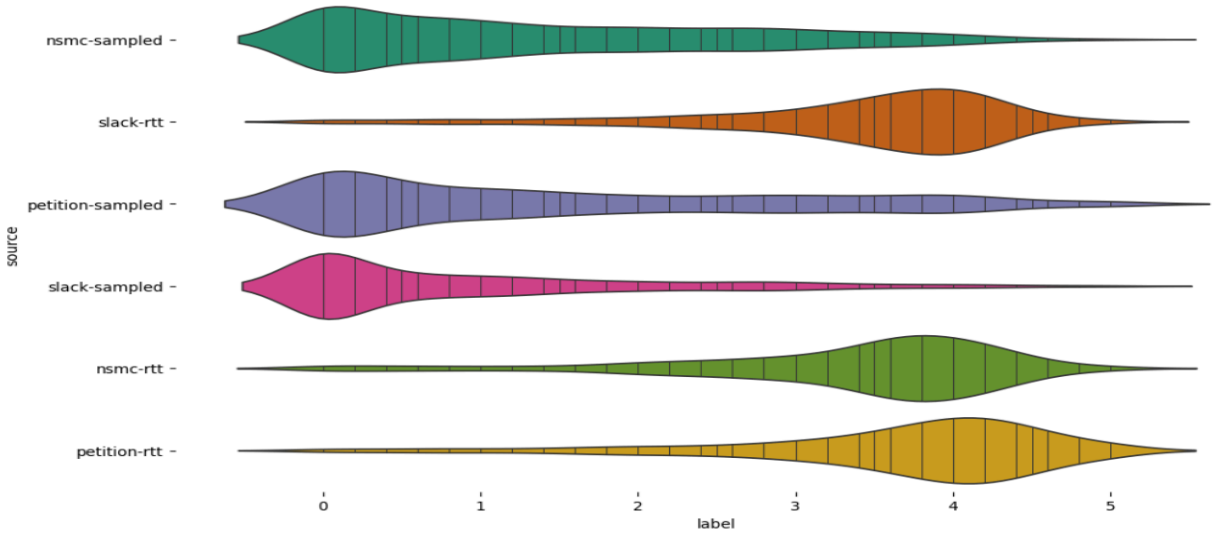


<그래프>는 train data set에서 우리의 목표인 label의 분포를 나타낸 것이다.가볍게 보아서도 알 수 있듯이 라벨별 데이터의 수의 불균형이 보인다. 이를 수치적 데이터와 함께 확인해본 결과는 다음과 같다.

먼저, 0.0~1.0구간의 라벨은 전체의 약 50%를 차지한다. 특히 0.0~0.5구간의 라벨은 약 30%이다.

반대로, 4.5~5.0 구간은 전체 라벨 수의 1.6%로 매우 낮은 수치를 보여 주고있다.

왜 이런 분포를 보이는 지에 대해 알아보고자 추가적으로 각 source column에 대한 라벨 분포를 확인해 보았다.



해당 자료에서는 크게 y축에 있는 sampled와 rtt에 초점을 맞춰 그래프를 읽어보았다. sampled의 라벨은 주로 0.0~1.0에 많이 분포되어 있는것을 볼 수 있고 rtt의 데이터 분포를 보면 3.0~4.5의 데이터 분포를 볼 수 있다. 추가로, 라벨간의 차이가 대부분 0.2의 단위로 분포되어 있다.

4.1.4 방향성**(**가설**)** 설정

지금까지의 내용을 토대로 각각의 processing, Augmentation, Modeling 단계에서의 목표를 정하고자 한다.

- Preprocessing

  인터넷 댓글의 데이터 특성상 사람이 판단하기에도 형식상 틀린 표현들이 존재함에 '반복적 글자' 처리와 문법적처리 방법을 도입하기로 계획하였다.

- Augmentation

  데이터의 불균형 문제 : 입력 데이터에서 label 0에 대한 편향이 나머지 label에 비하여 높은 상태이다. 이는 피어슨 상관 계수를 낮게 할 수 있는 요인으로 판단되었기에 모든 라벨을 Uniform Distribution에 근접하게 하기 위하여 다양한 oversampling, undersampling 방법들을 진행하기로 했다.

- Modeling

  입력 문장들이 한국말인 점과 인터넷의 비격식체가 입력 데이터이기에 해당 데이터로 Pretrain된 모델을 우선 순위 먼저 찾고자 한다.

4.2 Data Preprocessing

- EDA를 통해 문장이 정확히 일치하지 않아도 핵심 내용이 동일하면 **5.0** 점수를 부여하는 경향을 발견했다. 하지만 모델은 두 문장이 조금만 달라도 token이 달라지고 embedding 벡터가 달라져, 핵심 내용을 잘 이해하지 못하거나 예측 점수를 낮춰버릴 우려도 있었다. 이를 개선하기 위해 특수문자**,** 자음 반복**,** 띄어쓰기 오류 등 문장의 핵심 내용과 관계없는 부분을 제거하는 전처리를 진행했다. Raw data와 전처리한 데이터를 base model에 input data로 넣고 pearson 점수를 측정한 결과, 전처리한 데이터의 pearson 점수가 높게 나왔음을 확인했다.

  전처리 순서

  1. 느낌표, 물음표, 점, 자음, 모음, 특수문자, 공백, 반복표현의 반복 횟수 최대 2개
  2. 네이버 맞춤법 교정기로 오타, 띄어쓰기 교정

  전처리 예시**:**

변경전	변경후**(**중복**,** 맞춤법**)**
‘오오오오!!캬캬캬캬!!!!!!이거완전웃겨죽겠네ㅋㅋㅋㅋㅋ	‘오오!! 캬캬!! 이거 완전 웃겨죽겠네ㅋㅋ’

  결과 비교 **(klue/roberta-base,** 하이퍼파라미터 모두 동일**):**

Data	**Raw Data**	**Preprocessed Data**
Pearson Correlation	**0.7806**	**0.8129(+0.0323)**

4.3 Data Augmentation

- Under Sampling
  - 데이터셋의 분포를 uniform에 가깝게 하기 위해 많은 데이터 수를 가지고 있는 라벨 값들의 데이터들을 데이터 개수 제한을 두는 방식으로 제외시켜 Under Sampling 하였다.
- Over Sampling
  - Synonym Replacement
    - KAIST에서 만든 Korean WordNet(KWN) 을 사용하여 동의어를 교체한 문장 데이터 Augmentation 기법을 사용하였다. sentence에서 형태소를 분석하고 “sentence_1”과 “sentence_2”의 공통된 명사를 찾아내 wordnet.pickle 파일에 존재하는 동의어와 교체한 후 데이터를 증강하였다. 마지막에 한글 특성상 명사의 마지막 받침 유무에 따라 조사가 바뀌는 특성을 파악하여 그에 맞게 교체해주었다.
  - Swap Sentence
    - 데이터의 수가 9324개로 많지 않고 “sentence_1” 과 “sentence_2” swap 하면 positional embedding 값이 달라지기 때문에 유의미한 성능 향상을 만들 수 있을

것이라고 생각했다. 그래서, 전체 데이터셋에 sentence swap을 적용한 방법과 데이터셋이 “label” value 0점인 데이터가 많은 imbalanced data이기 때문에 “label” value 0점인 데이터를 제외하고 sentence swap을 적용한 방법들을 사용하여 비교하였다

- Copy Sentence
  - 기존 데이터셋에 "label" 값이 0.0인 데이터가 많아 불균형한 상태이기 때문에 “label” value 0.0인 데이터의 column “sentence_2”를 “sentence_1”으로 initialize 해서 다른 데이터에 비해 부족한 “label” value 5.0으로 변경하여 기존 데이터셋에 추가하는 방식으로 데이터를 증강하였다.
- Translation (ko2en)
  - 사용하는 모델과 토크나이저가 영어에도 어느정도 성능이 나오는 것으로 확인되어 부족한 label value의 데이터만 증강하였다

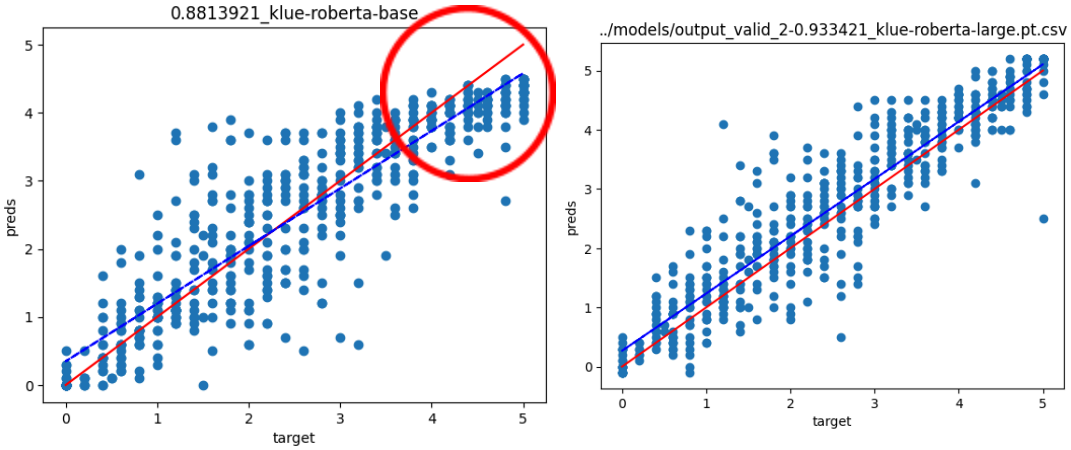
Undersampling 방식의 경우 모든 라벨을 엄격하게 uniform하게 만들기 위해 시도했던 방법인데, 결과가 안좋았다. 라벨의 분포를 valid set에 엄격하게 맞추기보다는 최대한 많은 데이터를 확보하는 게 더 중요하다고 결론을 내렸고 Synonym Replacement, Swap Sentence, Copy Sentence, Translation (ko2en)의 경우 각각 진행했을때 기본 train 데이터셋에 비해 성능 향상을 확인하여 여러 순서와 조합으로 실험을 진행하였다.

그리하여 최종적으로 확정된 것은 다음과 같다.

결과**: (klue/roberta-base,** 하이퍼파라미터 모두 동일**)**

Data	**Preprocessed**	**Preprocessed + Under Sampled**	**Preprocessed + Over Sampled**
Pearson Correlation	**0.8129**	**0.8137(+0.0008 from Preprocessed)**	**0.8526(+0.0397 from Preprocessed)**

- 추가적으로 모델의 라벨별 취약 구간을 확인한 결과, **target label 4.0~5.0** 구간에서 모델이 **target**보다 낮은 값으로 예측하는 경향이 있다는 것을 발견했다. 이를 개선하기 위해 데이터 증강**(data augmentation)** 과정에서 라벨 **5.0** 구간의 데이터를 추가하여 모델이 해당 구간에서 더 정확한 예측을 할 수 있도록 했다. 이후 재실험 결과, 모델이 5.0 구간을 훨씬 더 잘 예측하게 되었고, 성능이 눈에 띄게 상승했다.



x-axis: target (label)
y-axis: preds (Model Predictions)
Red line: True Regression Line
Blue line: Predicted Regression Line

4.4 Modeling

4.4.1 모델선정

EDA 결과, 데이터셋이 국민청원, 네이버 리뷰, Slack 등으로부터 왔고 비격식체가 많다는 사실을 발견했다. 이에 따라 비격식체 데이터로 학습된 **pre-trained** 모델을 찾기 위해 한국어가 가능하고, pre-train 당시 비격식체 데이터셋을 포함한 모델을 검색했다. 실험 결과, **klue/roberta, KoELECTRA-discriminator** 모델이 비격식체(예: Airbnb 글, 네이버 영화 리뷰)데이터셋을 포함해 pre-trained되었다는 정보를 확인했다. 그러나 모델을 서칭하는 방법에 익숙하지 않아 더 많은 모델을 찾지 못했고, 여러 모델의 특성(GPU 메모리, 출력 형태 등)을 충분히 파악하지 못해 모델 후보군의 폭이 좁았다는 사실이 아쉬웠다.

4.4.2 RoBERTa

4.4.2.1 RoBERTa model customizing

Pretrained 모델에 **Bidirectional LSTM**과 **GRU**를 추가하여 **Fine-tuning**을 하면 모델이 주어진 데이터셋에 대한 이해도를 높일 수 있을지에 대한 가능성을 탐구했다. 이를 위해 klue/roberta-base와 klue/roberta-large 모델의 뒷단에 Bidirectional LSTM과 GRU 레이어를 추가했다. 실험 방법은 pretrained 모델의 Transformer 출력에서 얻은 토큰 임베딩을 BiLSTM과 BiGRU를 통해 순차적 문맥 정보를 처리하고, Attention Layer로 시퀀스 내 각 hidden state에 가중치를 부여하여 중요한 정보를 강조한 후, 최종적으로 Fully Connected Layer에서 유사도를 예측하는 방식으로 진행했다.

Model: klue/roberta-base

Custom	**no custom**	**LSTM layer custom**	**GRU layer custom**
Pearson Correlation	**0.8708**	**0.8058**	**0.8150**

Model: klue/roberta-large

Custom	**no custom**	**LSTM layer custom**	**GRU layer custom**
Pearson Correlation	**0.8980**	**0.8902**	**0.8945**

실험 결과, base와 large 두 모델 다 기본 모델이 LSTM, GRU를 추가한 모델보다 더 좋은 성능을 보였다. 이는 Transformer가 이미 Self-Attention 메커니즘을 통해 문맥 정보와 관계를 잘 처리하고 있어, 추가적인 순차적 정보 학습이 오히려 학습 효율을 방해했을 가능성이 크다.

추가적으로, 모든 결과에서 large 모델이 base 모델의 성능보다 우수하다는 사실을 발견했다. 따라서 이후 실험들을 large 모델을 기반으로 지속했다.

결론적으로, LSTM/GRU 추가는 Self-Attention이 이미 충분히 문맥 정보를 잘 처리하는 상황에서 성능을 저하시켰다고 생각하여 모델 선정에서 제외했다.

4.4.2.2 RoBERTa model layer freezing

Pretrained 모델은 대규모 데이터셋을 통해 일반적인 언어 패턴을 학습하지만, 특정 task에 맞춘 fine-tuning이 필요하다. 특히 문장 유사성 예측(STS)과 같은 task는 특정 문맥과 패턴에 민감할 수 있어, Transformer 모델의 특정 레이어를 freeze하고 나머지 레이어를 fine-tuning하는 방식이 성능에 어떻게 영향을 미치는지 분석할 필요가 있었다. Transformer 모델에서 초기 레이어는 일반적인 언어 패턴을, 후반부 레이어는 task-specific 정보를 학습하는 경향이 있기 때문에, 어느 레이어까지 freeze할지에 따라 모델 성능이 크게 달라질 수 있다.

이를 검증하기 위해, klue/roberta-large 모델의 24개 레이어 중 freeze할 레이어 수를 조절(0, ~6, ~12, ~18 layers)한 후, valid set의 Pearson 계수를 비교하는 실험을 진행했다.

Model: klue/roberta-large

Frozen layers	**0**	**~6**	**~12**	**~18**
Pearson Correlation (Peak across epochs)	**0.9207 (0.9260)**	**0.9319 (0.9328)**	**0.9276 (0.9342)**	**0.9161 (0.9172)**

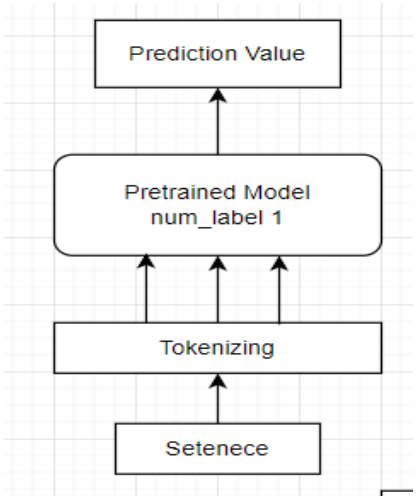
실험 결과, **0** 또는 **6**개 레이어를 freeze했을 때는 **Pearson** 값이 불안정하게 진동했고, **12**개 레이어를 freeze했을 때는 최적의 성능을 나타냈다. **18**개 레이어를 freeze한 경우에는 학습된 정보가 너무 제한되어 **Pearson** 값 고점이 낮게 측정되었다.

이를 평가한 결과, 0 또는 6개 레이어를 freeze할 때는 초기 레이어들이 학습 중에 지나치게 많은 가중치를 업데이트하여 불안정한 학습이 발생했다고 추정되며, 12개 레이어를 freeze했을 때는 모델의 기본 언어 이해 능력을 보존하면서 나머지 레이어만 fine-tuning함으로써 안정성과 task-specific한 미세 조정이 적절히 조합되어 최적의 성능을 낸 것으로 보인다. 반면, 18개 레이어를 freeze했을 때는 학습되는 레이어가 적어 모델이 새로운 데이터에 적응하는 능력이 감소하여, task-specific 학습이 제한되었기 때문에 **Pearson** 값 고점이 낮아졌다고 추정할 수 있다.

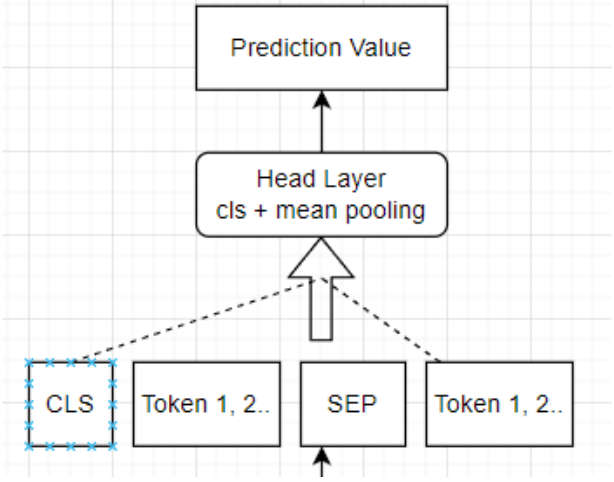
4.4.2.3 RoBERTa model

CLS 토큰이 문장의 전체 의미를 나타낸다는 학습을 기반으로, 다양한 토큰을 함께 사용할 경우 성능이 향상될 것이라는 가설을 세웠고, 이를 검증하기 위해 Pretrained 모델에서 num_labels를 수정하지 않고, 다양한 Hidden state를 활용하는 모델들을 비교 실험했다.

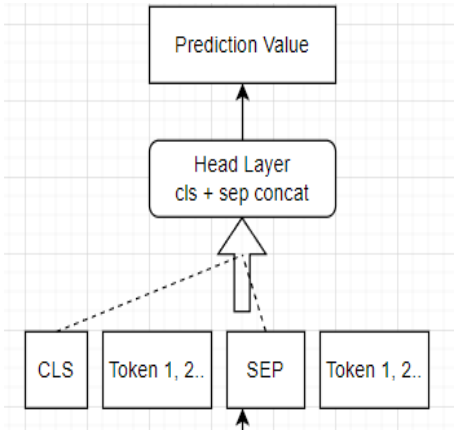
CLS 토큰과 SEP토큰을 단순 Concat한 layer를 추가한 모델, SEP 토큰 이후 토큰들을 mean pooling을 취하여 CLS 토큰과 Concat한 layer를 추가한 모델, CLS 토큰에 모든 hidden state의 토큰을 mean pooling한 layer를 추가한 모델을 비교한 결과, 세 가지 방식 간 성능 차이는 크지 않았으나, 기본 모델과 비교했을 때 5~10% 정도의 성능 향상이 있었다. 이는 데이터가 충분하지 않은 상황에서 다양한 토큰을 활용해 더 많은 정보를 제공한 것과, SEP 토큰이 두 문장의 구분을 명확히 하여 유사도 평가에 기여했기 때문으로 판단된다.



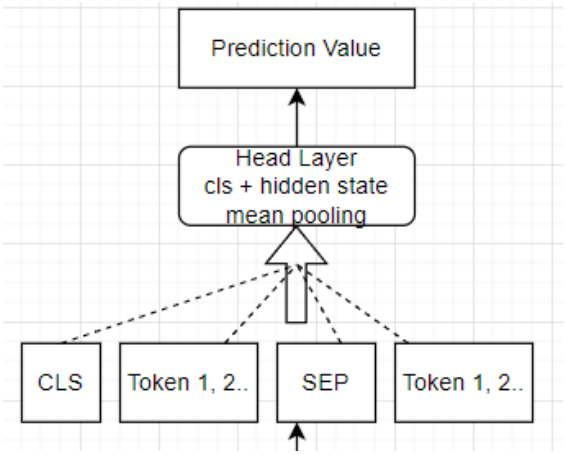
기본 모델



CLS + mean pooling(sep 이후 토큰)



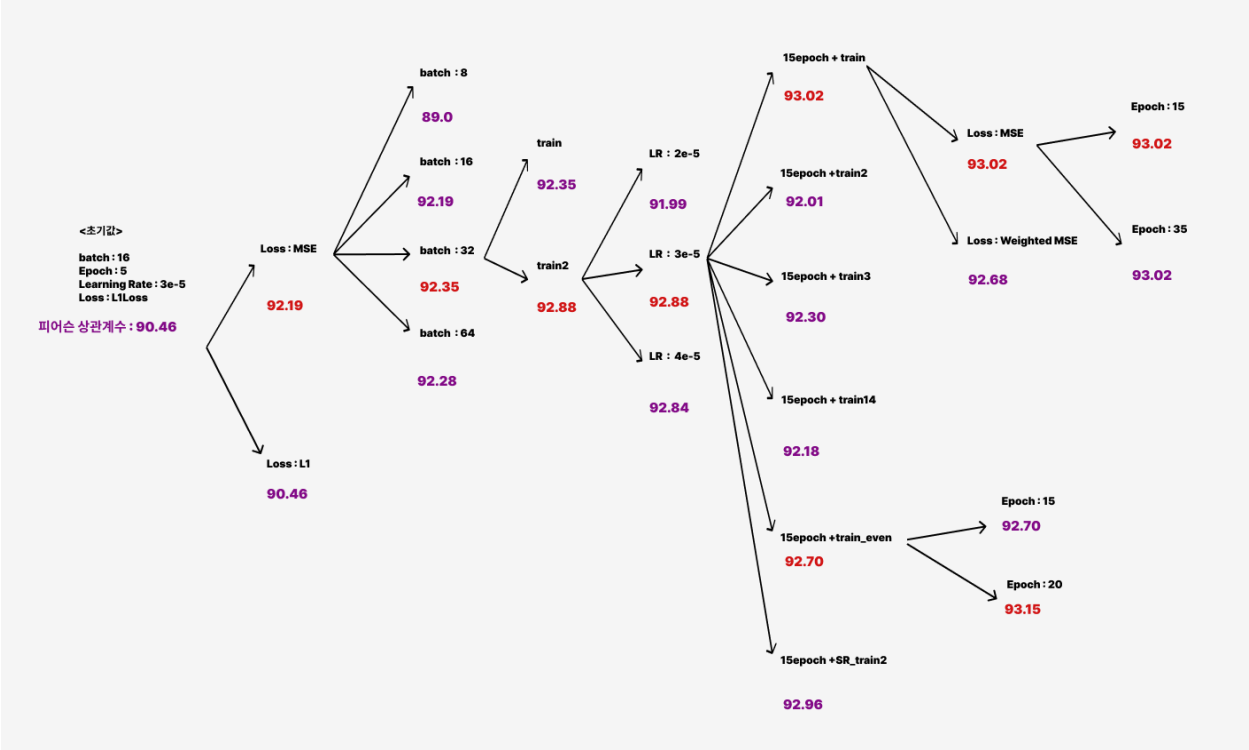
CLS + SEP concat



CLS + all hidden state mean pooling

4.4.3 KoELECTRA-discriminator : 하이퍼파라미터 튜닝

하이퍼파라미터는 피어슨 상관계수를 증가를 목표로 loss, batch, Learning Rate, Epoch순으로 찾아 나갔다. 그 결과 90.46에서 93.02으로 증가의 유의미한 하이퍼파라미터를 찾아냈다.



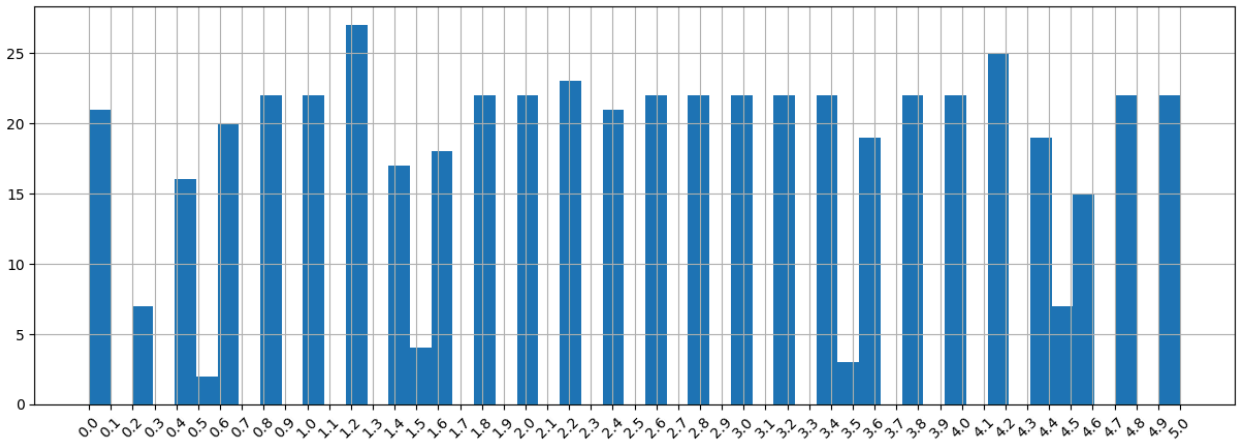
4.5 Ensemble

1. **soft voting** (일반 평균을 구하는 방식)
    - 평균의 함정을 우려하여 최소 5개 모델들의 결과값의 평균을 구해서 앙상블을 진행했고 개별 모델들의 Pearson Correlation 보다 높은 결과가 나왔다
2. **weight voting** (모델별로 가중치를 주는 방식)
    - 처음에는 모델들의 Pearson Correlation 값으로 가중치를 주는 방식을 고려했으나, 몇몇 모델들의 경우 Pearson Correlation 값과 리더보드에 올라가는 Public Score의 차이가 다른 모델들 보다 큰 것을 확인하여 weight voting은 soft voting을 통해 일반 평균을 구한 결과값 중 Public Score가 높았던 3개의 결과값에 대해 Public Score가 높았던 순서대로 0.5, 0.3, 0.2의 가중치를 주어 진행했다

시간 단축과 본 팀의 다양한 모델 결과물을 활용하기 위하여 위와 같은 Voting 방식들을 ensemble 방법으로 진행하였다. 1차적으로 soft voting으로 여러 결과물을 만들고 리더보드 상 좋은 결과물을 뽑아 nested weight voting 방법을 사용하여 최종 하나의 결과물을 만들었다. 그러나 weighted를 적용한 결과물은 1차적으로 soft를 적용한 결과보다 좋지 못한 성적이 나왔다.

4.6 Post-processing

EDA를 통해 전체 데이터셋의 라벨이 **0.2** 단위**(+0.5, 1.5, 3.5, 4.5** 라벨 데이터 소량**)**로 구성되어 있다는 사실을 발견했다. 그러나 모델은 0.0~5.0을 0.1단위로 모두 아우르는 예측을 했다.(하위 그래프 참고)



<<The label distribution of the valid set>>

따라서 **post-processing**을 통해 모델이 예측한 라벨이 정답 라벨에 존재하지 않을 경우, valid set의 label distribution을 기준으로 가장 가까운 라벨로 교정하는 방법을 적용했다.

1. 거리 계산: 각 라벨과 예측값의 차이(거리)는 다음과 같이 계산

$$d_i = |l_i - \hat{y}|$$

d_i: 라벨 l_i와 예측값 $\hat{y}$ 사이의 절대 거리
l_i: i번째 라벨
$\hat{y}$: 예측값

2. 가중치 거리 계산: 거리 d_i를 각 라벨의 빈도 c_i로 나눠, 가중치를 반영한 거리 w_i를 구함

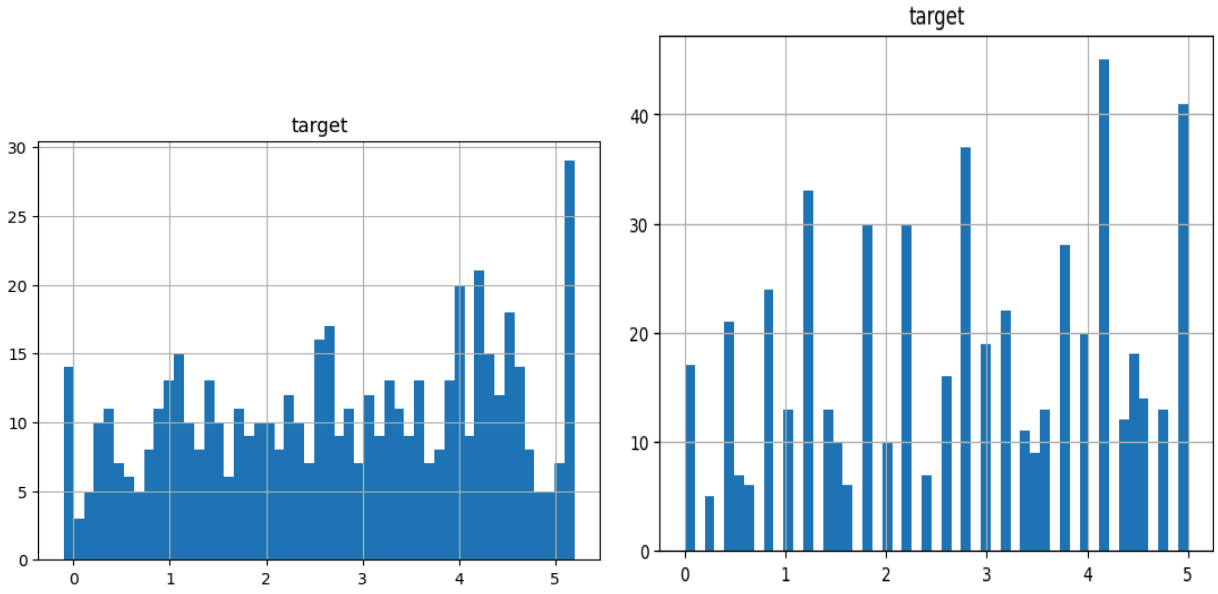
$$w_i = \frac{d_i}{c_i}$$

w_i: 가중치가 적용된 거리
c_i: 라벨 l_i의 빈도(등장 횟수)

3. 가장 가까운 라벨 선택: 가중치가 적용된 거리 중에서 가장 작은 값을 선택

$$\text{closest\_label} = \arg\min_i w_i$$

Post-processing 결과**:**



<Left: predictions **before post-processing**> <Right: predictions **after post-processing**>

실험 결과, **postprocessing**을 적용한 경우 성능이 향상되었으며, EDA에서 얻은 인사이트를 통해 간단하지만 효과적인 성능 개선을 이끌어낼 수 있었다.

[최종 리더보드 결과**]**

순위	분류	Pearson Correlation
12	Public Score	0.9296
7(🔺 5)	**Private Score**	**0.9387(+0.091)**

[5. 자체 평가 의견**]**

- 여건 상 시도하지 못한 내용
  - **Preprocessing**
    - n.5 label 값의 데이터가 적어 label값이 인접한 데이터 중 부족한 데이터로 clipping하고 데이터를 증강하지 못한 점
    - raw data에 <PERSON>, <ADDRESS> 이 들어있더라, 이는 보안때문에 일부러 교체한 단어일 것이고, Tokenizer가 해당 토큰을 잘 이해하도록 하나의 토큰으로 처리해보기
    - 데이터에 비격식체가 많은데, 모델 별로 tokenizing이 제대로 되는지(UNK 토큰 확인 등)를 확인하고 전처리하면 모델이 더 좋은 성능을 냈을 것 같다.
  - **Data augmentation**
    - sentence 안에 임의의 단어를 추가하는 방식인 Random Insertion과 sentence 안에 임의의 단어를 삭제하는 방식인 Random Deletion도 고려하였으나 문장의 의미가 달라질 수도 있다고 판단하고 진행하지 않았다
    - GPT에 부족한 label의 데이터를 100 문장씩 넣어서 증강하기
    - Copy를 통한 증강에서 label값을 크게 신경쓰지 않고 0인 데이터로 증강시켰는데 오히려 인접한 label값의 데이터나 모든 label값의 데이터를 균등하게 가져와서 copy하는 등 다양한 방법을 시도하지 못한 점
  - **Modeling**
    - 0.0~5.0(class=0.2단위 실제 라벨 개수) 라벨을 가지는 classification model로 생각하여 모델 뒷 layer를 customizing
    - output이 1개인 classification 모델 뒷 layer를 regression으로 customizing
    - LLM(Decoder only models)로 유사도 측정 -> 좋은 성능의 LLM은 GPU memory(32GB V100)에 올라갈 수 없고, memory usage가 낮은 LLM모델을 올리면 그만큼 성능이 떨어진다고 생각하여 실험 철회했으나 실험에 추가해봤으면 좋았을 것 같다.
    - Peft를 활용한 LLM사용
- 프로젝트를 진행하며 아쉬웠던 점
  - 중간 기록의 필요성 **:** 본인의 실험 내용을 다른 사람이 이해하기 쉽도록 정리
  - 큰틀의 부재 **:** 초반에 실험을 체계적 분할, 인원 배치 중구난방
  - 환경 통일 **:** 개발 환경, 자동화 구축, 버전 관리 팀원들간 통일 등을 초반에 해놓았으면 좋았을 것 같다.
  - 데이터 분석 **:** 데이터를 눈과 시간을 투자하는 방식으로 분석하고 전처리하지 못 한것
  - 토크나이저 분석 **:** 자체 토큰을 추가하여 학습을 진행하지 못 한것
  - 다양한 **tool** 사용**:** Streamlit, transformer visualization tool 등 다양한 도구들을 사용해보지 못한 것

NLP 기초 프로젝트

-개인 회고-

TEAM : 아기더키들**(NLP-09**조**)**

MEMBER : 곽희준**_T7308**

김민준**A_T7317**

김정은**_T7325**

박동혁**_T7335**

한동훈**_T7440**

[곽희준_T7308]

- 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가**?**

전체적인 프로젝트의 진행 방향을 고민하며 팀원들에게 적절한 작업을 분배했다. 처음으로 **end-to-end**로 모델을 구축해보는 프로젝트였기 때문에, 프로젝트의 모든 과정을 이해하는 데 중점을 두었다. 실험 환경 세팅, **EDA**, 데이터 전처리 및 증강, 모델링, 하이퍼파라미터 튜닝, 후처리까지 전 과정에 참여하면서, **Linux**, **Pytorch Lightning**, **WandB**, **Hugging Face Transformers** 등의 다양한 도구들을 익히게 되었다.

- 나는 어떤 방식으로 모델을 개선했는가**?**

EDA 결과를 바탕으로, 모델이 데이터를 더 잘 이해할 수 있도록 전처리 함수를 만들었고, 데이터 불균형 문제를 해결하고 모델이 더 많은 데이터를 학습할 수 있도록 데이터 증강을 진행했다. 또한, **pretrained** 모델에 여러 레이어를 추가한 커스텀 모델과 **layer freezing** 기법을 적용한 모델을 실험했다. **Loss** 그래프를 분석하면서 하이퍼파라미터 튜닝을 진행했고, **EDA**에서 분석한 라벨 분포에 맞춰 후처리 함수를 적용하여 모델 성능을 개선했다. 이 과정에서 모델의 **Pearson** 상관계수를 높이는 데 성공했다.

- 내가 한 행동의 결과로 어떤 지점을 달성하고**,** 어떤 깨달음을 얻었는가**?**

모델을 개선한 결과, 단일 모델로 **private score** 최고 성능을 달성했다. 이 과정에서 모델을 전반적으로 구축하는 방법을 깨닫게 되었고, 모델 개발에 대한 막연한 두려움이 사라졌다. 그러나 프로젝트를 진행하면서 지식과 경험의 부족으로 성능 개선에 어려움을 느꼈고, 더 깊은 이해와 경험이 필요하다는 점을 실감했다. 이를 바탕으로, 더 나은 **AI** 개발자가 되기 위한 노력을 지속해야겠다는 다짐을 하게 되었다.

- 전과 비교하여 새롭게 시도한 변화는 무엇이고**,** 어떤 효과가 있었는가**?**

이번 프로젝트에서 팀장 역할을 맡게 되면서, 나의 스케줄링뿐만 아니라 전체 프로젝트의 방향성을 설정하고, 팀원들에게 적절하게 작업을 배분하며 결과물을 평가하는 역할도 수행해야 했다. 또한, 팀 분위기를 활기차고 열정적으로 유지하기 위해 모범을 보이며 쉼 없이 노력했다. 이 과정에서 팀원들의 강점과 약점을 파악해, 각자의 능력을 최대한 발휘할 수 있도록 세분화된 작업을 배분했다. 팀 내 소통을 강화해 프로젝트가 원활히 진행되도록 지원하고, 문제가 발생할 때는 빠르게 해결책을 제시했다. 이러한 노력 덕분에 프로젝트를 일정 내에 성공적으로 마무리할 수 있었고, 팀원들의 참여도와 성과도 크게 향상되었다. 리더십 역량을 기르며 팀을 이끄는 경험을 쌓았고, 프로젝트의 큰 그림을 보는 능력도 함께 발전시킬 수 있었던 값진 기회였다.

- 마주한 한계는 무엇이며**,** 아쉬웠던 점은 무엇인가**?**

이번 프로젝트에서 가장 큰 한계는 지식과 경험의 부족으로 인해 성능 개선에 어려움을 겪었다는 점이다. 특히, 하이퍼파라미터 튜닝과 데이터 증강 과정에서 여러 시도를 했지만, 최적의 결과를 얻기까지 많은 시행착오가 필요했다. 또한, 모델을 더욱 효과적으로 개선하기 위한 시간적 제약도 아쉬운 부분이었다. 모델 성능을 높이기 위한 다양한 방법들을 충분히 시도하지 못했고, 최신 기술이나 모델 구조에 대한 더 깊은 이해가 부족했다는 점도 한계로 다가왔다.

- 한계**/**교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가**?**

다음 프로젝트에서는 더 체계적인 학습과 지식 보완을 통해 프로젝트 진행 속도를 높이고, 모델 성능을 극대화하는 데 집중할 계획이다. 구체적으로는, 최신 연구 논문과 모델 구조를 지속적으로 학습하여 모델 개선에 더 효과적으로 접근할 것이다. 또 하이퍼파라미터 튜닝을 보다 과학적으로 접근해 볼 것이며, 데이터 증강과 모델 구조 개선에도 더 많은 시간을 투자해볼 생각이다. 마지막으로, 주어진 일정 관리를 철저히하여 다양한 실험을 충분히 수행할 것이다.

[김민준A_T7317]

EDA

- 주어진 데이터의 기본적인 설명, 결측치 유무 및 중복 데이터의 유무를 확인하였다.
- “label” 값별 분포를 확인한 결과, 데이터셋이 “label” 값이 0.0인 데이터가 다른 “label” 값을 가진 데이터들보다 훨씬 많은 불균형 데이터(imbalanced data)인 것을 확인하였다.

Data Augmentation

1. **Swap Sentence**
    - 데이터 수가 9324개로 많지 않기 때문에 “sentence_1”과 “sentence_2”를 swap한 데이터를 추가하면 성능 향상이 있을 것으로 예상하였다.
    - 또한, 데이터셋이 “label” 값이 0점인 데이터가 많은 불균형 데이터이기 때문에 “label” 값이 0.0점인 데이터를 제외하고 sentence swap을 적용해보았다.
2. **Copy Sentence**
    - “label” value 0.0인 데이터의 “sentence_2”를 “sentence_1”와 일치 시키고 다른 데이터에 비해 부족한 “label” 값 5.0으로 변경하였다.
    - copy하는 데이터의 수에 따라 또는 copy 하게 되는 기존 데이터를 제외하는지에 따라 성능 변화가 있을 것으로 예상하였다

위의 두 방법들과 다른 팀원들의 Data Augmentation 방법들을 다양한 순서와 조합으로 실험을 진행하며 각 “label” 들의 분포가 최대한 Uniform Distribution에 근접하게 만들려고 노력했다.

Ensemble

1. **soft voting**
    - 일반 평균을 구하는 방식
        - 평균의 함정을 우려하여 최소 5개 모델들의 결과값의 평균을 구해서 앙상블을 진행하였고, 개별 모델들의 pearson correlation 값보다 높은 결과가 나왔으나, 시간과 남은 제출 횟수가 많지 않아 여러 실험을 진행하지 못하여 아쉬웠다.
2. **weight voting**
    - 모델별로 가중치를 주는 방식
        - 처음에는 모델들의 pearson correlation 값으로 가중치를 주는 방식을 고려했으나, 몇몇 모델들의 경우 pearson correlation 값과 리더보드에 올라가는 public score의 차이가 다른 모델들 보다 큰 것을 확인하기도 하였고 제출 횟수가 얼마 남지 않아 weight voting은 soft voting을 통해 일반 평균을 구한 결과값 중 public score가 높았던 3개의 결과값에 대해 public score가 높았던 순서대로 0.5, 0.3, 0.2의 가중치를 주어 진행해 보았다.

한계/교훈 및 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 사실 이번 프로젝트를 진행하며 기록을 잘하지 못하였는데 다음 프로젝트에서는 기록을 잘하여 시간을 효율적으로 사용하고 팀원들과의 협업을 더 수월하게 할 수 있게 해야겠다.
- 강의를 우선적으로 다 듣고 프로젝트를 진행하기로 하여 본격적으로 프로젝트 시작 후 조급한 마음에 체력관리를 잘 하지 못했는데 초반 계획을 잘 짜서 진행해 보고 싶다.
- 첫 프로젝트이고 학습을 목표로 했기 때문에 모든 팀원들이 프로젝트의 end-to-end 과정을 빠짐없이 경험해보자고 했지만, 나같은 경우에는 Data Augmentation 실험 과정에 몰두하여 모델의 구조를 바꾸는 등 모델링 부분은 제대로 경험해보지 못한 것 같다.

[김정은**]**

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가**?**
- 학습목표 : 다양한 시도와 실험을 통해 STS 문제에 대한 모델 성능 개선

순서	시도한 것
STEP-1) EDA	- 통계 분석 - 문장데이터 분석
STEP-2) Preprocessing	- 특수/중복문자 제거
STEP-3) Augmentation	- Synonym Replacement - Swap Sentence - Under Sampling - Copy Sentence
STEP-4) Modeling	1. Large Model - klue/roberta-large - sentence-transformers/ paraphrase-multilingual-MiniLM-L12-v2 - snunlp/KR-ELECTRA-discriminator - xlm-roberta-large - beomi/KcELECTRA-base - monologg/koelectra-base-discriminator - kykim/electra-kor-base 2. Small Model - klue/roberta-small - klue/roberta-base - rurupang/bert-base-finetuned-sts - rurupang/roberta-base-finetuned-sts - kykim/electra-kor-base
STEP-5) Ensemble	- Bagging - Boosting - Soft Voting - Weight Voting

2. 나는 어떤 방식으로 모델을 개선했는가**?**

가설	방식
데이터의 올바른 Augmentation을 통해 데이터 양을 늘리고, 모델의 크기를 그에 맞춰 키우면 성능이 좋아진다.	- 편향을 제거하는 방식과 기존 문장 데이터와 label을 변형하지 않는 방식으로 Augmentation 진행 - 성능이 개선되는 방식만을 채택하면서 Augmetation을 맞친 후 다양한 모델에 적용

3. 내가 한 행동의 결과로 어떤 지점을 달성하고**,** 어떤 깨달음을 얻었는가**?**

- 문장 및 label 데이터가 같은 방식의 Augmentation은 모델 성능 개선에 큰 영향을 주진 않음
- train과 dev 데이터를 합치고 8:2 비율에 맞춰 split하는 과정은 과적합이 발생할 수 있음
- large 모델 자체의 성능은 큰 차이가 없으며 hyparameter 튜닝, ensemble 등의 기법으로 추가적인 성능 개선이 필요함
- 모델 train 단계에서 가장 크게 영향을 미치는 건 데이터의 양과 품질에 있음

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고**,** 어떤 효과가 있었는가**?**

이전 방식	새로운 시도	효과
혼자서 End-to-end 로 실험을 진행할 때에는 모델과 hyperparameter를 고정하고 데이터 Aungemtation 방식을 하나씩 적용하면서 성능을 확인	정확한 데이터의 Augmentation은 모델 성능이 확실히 개선된다는 가정 하에 데이터의 품질을 유지하면서 Augmentation을 할 수 있는 기법을 새롭게 고민하기 시작	실험을 사용하며 성능을 확인하는 것보다 효율적이면서도 확실한 성능 개선 효과를 확인할 수 있었음

5. 마주한 한계는 무엇이며**,** 아쉬웠던 점은 무엇인가**?**

아쉬운 점	개선 방향
train 데이터의 편향을 맞추는 Augmentation을 고민하느라 데이터 내의 이상치의 유무를 파악하지 못한 점	정확한 label의 데이터만 Preprocessing을 통해 얻어내고 해당 데이터만 Augmentation을 진행하면 더 결과가 좋지 않았을까?
train 데이터와 dev 데이터의 중복으로 Pearson Correlation이 거의 만점에 가까운 결과가 나온 점	train과 dev 데이터를 합치고 split하는 과정 중간에 Augmentation이 있었는데, Augmentation을 같은 문장 조합과 label이 아닌 완전히 다른 수준으로 진행했다면 유의미하지 않았을까?
Wandb sweep 등 시각화툴의 미사용으로 너무 많은 실험을 진행한 점	훈련시키는 모델에게 가장 적합한 Hyperparameter를 찾기 위해 매번 실험을 진행하였는데, 이를 시각화툴을 사용해 한번의 훈련으로 결과를 바로 찾아낼 수 있지 않았을까?
데이터를 하나하나 뜯어보며 EDA를 진행하지 못한 점	단기간 내에 좋은 성능을 내야하는 상황에서 데이터를 꼼꼼하게 보지 못했는데, 이를 진행하면서 노이즈를 제거하는 과정이 포함되었다면 큰 성능 개선을 볼 수 있지 않았을까?
모델 선정 기준을 단순히 성능이 검증된 큰 모델로만 생각한 점	우리 데이터셋의 양과 특징에 맞는 작은 모델로 성능을 끌어올릴 수 있는 방법을

	고민해보았다면 어떤 결과를 가져올 수 있을지?

6. 한계**/**교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가**?**

- 단순히 통계적 지표를 확인하는 EDA에서 더 나아가 데이터 하나하나 뜯어보며 이상치 제거 등의 Preprocessing 기법을 적용해보기
- 기존 데이터와는 다른 수준의 Augmentation 기법을 고민하고 적용해보기
- 주어는 데이터셋의 양에 맞는 작은 모델을 가지고 다양한 파인튜닝 기법을 적용해 Modeling 해보기
- Hyperparameter Tuning은 Wandb sweep을 통해 효율적으로 진행하기

[박동혁**_T7335]**

1) 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가**?**

항상 팀장의 역활을 해오다 보니 팀원으로서는 거의 처음으로 참여를 하게되었다. 팀장을 할때는 상대의 생각보다는 팀의 방향을 결정하기 위하여 내 생각을 많이 고집하였는데, 이번에 팀원으로서 활동을하며 다른 사람의 생각을 이해해보고 그것대로 따라가보는 것을 연습해 보았다. 이는 확실히 혼자 결정을 내려서 일을 진행할 때보다 더 많은 여유를 주었다.

2) 나는 어떤 방식으로 모델을 개선했는가**?**

먼저, 초기 하이퍼파라미터를 기준점 삼아서, 손실함수,옵티마이저, batch, 학습률, 전처리 데이터 셋 ,epoch 순으로 실험해보며 결과를 이끌어 냈다. 그 후, 예상되는 정답 회귀선과 예측 회귀선을 그려보고 분산이 높은 것으로 판단이되어 weighted MES Loss함수를 만들어 정답 회귀선에서 많이 떨어진 데이터를 더 학습시킬 수 있도록 만들었다.

3) 내가 한 행동의 결과로 어떤 지점을 달성하고**,** 어떤 깨달음을 얻었는가**?**

솔직히 예상은 했지만, 직접 하이퍼 파라미터를 찾는 것은 시간적으로 매우 비효율적이었다. 하지만 각 하이퍼파라미터들에 대한 감각을 느껴보고 싶었다. Public 점수에서도 좋은 성적을 내자, 팀원들도 본인이 찾은 값들을 신뢰하여 의지해주시 시작했다. 이렇게 직접 하이퍼 파라미터를 찾다보니 알게된것은 최적의 하이퍼 파라미터는 크게 어떤모델인가?와 어떤 데이터를 쓰는냐?에 따라 바뀐다는 것이었다.

4) 전과 비교하여 새롭게 시도한 변화는 무엇이고**,** 어떤 효과가 있었는가**?**

모델과 데이터의 특징 관계를 자세히 살펴보았다. 과거에는 한 모델에 집착하여 결과가 좋지 않아도 계속 시도한 경험이 있다. 하지만 이번에는 학습데이터에 네이버영화 감상 리뷰가 있고 또한 사용할 pretrained modeld 같은 출처의 데이터로 학습된 것을 확인하고 사용하였다. 그 결과 거의 대부분의 경우의 수에서 타 모델보다 성능이 좋았다.

5) 마주한 한계는 무엇이며**,** 아쉬웠던 점은 무엇인가**?**

손이 느리고 사소한 것에 집착을 하는 경향이 있어 진행이 느렸다. 또한 아직 모델을 레이어 층으로 분석해보는 것에 시간적 한계와 실력의 한계를 느꼈다. 팀 활동이자 성과가 조금이라도 나와야 했기에 중간중간에 실험이 막히면 왜 예상한 것과 다른 결과가 나왔을까 보다는 바로 다른 경우의 수를 실험해 본 점이 너무 아쉬웠다.

6) 한계**/**교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가**?**

제일 많이 느낀 것은 실력의 부족과 그로 인하여 발생된 시간의 한계였다. 실력이 부족한 만큼 다른 사람보다 한 발이라도 먼저 프로젝트를 시작해볼 생각이다. 또한 그로 인하여 생긴 시간에는 왜 예상과 다르게 나왔고 그 근거로 어떤 것을 해봐야 할까에 대한 답을 확실히 알아가고 싶다.

[한동훈_T7440]

- 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?
    - 학습 목표 : 강의에서 이론과 실습으로 배웠던 내용을 실제로 적용해보고 익숙해지기
    - EDA
        1. 시각화
            - pandas, matplotlib를 활용한 시각화
        2. 전처리
            - 0.2단위로 구분된 label값에 n.5의 데이터를 인접한 데이터 중 더 적은 수의 데이터로 값을 변경함 → 불균형한 학습데이터와 달리 dev데이터는 균등했으며 편향이 없도록 학습해야 하기 때문에
        3. 증강
            - 영어도 토크나이징이 다소 잘 작동하는 것을 확인하여 부족한 label의 데이터를 증강 → 좀 더 많은 데이터를 확실히 확인하고 결정했으면 좋았을 것
            - swap, copy
            - 단어 랜덤 삭제 → 미적용
            - 동의어 데이터 증강 → 미적용, api나 model을 쓰지말고 그냥 gpt에 100개씩 넣어서 증강시켰으면 어땠을까
    - 모델링
        1. 모델 선정
            - 유명한 모델, 비슷한 프로젝트에서 어느정도 성능이 보장된 모델, 학습 데이터가 밀접한 관계가 있어보이는 모델을 선정 - 1, 2는 좀 더 근거를 찾았어야 했다.
        2. 모델 분석 - 진행되지 않았음 → 논문을 읽었어야했나, 무엇을 분석할 수 있었을까?
        3. 모델 수정
            - head layer modify
    - 성능 평가 및 분석
        1. 시각화
            - tensor board
- 나는 어떤 방식으로 모델을 개선했는가?
    - 문득 cls토큰을 쓰는 이유는 무엇인지 관심을 갖게되었고 팀원의 발표에서 cls뿐 아니라 sep토큰도 사용하는 것이 성능향상에 영향을 주었다는 내용으로 head layer를 다양한 방식으로 수정했다.
- 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?
    - 데이터보단 모델에 좀 더 비중을 두어 시도를 했던 것같다. 하이퍼파라미터 튜닝 tool의 사용이 미흡한 것과 많은 시도를 해봤음에도 성능향상에 한계가 찾아온 것을 보고, 학습에 데이터와 모델이 미치는 영향이 3 : 7 정도로 생각했던 것이 6 : 4 정도로 느껴졌다.
    - tool을 이용한 객관적인 분석 하루가 체계적이지 못한 7일보다 훨씬 많은일을하는 것을 느꼈다.
- 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?
    - 주어진 데이터가 충분히 정제되어 있을거라고 판단하여 처음부터 닫힌사고로 진행했고, 기본 과제에 연장선의 느낌으로 처음을 시작한 것이 결국 아쉬움이 많은 결과를 냈던 것 같다.
    - 내가 직접 개발 환경, 자동화 세팅을 하지 않았더라도 당일 참가하지 못했던 점이 이후에 걸림돌이 되어 다양한 시도를 하지 못한 점이 아쉬웠다.
- 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?
    - 개발환경 세팅과 자동화는 누군가 하면 내가 하지 않아도 되겠다는 생각을 했었지만 이후 다양한 시도에 지장이 생기기 때문에 직접 해볼생각이다.
    - 결과를 많이 내는 것보다 결과를 낸 기록을 남기는 것이 더 중요하다고 느껴 다음부터는 기록을 중심으로 할 것이다. → 결론은 기록이 직접적인 연관이 있는 경우 당연히 낼 수 있는 것이며 나중에 다시 보게되면 또 다른 견과를 도출해 낼 수 있으며, 팀원의 생각도 들을 수 있기 때문에
    - 데이터를 전처리하는 것이 전부 컴퓨터가 해야한다는 생각에 직접 눈과 시간을 할애하는 방식을 취하지 않았는데 해당 방법을 사용한 조가 좋은 성능을 낸 것을 보고 이러한 방식도 고려해볼 것 - 하더라도 더 효율적인 방식을 생각해서 진행
    - 모든일에 근거를 찾고나서 행동한다면 개발의 속도가 더뎌질 것이다. 하지만 근거는 중요하기 때문에 앞으로의 행동수칙은 다음과 같다.
        1. 시도할 수 있는 방안과 그에 따른 근거 및 기대를 간단히 정리
        2. 해당 방안을 구체적으로 알아보고 실행
        3. 1과 2를 하면 더 이상 기댓값이 높은 방안이 나오지 않는 상황에서 다양한 방법을 시도할 때 성능이 낮아진 이유까지는 여유가 안될지라도 성능이 좋아진 상황에서는 반드시 분석하여 근거를 찾아낼 것
        4. 코드를 단순 복붙을 하지말고 좀 더 이해하고 현재 상황에 맞게 커스터마이징, 주석을 직접 달 것