



# Wrap-Up Report

## 1. 프로젝트 개요

### STS 성능 개선 솔루션 프로젝트

- 프로젝트 주제 : 문맥적 유사도 측정(STS)
- 프로젝트 개요
  - 두 문장의 문맥적 의미의 유사성을 판단하여 0.0~5.0 사이의 점수로 출력하는 모델을 설계하고 개선 방법을 적용하여 성능을 개선하고자 함
  - 현 시점까지 부스트캠프에서 배운 Transformer 기반의 사전 학습 모델, 학습방법, 데이터 전처리 등을 종합적으로 시도해보고자 하였음
- 활용 장비 및 재료
  - GPU : v100 \* 5
  - 협업 툴 : Github, Notion, Wandb
  - 개발 환경 : Ubuntu 18.04
  - 라이브러리 : torch 1.12.0, torchmetrics 0.10.0, wandb 0.13.4, sentence-transformers 2.2.2
- 프로젝트 구조 및 사용 데이터셋의 구조도

#### [프로젝트 구조도]

```
├─ code
│  ├── args.py
│  ├── inference.py
│  ├── requirements.txt
│  ├── sts
│  │   ├── dataloader.py
│  │   ├── metric.py
│  │   ├── model.py
│  │   └── utils.py
│  └── train.py
├─ notebooks
│  └── EDA.ipynb
├─ data
├─ saved_models
├─ submissions
└─ wandb_checkpoints
```

#### [데이터셋 구조도]

```
├─ id : 각 data의 고유 id
├─ source : data 출처
├─ sentence_1 : 유사도를 비교하는 첫 번째 문장
├─ sentence_2 : 유사도를 비교하는 두 번째 문장
├─ label : 유사도 점수 [0.0, 5.0]
└─ binary-label : 유사도 점수가 2점 이하인 경우 0, 3점 이상인 경우 1로 변환된 이진 데이터
```


	id	source	sentence_1	sentence_2	label	binary-label
0	boostcamp-sts-v1-dev-000	nsmc-sampled	역선은개를 총뿔뿔고 끝입니다	역선은 총내만 내고 그마저도 후반부에는 슬로우모션 처리	2.0	0.0
1	boostcamp-sts-v1-dev-001	slack-rtt	감격스러워 입학으심?	너무 감동해서 입 다물어?	3.4	1.0
2	boostcamp-sts-v1-dev-002	nsmc-rtt	이번 년도에 본 영화 중 가장 최악의 영화.....	올해 본 영화 중 최악...	4.0	1.0
3	boostcamp-sts-v1-dev-003	slack-rtt	특히 평소 뮤직채널에 많은 영감을 불러넣어주시는!	특히, 당신은 항상 많은 음악 채널에 영감을 줍니다!	3.4	1.0
4	boostcamp-sts-v1-dev-004	slack-sampled	다음 밍스레이지가 기대됩니다~ ♪	다음 후기도 기대됩니다~~	1.4	0.0

### • 기대효과

- 주어진 문장 쌍의 유사도를 0-5점 사이의 지표로 나타낼 수 있다.
- 약 2달동안 배운 내용들을 활용하여 프로젝트에 적용해본다.
- 부스트캠프 첫 협업 프로젝트를 경험해보며 다양한 협업 툴들을 사용해본다.

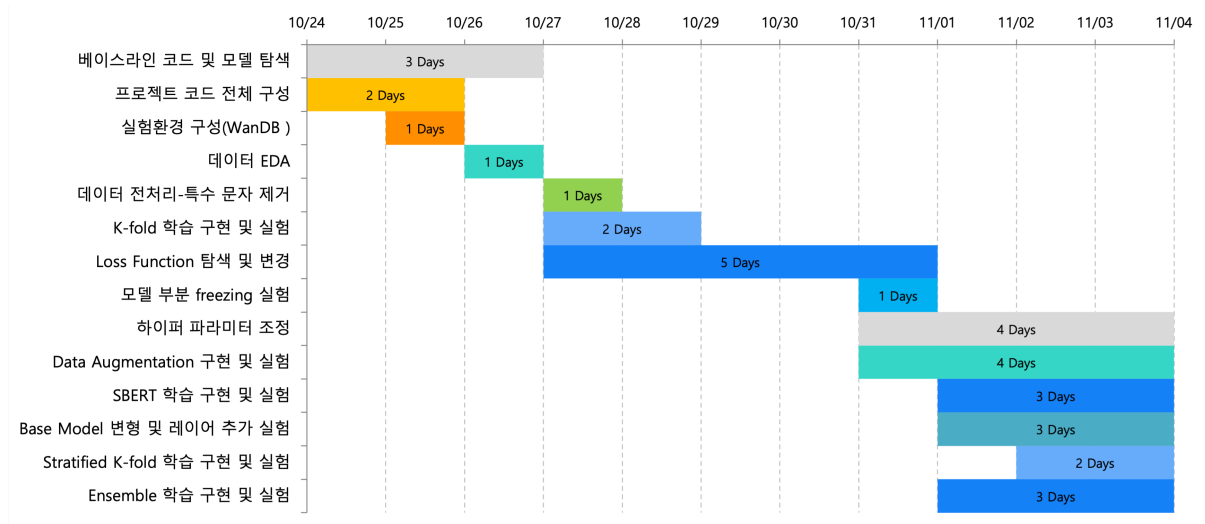
## 2. 프로젝트 팀 구성 및 역할

공통 : hyperparameter 조정 및 실험

- 김지수(팀장): Project Manager, 프로젝트 전체 구성, 학습 개선 팀 
  - 프로젝트 리드
  - 학습 개선: K-fold, 학습 분석 툴 wandb, 코드 리팩토링
- 김산(팀원): 데이터 팀, 모델 팀
  - 특수문자 제거 기능 구현
  - SBERT 학습 구현
- 박수현(팀원): 데이터 팀
  - 주어진 데이터에 대한 EDA 분석
  - retranslation을 통한 데이터 증강 및 증강 데이터 EDA
- 엄주언(팀원): 모델 팀, 학습 개선 팀
  - 데이터 : 주어진 데이터에 대한 EDA 분석
  - 모델 : Loss Function 변경 및 추가 , 모델 partial freezing
  - 학습 개선 : Ensemble voting, stacking, Stratified K-fold
- 현승엽(팀원): 모델 팀
  - Baseline 모델에 linear layer를 추가한 모형 구현

## 3. 프로젝트 수행 절차 및 방법

### 3.1 프로젝트 마일스톤



## 3.2 협업 방식

본 프로젝트는 10월 24일부터 11월 3일까지 주말을 포함한 11일간 수행하였으며 Notion을 이용하여 작업 상황을 공유하였으며 Wandb로 팀원간 공유한 학습 결과를 토대로 실험을 진행하였다. 중심이 되는 프로젝트 코드는 Github을 사용한 공유 프로젝트 형태로 수행하였다.

Github의 경우, 각자 담당한 기능에 대한 브랜치를 나누어 구현한 후에 확인 절차를 거쳐 메인 브랜치로 병합하여 문제가 발생하더라도 이전 버전에 접근하여 원인을 파악할 수 있도록 하였다. 또한 commit 작성 방식을 gitmoji 등으로 통일하여 관리하였으며 Github에서 제공하는 issue와 코드 리뷰 기능을 활용하여 프로젝트에 관련된 문제를 토론하고 해결하고자 하였다.

## 4. 프로젝트 수행 결과

### 4.1 전체 프로세스

#### EDA/preprocessing

##### 1. EDA

- 데이터 이상치 확인
  - 대회에서는 binary-label은 label이 2 이하이면 0, 3 이상이면 1로 분류된다고 했으나 실제 데이터에서는 label 값이 2.5 이상이면 1, 그 외에는 0으로 분류되는 것을 확인하였다.
  - 각 데이터셋의 평균과 분산을 확인한 결과 통계적으로 상이한 분포를 지니고 있음을 확인하였다.
  - 데이터 내 누락된 데이터(NaN)은 존재하지 않으며, sentence\_2 데이터에서는 중복된 데이터가 존재하나 전체 columns들을 고려한 row를 기준으로 보았을 때는 중복된 데이터가 나타나지 않았다.
- 속성 간 관계 분석
  - sentence\_1과 sentence\_2의 문장 길이를 분석하여 대체로 20-30의 길이로 구성되며, 하나의 row에 주어진 두 문장의 길이는 거의 유사하다는 것을 확인하였다. 또한, sentence\_1과 sentence\_2에서 문장 길이가 10보다 짧은 데이터는 존재하지 않았으며, 문장 길이가 100을 넘는 소규모 데이터가 포함되어 있어 이에 대한 전처리가 필요하다고 판단하였다.

- 레이블 간 관계 분석
  - 각 레이블을 절대값으로 매핑하여 bucketing을 수행하였을 때, 0이 가장 많은 비율을 지니고 있으며 반대로 5는 매우 작은 값을 지니고 있어 데이터가 불균형하게 분포하고 있음을 확인하였다. 불균형한 데이터 분포를 해결하기 위해서는 강제로 0에 대한 데이터 개수를 줄이거나 data augmentation을 통하여 데이터를 증가할 필요가 있다고 판단하였다.

## 2. 데이터 전처리 : Data Cleaning

문장에서 나타나는 특수문자를 제거했으며, 문장 의미에 기여한다고 판단하는 ‘,’, ‘?’, ‘!’, ‘.’ 는 제외하고 제거하였다.

## 선정 모델 개요

- **klue/roberta-large**
  - BERT와 비교하여 더 많은 데이터셋으로 훈련되었으며, Dynamic Masking을 통해 10가지 방법으로 masking되도록 처리하여 기존 BERT의 MLM과 비교했을 때 성능을 소폭 늘렸다.
  - NSP Loss 대신 다수의 document에서 연속적으로 샘플링된 전체 문장들을 입력으로 사용하여 훈련을 진행했다.
  - Batch\_size를 훨씬 크게 늘려서(8K sequence batch) 학습했다.
  - BERT의 여러 단점들을 보완하고 개선시켰으며, 특히 KLUE github 성능 지표에서 가장 우수한 성능을 기록하였기에 roberta 를 선정했다.
  - hugging face : <https://huggingface.co/klue/roberta-large>  
참조 : <https://jeonsworld.github.io/NLP/roberta/>
- **monologg/koelectra-base-v3-discriminator**
  - 기존의 ELECTA구조를 한국어 데이터로 학습시킨 모델. Generator로 부터 나온 토큰을 discriminator에서 이 토큰의 real/fake 여부를 분류하는 방법으로 학습하는 ‘Replaced Token Detection’ 방식을 기존의 ELECTRA와 동일하게 적용했다.
  - 34GB의 한국어 텍스트 데이터셋(뉴스, 나무위키, 신문, 문어, 구어, 메신저 등)으로 훈련되었고, BERT등과 비교했을 때 더 좋은 성능을 보였다.
  - <https://huggingface.co/monologg/koelectra-base-v3-discriminator>

## 활용 기술

- **data augmentation by retranslation**
  - 한 ↔ 영 retranslation 으로 data를 증강하였다. 증강된 데이터의 품질을 유지하기 위해 코사인 유사도를 사용하였다. 기준으로 원래 문장과 유사도를 판별하여 0.9이상인 문장만 학습 data에 추가하였다.
    - 기계번역시 papago crawling과 google translation api를 사용하였다.
    - 증강된 dataset에서 결측치를 제외하고, 코사인 유사도를 계산해 코사인 유사도별로 dataset을 생성하였다. threshold 코사인 유사도 값이 올라갈수록 증강된 데이터의 개수는 감소하지만, 품질은 증가한다. 데이터의 양과 질 모두 학습의 중요한 요소이기에 최적 지점을 찾고자 다양한 dataset으로 benchmark를 확인하였다.
    - 4.2 검증 결과 -3. data augmentation 성능 비교 란을 참고하면 코사인 유사도 0.9인 dataset에서 가장 큰 성능개선이 있었음을 확인할 수 있으므로, 해당 dataset을 학습에 사용하였다.

◦ 참조 : [최종 3th : \[Private 5위 - 0.83705 / Back Translation\] - DAICON](#)

- **모델 부분 동결**

base 모델의 파라미터를 부분적으로 동결하여 학습을 진행. 1/3, 2/3의 비율로 파라미터를 부분적으로 동결한다. 그후에 동결하지 않은 파라미터만을 학습하는 것으로 성질은 비슷하나 훨씬 큰 데이터에서 학습된 베이스라인 모델의 성능을 유지하고자 하였다.

참조 : <https://dacon.io/en/codeshare/3039>

- **Loss function 변경**

기존의 베이스라인 코드에서 제시한 loss function인 L1 loss는 미분 불가능한 지점이 존재하다는 특징이 존재한다. 때문에 해당 loss가 학습에 적절하지 않을 수 있다고 판단하여 다양한 loss를 탐색 및 적용하여 보았다.

- L2 loss : 모든 지점에서 미분이 가능하나 함수 내에 제곱을 하는 구간이 존재하므로 이상치에 취약하다는 단점이 존재
- Huber loss : L1과 L2 loss의 장점을 취한 형태. 모든 지점에서 미분이 가능하나 이상치에는 강건한 형태

- **모델 변형 및 추가**

기존 klue/roberta-base, klue/roberta-large의 마지막 output인 hidden state vector들을 input으로 하는 fully connected layer를 추가하였다.

(각각의 dropout은 0.2의 비율을 적용. [batch\_size, sequence\_length, representation\_dimension] 순서)

- Structure

[Dropout → Linear → Activation] Block을 쌓아서 구현했다.

- Layer 1

Block1 → Block2 → Block3(output)

- Layer 2,3

Block1 → Block2 → Block3 → Block4 (output)

- Dimension

- Layer 1

[16, 512, 768] → [16, 512, 384] → [16, 1]

- Layer 2

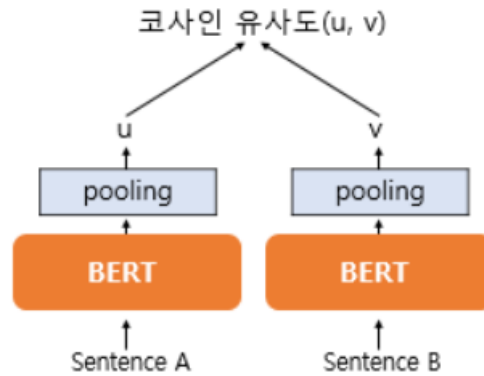
[16, 512, 768] → [16, 512, 512] → [16, 512, 256] → [16, 1]

- Layer 3

[16, 512, 1024] → [16, 512, 512] → [16, 512, 256] → [16, 1]

- **SBERT**

- BERT의 문장 임베딩 성능을 개선시킨 방식이다.



- 두 문장을 독립된 모델에 입력하고, 출력된 모든 토큰들에 대한 output vector에 평균 풀링(혹은 맥스 풀링)을 적용하여 임베딩 벡터 2개(u, v)를 구한다.
- 이후 임베딩 벡터 2개에 대해 코사인 유사도를 구한다. 이때, 코사인 유사도의 범위는 [-1, 1]이지만 우리가 예측하고자 하는 값의 범위는 [0, 5]이므로 레이블 값들을 5로 나누어 학습을 진행한다.
- 본 프로젝트에서는 roberta-large로 SBERT를 사용하였다.
- 참조 : <https://wikidocs.net/156176>

#### • K-fold / Stratified K-fold

데이터 쌍은 10,000개 남짓으로 그 수가 굉장히 부족하여 모델의 성능 향상에 있어 고려가 필수적이었다. 따라서 K-fold cross validation을 구현하여 train 데이터셋을 모두 활용할 수 있게 하였다.

- K-fold: num\_folds 를 인자로 받아 train+dev 데이터셋을 합하여 fold 수 \* epochs 만큼 학습을 진행.
- Stratified K-fold: 레이블의 범위 0~5의 분포가 균등하지 않다는 문제를 해결하기 위해 추가. 기존의 레이블을 절대값처리하여 연속값이 아닌 0, 1, 2, 3, 4, 5로 클래스화를 한 후 해당 값을 토대로 stratified K-fold 학습을 진행

#### • Ensemble - soft voting / weight voting

여러 모델이 도출한 결과값을 토대로 평균(soft voting)을 내거나 가중평균(weight voting)을 내어 극단적인 값을 완화하는 것으로 성능 향상을 도모하였다.

- soft voting : 동일한 가중치로 평균
- weight voting : 모델마다 다른 가중치로 평균

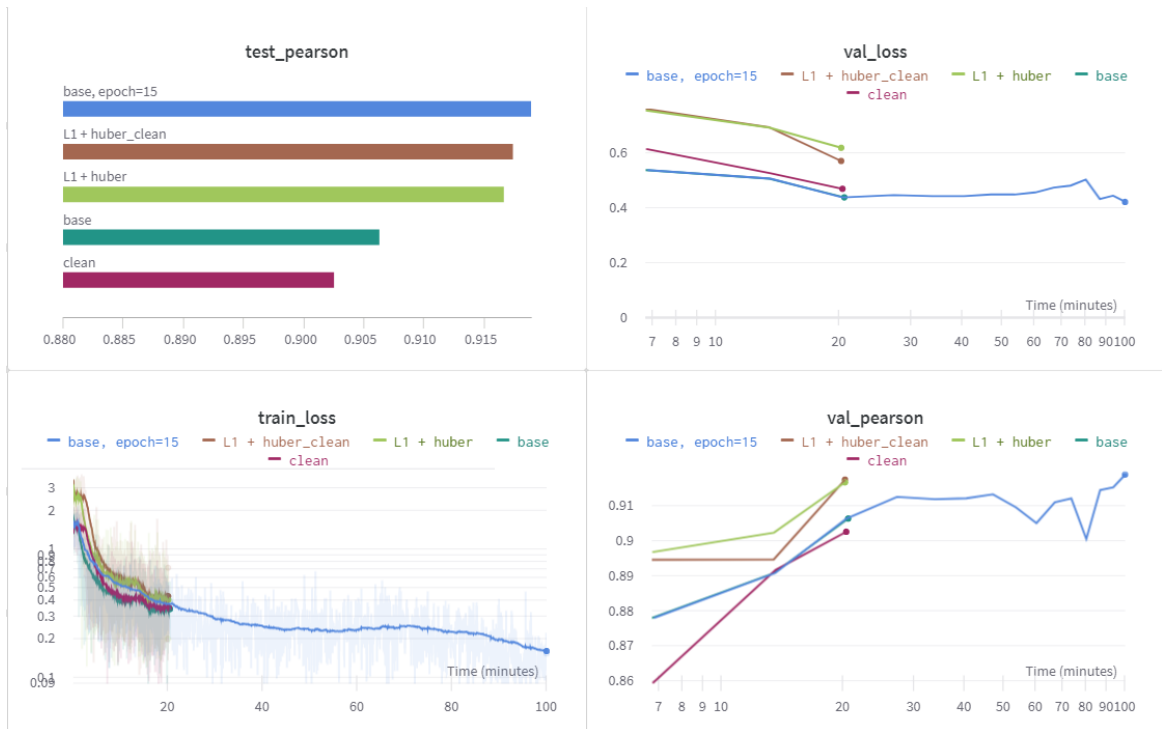
#### • Ensemble Learning (soft voting / stacking)

여러 모델을 동시에 학습하여 성능을 향상해보고자 시도

- soft voting learning: loss를 voting 형태로 평균을 내어 학습
- stacking learning : pretrained model을 기준으로 도출된 값을 concat한 후에 linear layer를 통과하여 값을 예측하도록 학습

## 4.2 검증 결과

### • 전반적인 기법 비교



구현했던 기법들의 성능을 비교하기 위해 변인을 동일하게 놓고 실험을 진행하였다. (epoch=15모델은 변인이 epoch로 base model과 비교함)

위의 그래프를 기반으로 Loss function이 L1 + Huber loss이며 epoch는 15인 모델을 새로운 base model로 선정하고 학습을 진행하였다.

#### • Base Model 성능 비교 (epoch=5 기준)

적절한 성능을 지닌 base model을 선택하기 위하여 5 가지의 모델을 토대로 분석을 진행하였다.

klue/roberta-large의 경우, 그래픽 카드의 메모리 크기의 한계로 인하여 batch size를 8로 감소시킨 후 동일하게 5 epoch 학습을 수행하였다. dev dataset으로 성능을 측정한 결과 klue/roberta-large가 가장 좋은 성능을 보였으며 다음으로 monologg/koelectra-base-v3-discriminator가 그 다음으로 좋은 성능을 보여 두 모델을 베이스 모델로 선택하였다.

모델 종류	batch size	test pearson
klue/roberta-small (Default Model)	16	0.848
monologg/koelectra-base-v3-discriminator	16	0.866
klue/roberta-large	8	0.8963
smartmind/roberta-ko-small-tsdae	16	0.156
monologg/koelectra-small-v3-discriminator	16	0.829

#### • 전처리 성능 비교 (epoch=5, batch size=16 기준)

특수문자들을 제거함으로써 성능이 소폭 상승하였다. 이를 통해 데이터셋에서 문장의 의미에 관계 없는 특수문자들이 학습의 방해요소로 작용했음을 알 수 있다.

기법 종류	모델 종류	test pearson
Default	klue/roberta-small	0.848
data cleaning	klue/roberta-small	0.8683

- **Data Augmentation 성능 비교** (epoch=5, batch size=16 기준)

증강된 data를 필터링할 적절한 코사인 유사도 값을 선정하기 위해 필터링 기준별로 다양한 dataset을 만들어 klue/roberta-small 모델에서 hyperparameter를 통일하고 실험을 진행하였다. dev dataset으로 성능을 측정한 결과 코사인 유사도 0.9이상을 증강 데이터로 사용한 dataset이 가장 좋은 성능을 보여 채택했다.

기법 종류	모델 종류	test pearson
Default	klue/roberta-small	0.848
auged_similarity 1.0	klue/roberta-small	0.869
auged_similarity 0.9	klue/roberta-small	0.876
auged_similarity 0.8	klue/roberta-small	0.868
auged_similarity 0.7	klue/roberta-small	0.858

- **모델 변형 기법 성능 비교**

1. **Loss function 변형** (monologg/koelectra-base-v3-discriminator 기준)

Huber loss와 L2 loss는 빠르게 수렴하긴 하나 단일로 사용하기에는 일정 epoch 후에 바로 오버피팅이 발생하는 결과를 확인하여 L1+Huber loss를 함께 학습하여 결과 비교하였다. 그 결과 L1+Huber loss가 수렴하는데 필요한 epoch 수가 가장 큰 반면에 다른 방법에 비하여 높은 성능을 확인할 수 있었다.

기법 종류	epoch	batch size	test pearson
Default(L1)	5	16	0.866
L1 + Huber Loss	20	16	0.922
L2 loss	5	16	0.911

2. **베이스 모델 파라미터 partial freezing** (monologg/koelectra-base-v3-discriminator 기준)

베이스 모델의 파라미터를 1/3, 2/3 가량 동결한 후 학습을 진행하여 테스트 결과를 확인하였다. 그 결과 1/3 가량 학습을 진행하였을 때, 기존에 동결을 하지 않은 모델 학습보다 높은 성능치를 확인할 수 있으며 2/3에 해당하는 파라미터 동결에서는 오히려 성능치가 감소하는 모습을 확인할 수 있었다.

기법 종류	epoch	batch size	test pearson
Default	15	32	0.905
freeze 1/3	15	32	0.917
freeze 2/3	15	32	0.899

3. **모델 변형 및 레이어 추가 비교**

Layer1~3는 dropout(0.2), linear, tanh 를 차례로 통과하는 layer을 두 번 쌓은 구조. Fully connected layer의 차원 수를 다르게 하며 학습시켰다. 기본 모형의 차이(base/large)도 있지만 복잡한 구조일수록 상대적으로 좋은 성능을 보이는 것 같지만 큰 차이를 보이지 않았다.

기법 종류	epoch	batch size	test pearson
klue/roberta-base + Layer1	5	16	0.8965
klue/roberta-base + Layer2	5	16	0.9006
klue/roberta-large + Layer3	5	16	0.9045

- **SBERT 학습 기법 성능 비교**



다른 학습 방식에서는 높은 성능을 기록하던 koelectra가 SBERT로 사용되었을 때는 오히려 성능이 떨어졌다. 정확한 이유는 알 수 없지만, SBERT에서 cosine similarity loss를 사용했기에 roberta가 이 loss에 보다 적합하지 않았나 생각한다.

또한, roberta-large와 roberta-base가 성능면에서 큰 차이를 드러내지 않았다. 이는 다른 실험에서도 나타난 결과이므로, 모델에 비해 학습 데이터가 부족해서 모델의 크기에 따른 성능 차이가 크지 않았다고 생각한다.

모델 종류	epoch	test pearson
klue/roberta-large	10	0.8914
klue/roberta-base	15	0.88
KR-SBERT-V40K-klueNLI-augSTS	4	0.8482
koelectra-base-v3-discriminator	15	0.8049

- **K-fold/stratified K-fold 성능 비교** (roberta-large, batch=8, K=5 기준)

데이터의 양을 대략 16% 정도 증가시키는 효과가 있는 만큼, Ensemble을 제외하고 가장 많은 점수의 향상이 있었다. Label간의 밸런스가 맞지 않는것을 고려했을 때, Stratified의 추가 구현은 약간이지만 확실히 성능이 오르는 모습을 볼 수 있었다.

따라서 이후의 모든 제출에는 Stratified K-fold를 포함하여 submission을 진행하였다.

기법 종류	Loss	epoch	batch size	test pearson	score
Stratified-K-fold	L1 + Huber	3	8	0.9288	0.909
K-fold	L1 + Huber	3	8	0.9748	0.908
Stratified + Stacking (+ koelectra)	L1	1	8	0.8353	

- **Ensemble Learning 성능 비교** (Loss=L1 기준)

단일 모델을 사용할 때와 비교하여 여러 모델을 앙상블했을 때 성능이 오히려 하락하는 결과를 확인할 수 있었다. 특히 soft voting의 경우에는 사실상 두개의 모델을 각자 학습하는 것과 동일하였기 때문에 분석되며 stacking의 경우 처음부터 학습하기 보다는 일정 이상의 성능을 지닌 모델을 토대로 freeze 한 후에 linear 레이어만을 학습하였을 때 더 좋은 효과를 낼 수 있었을 것으로 보인다.

모델	기법 종류	epoch	batch size	test pearson
roberta-small + koelectra-base-v3	Soft Voting	5	16	0.8103
roberta-large + koelectra-base-v3	stacking	5	16	0.8610

## 5. 자체 평가 의견

- **잘했던 부분**
  - Github을 극한까지 잘 활용했음
  - 다양한 방법론을 제시해보았으며 모두 실험을 진행해보았음
  - 팀원간의 역할을 유연하게 맡아 잘 분배되도록 하였음
- **아쉬운 부분**

- Hyper parameter 조정 안해봄(sweep)
- 모델을 변형해보거나 다양한 레이어를 추가하는 시도가 부족했음
- 데이터 전처리 과정에서 시도가 부족했음
- 체계적인 실험 과정이 부족했음

## 6. 개인 회고

### 김산

#### • 이번 프로젝트에서 나의 목표는 무엇이었는가?

리더보드에서 상위권을 차지하기보다는 AI 프로젝트 경험을 쌓는다는 것에 무게를 두었습니다. AI 개발자 5명이 모여서 프로젝트를 진행하는 것은 처음이었기에 어떻게 협업하고 커뮤니케이션 하는게 좋을지 고민해보고, 나아가 역할분담과 모델 개발을 각자 어떻게 해야 할 지에 대해서도 생각해보는 것이 목표였습니다.

#### • 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

코드공유를 위해 깃허브를 사용하였고 모델의 성능 평가를 공유하기 위해 wandb를 사용했습니다. 깃에서는 각자 브랜치를 생성하여 기능을 구현하고 Pull Request를 적극적으로 활용했습니다. 커뮤니케이션은 주로 줌과 노션으로 진행되었습니다. 제가 맡은 기능을 구현하고 이를 설명함에 있어서 어떤 기능을 어떻게 구현했는지보다는 어떻게 사용하는지에 대해 중점적으로 얘기한 것 같습니다.

역할분담은 PM, 데이터, 모델 3가지로 나누려고 했지만, 할 일들을 모두 적어놓고 각자 맡아서 하는 것도 편리하고 직관적이라고 생각하여 Todo 리스트를 적어놓고 각자 구현하는 방식으로 진행했습니다. 또한 모델 개발의 경우 각자 구현한 것을 깃허브에 올리고 줌으로 어떤 기능을 구현했는지 얘기해주는 것으로 진행 상황을 공유했습니다.

#### • 나는 어떤 방식으로 모델을 개선했는가?

텍스트 데이터에서 상대적으로 필요없거나 문맥적 의미와 관계가 없는 특수문자를 제거하였습니다. 또한, 점수를 0,1,2,3,4점대로 나누어서 분포를 관찰한 결과 0점대의 분포가 압도적으로 많다는 것을 확인하여 undersampling을 시도해봤습니다. undersampling을 통해 0점대의 데이터 중 무작위로 선택된 데이터들을 삭제하여 데이터 분포를 맞춰주었습니다. 이외에도 모델이 예측한 점수의 분포를 살펴보는 코드를 작성했는데, dev\_set과 비교했을 때 SBERT의 추론 결과에는 0점대는 낮고 4점대는 높은 분포가 나타났습니다. 이에 착안하여 모델이 추론한 결과값에 따라 값을 조정하는 방법을 고민했지만, 성공적이진 못했습니다.

#### • 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

undersampling을 적용한 train\_set으로 학습한 SBERT는 각 데이터의 유사도를 3~5점대 사이의 점수들로 편향적으로 예측했습니다. 그러나 막상 이를 다른 모델들과 함께 앙상블에 사용한 결과 당시 가장 높았던 점수를 웃돌았습니다. 이를 통해 다소 이상한 모델이라도 어쩌면 앙상블에 사용되었을 때 Regularization으로 사용될 수 있지 않을까 하는 생각을 하게 되었습니다. 또한, 앙상블로 여러 모델들을 합했을 때 성능이 월등히 좋아지는 것을 통해 앙상블 기법이 대회에서 굉장히 중요하다는 점, 그리고 성능 향상에 크게 작용한다는 것을 느끼게 되었습니다.(사실 이전에는 앙상블이 정말 효과적인지 의심하는 생각을 가지고 있었습니다..)

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

우선 SBERT 학습에 wandb를 적용하기 위해 라이브러리 코드를 분석했던 것이 가장 새로운 시도였습니다. 기존에는 단순히 라이브러리를 가져다가 적당히 사용하기만 했는데, 이번에는 wandb 없이는 학습이 어떻게 진행되는지를 분석할 마땅한 방법이 없어서 어쩔 수 없이 깃허브를 찾아 들어가 코드를 한줄 한줄 봤던 것 같습니다. 분석을 토대로 SBERT 학습 방식을 오버라이딩하여 코드로 구현하였고, 생각했던 것만큼 어렵지 않았기에 앞으로 라이브러리를 보다 유연하게 사용할 수 있을 것 같습니다.

또한 문득 생각이 들어서 모델이 예측한 결과의 분포를 분석했었고, 이를 통해 결과값을 조정하고자 고민했었습니다. 비록 성공적이지는 못했지만, 현재 모델이 어떤 특성이 있고 어떤 문제가 있는지를 파악할 수 있었습니다. 이 경험으로 데이터 분석을 통해 모델 개선을 위한 독창적인 방법을 찾아낼 수 있다는 것을 느꼈습니다.

#### • 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

아쉬운 것이 너무나도 많지만, 데이터들을 보다 자세히 들여다보지 못했던 점이 가장 아쉽습니다. 앞서 언급한 것처럼, 데이터를 여러 방향으로 분석하다보면 어떤 특징이나 문제가 있는지를 찾아낼 수 있을 것입니다. 그러나 그런 부분에서 조금 소홀하지 않았나 하는 아쉬움이 있습니다.

또한 다른 팀의 발표를 들었을 때, 전체적으로 조금 틀에 갇힌 사고로 프로젝트에 임하지 않았나 싶습니다. 단순히 모델과 파라미터들을 바꾸는 것에 그치지 않고, 관련 논문들을 살펴보고 힌트를 얻고 loss, optimizer, preprocessing 등 다방면에서 다양한 시도를 할 수 있었는데 이런 것들을 생각하지 못했던 것이 후회됩니다. 조금 더 열린 생각을 가지고 이것저것 건드려보는 것이 결과를 좋게 만들 수도 있고, 혹은 결과와 무관하더라도 좋은 공부 되었을 텐데 이런 시도들을 생각조차 해보지 못한 것이 가장 아쉽습니다.

#### • 한계/교훈을 바탕으로 다음 P-Stage에서 스스로 새롭게 시도해볼 것은 무엇일까?

다음에는 보다 데이터 분석에 많은 시간을 투자해볼 것입니다. 다른 사람들은 어떻게 분석하는지를 참고하면서 데이터를 어떻게 활용할 수 있을지에 대해서 보다 깊게 고민하겠습니다.

그리고 무엇보다 다양한 시도를 해보고 싶습니다. “이건 쓸모없어”라기보다는 “Why not?”이라는 마인드로 전체 모델부터 미세한 파라미터까지 이것 저것 수정해보고 결과를 분석하면서 내가 수정한 것이 어떤 효과가 있었는지, 그리고 이 효과가 모델에 어떤 영향을 미쳤는지, 왜 이런 영향을 미쳤는지에 대해서 분석해볼 것입니다.

## 김지수

### 이번 프로젝트에서 나의 역할은 무엇이었는가?

대회를 시작하고 난 뒤, 팀 내에서 대회의 경험이 있던 사람은 저밖에 없었기 때문에 빠르게 실질적인 팀장의 역할을 맡게 되었습니다. 팀원들의 경험이 전무했으므로 목표 지정부터 스케줄 관리까지 자연스럽게 **관리자 포지션에서 팀원을 이끌고 프로젝트의 방향을 설정**하는 것이 이번 프로젝트의 저의 역할이 되었습니다.

### 나의 목표는 무엇이고 그 목표를 달성하기 위해 무엇을 어떻게 했는가?

#### Do not reinvent the wheel

이런 대회를 마주했을 때 제일 조심하고 경계해야 되는 일은 ‘이미 만들어져 있는 방식을 다시 만들지 말자’입니다. 이는 기술적인 측면과 협업 측면에서 동시에 일어날 수 있는 일인데, 협업 프로세스가 잘 정리되지 않았거나 또는 목표 설정이 불분명할 때 종종 일어나게 됩니다. 이러한 리스크를 최소화하고 대회를 진행하기 위해서 정한 저의 목표는 다음과 같습니다.

모델 개선을 위한 **기술적 측면에서의 목표 설정**, 팀원들의 협업과 토론을 위한 환경을 제공하는 **협업 프로세스 관리**, 리더로서 팀원들이 실제로 제대로 역할을 수행하고 필요한 부분이 없는지 관리 감독하는 **관리자의 역할** 크게 세 가지로 목표를 잡고 대회를 진행하였습니다.

## 기술적 측면에서의 목표 설정

모델 개선을 위해 대회와 같은(혹은 비슷한) 이전의 다른 대회들에서 정보를 취합하였으며, 많은 구현이 필요한 대규모 목표와, 작지만 성능을 향상할 확률이 높은 작은 목표를 나누어 정하였습니다. 제가 정하고 구현 또는 직접적으로 도움을 준 task는 다음과 같습니다.

- **K-fold**  
대회의 데이터셋은 10,000 문장 쌍 남짓으로 얼마 되지 않아 데이터셋의 크기가 모델에 영향을 가장 많이 줄 것이라 판단하였고, 따라서 데이터를 train시에 모두 활용할 수 있는 K-fold cross validation 기법을 적용하여 Ensemble 다음으로 큰 폭의 성능 향상에 기여 하였습니다.
- **분석 도구 및 방법론 설정**  
대회의 Baseline 코드는 train - validation - test의 데이터 구조가 아닌 validation 데이터셋을 test 데이터셋과 공유하는 구조를 가지고 있었으며, 이는 모델의 실 성능 측정을 희석되게 합니다. 따라서 validation셋을 나누는 코드를 구현하고 인자로 만들어 모델을 제대로 평가할 수 있게 하였으며, 제출용으로는 train 데이터셋 전체를 training에 쓰게 하였습니다. 또한 모델 분석 도구인 **wandb**를 구현하여 그래프로 분석을 쉽게 하였습니다.
- **기타**  
부동소수점에 변화를 주어 속도를 개선하는 mixed precision, 모델을 모니터링 하며 학습을 중지시키는 Early stopping 구현, sentence BERT와 같은 새로운 구조의 모델 구현 제안, Back-translation으로 데이터 증강하는 방법론 제안 등 팀원들에게 목표를 제안하고 할당

## 협업 프로세스

Baseline 코드의 프로젝트 파일 구조를 협업하기 쉬운 **파일별로 모듈이 할당되는 구조**로 변경하였고, 이는 **프로젝트 구조도**에서 확인할 수 있습니다. 또한 Github과 Git을 엄격하게 쓰게 하여 개선 사항과 문제 제기는 Issues 에서, 코드 구현은 팀원별로 branch를 나누게 하고 pull request로 code review 없이는 main에 merge하지 않게 하였습니다.

## 위의 목표로 달성한 것, 한계점, 깨달음

모델의 개선과 코드 구현도 물론 대회에 많은 기여가 되었지만, 이번 대회에서 제일 잘 된 점을 평가하자면 협업 프로세스를 명확히 하고 할 일이 팀원들에게 적확하게 부여되었던 점입니다. 결국 대회는 5명의 팀원이 함께 하는 일이고, 혼자서 아무리 잘 한다고 해도 절대적인 시간이 부족하게 됩니다. 하지만 서로 본인의 상황을 공유하고, 할 일을 정함에 따라 대규모 목표와 작은 목표 모두 시도할 수 있게 되었습니다. 물론 제출의 양적인 면이나 점수 자체로는 부족했던 점이 많았으나, 그럼에도 불구하고 팀의 잠재 능력은 최고로 쓸 수 있어 아쉬움이 없는 대회였습니다.

## 한계/교훈을 바탕으로 다음 P-Stage에서 스스로 새롭게 시도해볼 것은 무엇일까?

처음으로 주도적으로 진행하는 대회였기 때문에 많은 고민이 있었던 것이 사실입니다. 다음 Stage에서는 고민과 취합 과정은 더 빠르게 해결하고 성적도 챙겨갈 수 있는 대회가 되길 희망합니다.

## 박수현

- 이번 프로젝트에서 나의 목표는 무엇이었는가?
  - 딥러닝 프로젝트의 진행 과정과 에너지를 어느 정도 분배해야 하는지 배우기
  - 협업 툴(GitHub, Notion, WandB 등) 과 환경설정(서버 구축, ssh 연결 등) 숙달하기
  - 강의에서 배운 테크닉(k-fold, ensemble, 다양한 모델의 성능 비교 등)이 적용된 모델 경험해보기
  - baseline 코드 이해하기
  - 프로젝트의 mainstream인 모델 개선에 참여하기에는 실력이 부족하지만, side job 하나 맡아서 책임감 있게 완수하기

- 스스로를 잘못 평가해 능력 이상의 과제를 맡겠다고 하고, 완성하지 못하면 팀원들에게 하중이 부가될 수 있다고 생각했다

• 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 딥러닝 프로젝트 경험이 많은 팀장이 제시한 과정을 개인 노선에 정리했다
- 팀 GitHub에 push/main branch 에 merge/pull request merge를 경험했다
- side job인 data augmentation 을 수행했다

• 나는 어떤 방식으로 모델을 개선했는가?

- Data augmentation 담당.
- emoji를 필터링하고 Google translator, papago를 이용하여 주어진 dataset에 한-영 retranslation 을 수행했다.
- 증강된 데이터를 cosine similarity를 기준으로 필터링하여 다양한 dataset를 만들었다. 동일 모델로 학습시켜 benchmark를 확인하고 가장 높은 성능을 보인 augmentation dataset을 사용했다.

• 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 약 40%의 데이터 증강을 달성했다.
- 증강된 dataset이 대부분 benchmark score가 증가함을 확인하여 모델을 학습시키는데 데이터가 중요함을 깨달았다. 필터링하는 cosine similarity score 를 너무 낮게 잡은 경우에도 benchmark score가 하락하는 것으로 보아 데이터의 양뿐 아니라 질도 중요함을 깨달았다.
- 대회 종료 private 에서는 benchmark과 달리 cosine similarity 0.9가 아닌 0.7 기준으로 filtering된 것이 가장 성능이 좋았던 것으로 나타나, 지식적 배경이 더 있었다면 현명한 결정을 할 수 있었을 것이라는 아쉬움이 있었다.

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

- 협업을 위해 팀 GitHub을 사용하여 타인과 코드를 공유했다
- Data Augmentation 을 위해 Dacon을 참고하여 web crawling을 시도했다
- 성능 향상을 위한 다양한 기법(loss function 변경, Kfold(normal, stratified), ensemble(soft voting, hard voting, bucketing))을 사용한 모델들의 구조 파악  
→ 팀 competition 참여 경험이 없어 새로운 시도를 따로 구분하는 것이 의미가 없다. 그러나 협업과 대회를 위한 워크플로우 및 필요 툴들을 익혀 다음 프로젝트를 위한 기반을 마련했다

• 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- baseline code를 이해하는 것 이상으로 모델 자체의 성능개선에 기여하지 못했다
- dacon 의 코드를 참고하여 data augmentation을 했는데, 성능 개선에 가장 효율적인 augmented data를 필터링하는 기준을 설정하지 못했다
- 상기 [행동의 결과와 깨달음] 에 언급하였듯 결과에 대한 지식적 근거가 부족했다

- retranslation이 동작하는 시간이 오래 걸릴 것을 예상하지 못해 결론적으로 모델을 학습시킬 시간이 부족했다.

• **한계/교훈을 바탕으로 다음 P-Stage에서 스스로 새롭게 시도해볼 것은 무엇일까?**

- 모델 성능 개선에 직접적으로 참여해보겠다
- 주어진 시간에 따른 time scheduling을 보다 면밀하게 하겠다
- 모델의 성능을 설명할 수 있는 지식적 근거를 뒷받침하기 위해 강의와 스터디에 힘쓰겠다

## 업무주언

1. 이번 프로젝트에서 나의 목표는 무엇이며 나의 학습 목표를 달성하기 위해 무엇을 하였는가?

제가 이번 프로젝트에서 가장 중요하게 생각한 것은 **협업**과 **다양한 실험적 시도**였습니다. 동일한 딥러닝 엔지니어 포지션으로서 5명이 프로젝트를 진행하는 것은 처음이었기 때문에 이번 프로젝트의 워크플로우에 익숙해지는 것. 이게 바로 제가 얻을 수 있는 가장 값진 경험이라고 생각하였습니다.

2. 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

먼저 저는 본 팀이 데이터나 모델, 학습 등의 담당을 유연하게 맡을 수 있도록 하였다는 장점을 토대로 적극적인 참여를 통해 팀에 이바지하고자 노력하였습니다.

제가 협업을 위하여 가장 먼저 진행한 것은 **프로젝트 코드의 편리성 고도화**입니다. 기존의 베이스라인 코드에서는 checkpoint를 저장하고 로딩하거나 학습률 scheduler 등이 구현되어있지 않는 상태였습니다. 이러한 기능들이 빠르게 구현되어있다면 팀이 적절한 실험을 할 수 있다고 판단하였으며 팀원들과 상의 후 프로젝트 초기에 해당 기능을 빠르게 구현하여 사용할 수 있도록 하였습니다. 또한 대회의 마지막 무렵이 되기 전, 미리 앙상블 voting 기법을 구현하여 담당을 나누어 실험한 모델 결과를 취합하여 기법을 구현해야한다는 부담을 덜게 하려 노력하기도 하였습니다.

3. 나는 어떤 방식으로 모델을 개선했는가?

저는 이번 프로젝트에서 모델에 대한 변경과 학습 개선을 통하여 개선하고자 하였으며 성능을 높이기 위해서 어떤 방법론을 사용해야한다.에 얽매이지 않고 최대한 다양한 실험을 진행해보고자 하였습니다. 베이스라인 코드에서 제공한 L1 손실함수를 사용하는 것이 적절할까에 대해 의문을 품고 L2 손실함수나 Huber 손실 함수, cross entropy 등 다양한 손실함수를 사용하여 학습을 시도하여 보았으며 여러 손실 함수를 동시에 사용해보며 모델에 성능을 높일 수 있었는지에 대하여 실험해보기도 하였습니다. 또한 모델의 파라미터를 부분적으로 freeze하였을 때, 성능 변화가 어느정도 미치는지에 대하여 실험하여 보기도 하였습니다. 또한 앙상블의 voting이나 stacking을 결과에 대한 앙상블이 아니라 직접 학습 용도로 구현하였을 때의 결과를 확인하고 싶어 학습 용도로 구현하여 실험하여 보기도 하였습니다. 또한 주로 분류문제에서 사용되는 stratified K-fold를 label에 절대값을 취하는 방식으로 bucketing하여 구현하며 모델을 개선하고자 하였습니다.

4. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

L1과 Huber 손실함수를 결합하여 사용하였을 때, 다소 괜찮은 성능 결과를 확인하여 loss function을 여러개 사용하는 방식이 잘못된 것이 아님을 알게 되었습니다. 뿐만 아니라 stratified K-fold를 통한 방법이 K-fold보다 괜찮은 성능을 보이기도 하였습니다. 여러 다양한 가설을 갖고 구현하고 실험하였던 과정에서 대부분 쓰러진 실패를 경험하였으나 소수의 방법은 프로젝트에 기여하기도 하였으며 다양한 방법론들이 프로젝트에는 어떤 결과를 가져오는지 확인할 수 있게 되어 굉장히 값진 경험을 하였다고 생각합니다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

아쉬웠던 점은 베이스라인 코드에 너무 벗어나지 않으려 하였다는 것입니다. 사실 프로젝트 동안에는 반드시 베이스라인 코드를 중심으로 코드를 작성해야한다는 고정 관념이 존재하였던 것 같습니다. 때문에 모델이나

huggingface 호출 함수의 파라미터 등을 조정하지 않고 당연히 그대로 사용하려고 하여 다양한 변화를 주어 실험해보지 못한점이 아쉽습니다. 뿐만 아니라 pretrained model을 사용하기 때문에 모델을 변형하거나 레이어를 추가하는 방식은 성능에 큰 영향을 주지 못할 것이라는 고정 관념에 빠져 많은 실험을 진행해보지 못하였다는 사실이 너무 아쉬운 것 같습니다.

또한 이번 프로젝트에서 모델이나 학습 개선에 너무 집중한 나머지 데이터는 당연히 양질의 데이터를 가지고 있다고 가정하고 대부분의 시간을 사용하였다는 사실이 너무 아쉽습니다. 제가 데이터에 좀 더 신경을 썼다면 이번 프로젝트에서 보다 양질의 결과를 얻을 수 있었을 것 같습니다.

실험 또한 굉장히 중요한 요소중 하나인데 담당할 역할을 구현하는데 집중한 나머지 실험이 체계적으로 이루어지지 않은 것 같습니다.

#### 6. 한계/교훈을 바탕으로 다음 P-Stage에서 스스로 새롭게 시도해볼 것은 무엇일까?

체계적인 프로젝트 구성이 조금 아쉽다고 생각되어 다음 P-Stage에서는 보다 체계적으로 구현과 실험 담당을 나누어 진행하고 보고서 또한 꾸준히 작성할 수 있도록 노력하고 싶습니다.

모델 변형이나 추가를 통한 방법론을 시도해볼 생각을 하지 못했다는 사실이 너무 아쉽습니다. 다음 번에는 관련 논문을 서치해보고 적절한 finetuning method나 modeling이 있다면 적극적으로 시도해보고 싶습니다.

## 현승업

- 이번 프로젝트에서 나의 목표는 무엇이었는가?

딥러닝 프로젝트가 처음이기 때문에, 처음부터 끝까지 하나의 과정을 전부 경험해보고 공부해보는 것이 목표였다. 프로젝트를 위해 어떤 함수, 클래스가 필요하고, 어떤 외부 기능이 쓰이고 그 구성은 어떻게 돼있는지 정도를 완전히 파악하는 것이 1차적인 목표였다.

또한 프로젝트가 처음이다 보니 팀을 이뤄 협업하는 것 또한 처음이었기 때문에 어떻게 해야 같이 효율적으로 프로젝트를 진행시킬 수 있는지에 대해 생각하고, 그 중 내가 할 수 있는 역할을 찾아 작은 것이라도 잘 마무리해보자는 것이 두번째 목표였다.

- 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

우선 주어진 베이스라인 코드들이 어떻게 구성되어 있고 각 파일의 역할을 무엇인지, 각 코드에서의 클래스와 함수들을 어떤 역할을 하는지 파악하기 위해 공부했다.

이후 우리 조에서 정한 TODO list중 한 가지를 골라 구현해 보려고 시도했지만 구조의 이해가 미숙한 탓에 어떤 부분을 수정해야 하는지 잘 알지 못했다. 그래도 그중에서 pretrained model의 가장 마지막 layer에서 나오는 representation을 이용해 task 수행을 위한 fully connected layer를 구현하는 것은 부스트 캠프를 통해 배웠던 내용을 이용해서 시도해 볼 수 있었다.

프로젝트가 끝나고 다시 한번 우리 조의 프로젝트 코드를 보면서 구조에 대한 이해를 좀 더 잘 할 수 있었다.

- 나는 어떤 방식으로 모델을 개선했는가?

결과적으로 내가 해 본 시도는 실패했다.

기존엔 CLS토큰에 대한 representation을 이용해서 STS task를 수행했는데, 문득 전체 토큰에 대한 representation을 이용해보면 어떨까라는 생각이 들었다.

그래서 아래 두 가지 방법에 대해 모델을 fine tuning해봤다.

1. Input 토큰들의 representation 벡터들 dropout layer, fully connected layer, tanh layer 를 통과시켜 최종 output을 계산
2. Input 토큰들의 representation 벡터들의 평균을 계산한 벡터를 이용하여 이를 기존의 CLS토큰에 대한 representation 처럼 사용.

이 과정에서 dropout 비율, fully connected layer의 차원과 layer의 수, 비선형 함수 사용 등의 정해야하는 하 이퍼 파라미터들이 존재해서 미흡하지만 다양한 시도를 통해 모델을 완성하였다.

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

다른 base model을 사용한 모델들보다 성능이 떨어졌었다.

일단 문제는 위에서 언급한 구조로 많은 시도를 해봐야 하는데 그게 부족했던 것 같다. 다양한 layer의 수, 차 원 수를 적용시켜 볼 수 있었고, 다른 non linear function을 사용해 보는 등의 시도를 해봐야 하는데 그게 부 족했다. 추가하는 layer들을 내가 무작위로 정하는 것이 아니라 이에대한 방법이 있는지에 대해서도 좀 더 공부 를 해봐야 했다.

또한 구현을 하진 못하더라도 이런 저런 아이디어를 내는 것을 충분히 할 수 있었는데, 어차피 알아내도 써먹지 못한다고 생각해서 뭔가를 더 찾아보고 생각해보는 시간은 상대적으로 적었던 것 같다.

프로젝트 진행중에는 우리 조에서 진행하고 있는 것들을 이해하는 것만으로도 벅차다는 느낌이 들어서 그랬지 만 지나고 생각해보니 우리 조에서 적용하지 않은, 생각하지 못한 다양한 방법이나 접근법들이 존재했는데 이 런 방법을 제시하는 역할은 충분히 더 잘 할 수 있었는데 못했다는 생각이 들어 아쉬웠다.

프로젝트 뿐만 아니라 항상 주어진 환경에서 내가 할 수 있는 건 무엇인지 찾아보고, 실행해보려는 자세와 마인 드가 필요하다고 생각한다.

- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

이번엔 뭔가 프로젝트 전반에 대해 파악하고 공부하는데 시간을 많이 써서 정작 모형을 구현하고 학습시키는 데에는 집중하지 못한 것 같다.

이번 대회를 통해 처음 경험을 해봤으니, 다음엔 성능 향상을 위해 다양한 관점에서 많이 시도해보고 싶다.

위에서 한번 얘기했지만, 잘 못한다는 생각에 자신감이 떨어지기 보단 뭐라도 찾아서 해보려는 자세를 가져야 겠다는 지금의 생각을 다음 기회엔 꼭 적용시켜 봐야겠다.

- 한계/교훈을 바탕으로 다음 P-Stage에서 스스로 새롭게 시도해볼 것은 무엇일까?

1. 다양한 모형들에 대해 공부하고 시도해본 후 결과를 분석하는 시도를 최대한 많이 해볼 것이다.
2. Argparse 사용법을 제대로 숙지해서 필요한 옵션을 추가/제거 해볼것.
3. wandb, sweep 등을 구현해보는 것.
4. 모델 구현뿐만 아니라 프로젝트의 다른 파트에서도 개선점이 있는지 생각해보며 프로젝트에 참여하기.