
NLP 기초 프로젝트

문장 간 유사도 측정

Semantic Text Similarity (STS)

Member

김진재, 박규태, 윤선웅, 이정민(팀장), 임한택

Wrap-Up Report : 문장 간 유사도 측정

🔗 2 backlinks

김진재_T7327, 박규태_T7334, 윤선웅_T7405, 이정민_T7420, 임한택_T7429

BoostCamp AI Tech 7기 NLP-15팀

1. 개요

이 문서는 2024년 9월 10일(화)부터 9월 26일(목)까지 진행한 NLP 기초 프로젝트 '문장 간 유사도 측정 (STS)' 프로젝트 Wrap-Up 보고서이다.

1.1. 팀 셋업

1.1.1. 팀 역할

팀원	역할
김진재	방법론 제안, 협업 환경 및 베이스라인 관리, 모델 탐색, 증강 기법 및 전처리 실험, 앙상블 코드 작성 및 실험
박규태	모델 탐색, 데이터 증강 및 앙상블 기법에 대한 실험, Bagging 기법 코드 작성 및 실험
윤선웅	협업 환경 및 베이스라인 관리, 데이터 분포 및 재분할 총괄, 모델 탐색, 데이터 증강 실험, 앙상블 코드 작성 및 실험
이정민 (팀장)	모델 탐색, 모델에 대한 증강 기법 및 전처리 실험, KoEDA 증강 실험, K-Fold Validation 실험
임한택	모델 탐색, 모델에 대한 증강 기법 및 전처리 실험, 앙상블 코드 작성 및 실험, Stacking 모델 실험

1.1.2. 그라운드 룰

프로젝트 시작 전, 각자 역할을 정하고 프로젝트를 성공적으로 수행하기 위해 그라운드 룰을 설정하였다. 세부 내용은 다음과 같다.

[Git 구조 설정]

- gitignore에 저작권, 보안 관련 문서 추가하여 관리하기
- 최대한 Public처럼 사용하기
- Git Flow를 이용하여 Feature, main으로 나누어 관리하기
- Local에서 Sandbox 브랜치를 이용하여 각자 자유롭게 코드 실험하기

1.1.3. 로드맵

일자	내용
11	프로젝트 시작
11-12	베이스라인 코드 테스트
12-13	팀원 역할 및 그라운드 룰 설정
13	main branch 생성
13-19	EDA 및 데이터 전처리 - (13-16) EDA - (17-19) 데이터 전처리
13-16	1차 실험 (기본 모델 앙상블) - 역할 분담 - 기본 모델 학습 (배치 사이즈, 학습률, 에폭 사이즈 조절하며 pearson 0.92 이상 모델 추출)
17-19	2차 실험 (데이터 증강 진행) - 데이터 증강 진행 (Swap, Undersampling, Oversampling) - 데이터 전처리를 활용하여 기본 모델 성능 평가 → 성능 저하
20	2차 실험 토론 및 베이스 코드 정리
20-23	Output 앙상블을 활용한 3차 실험
21	Stacking 모델 실험
23	3차 실험 토론
24-25	K-Fold 실험, Bagging 실험
26	최종 제출 및 대회 마감
27	Private 결과 발표

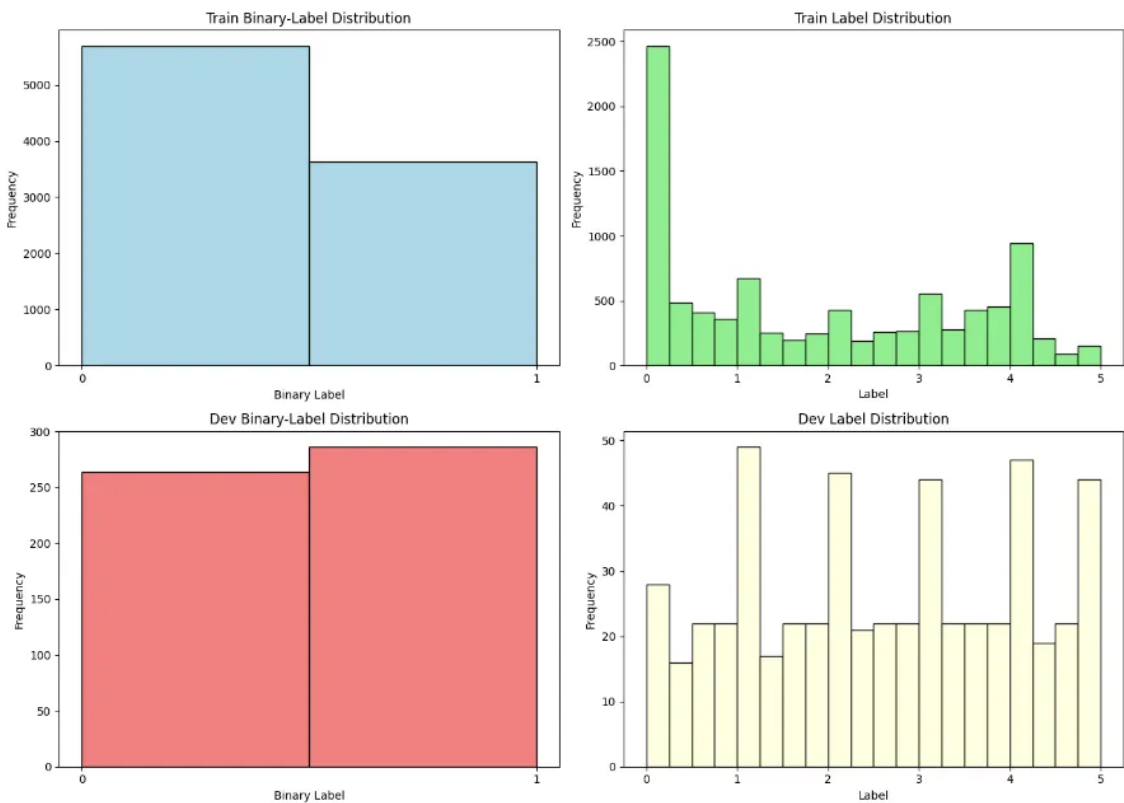
2. 데이터 분석

[Dataset 기본 정보]

Dataset 중 Task를 수행하기 위한 주요 요소에는 두 개의 문장 ('sentence_1', 'sentence_2')과 두 문장의 유사도를 0.0~5.0 사이로 평가한 점수 ('label')가 있다.

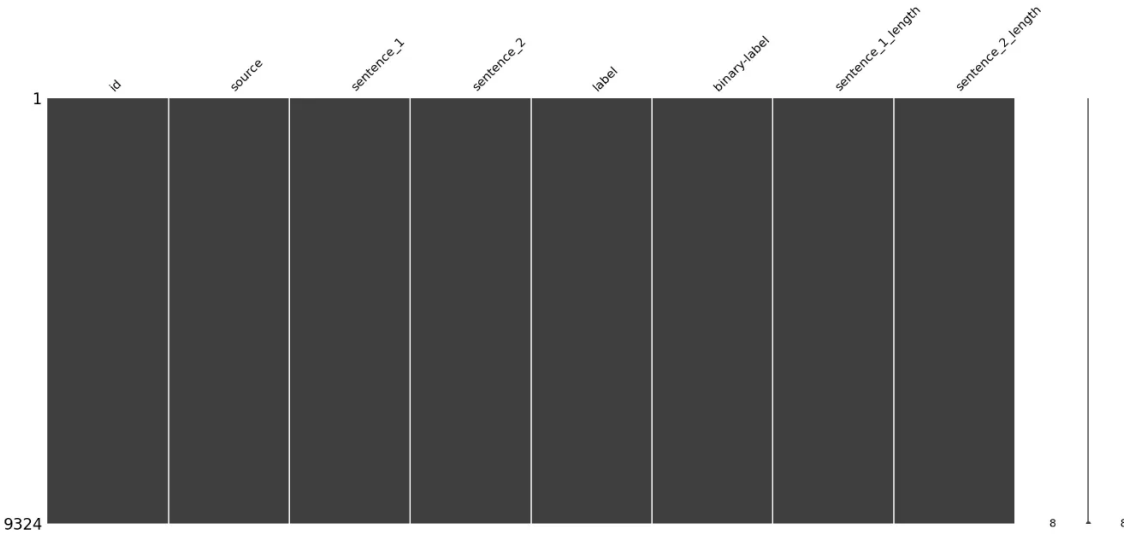
- Train Dataset : 9324개
- Dev (Validation) Dataset : 550개
- Test Dataset : 1100개

[Train, Dev dataset 의 크기와 라벨별 분포 시각화]



Train Dataset에서 Label 중 0에 가까운 값이 대다수로 고른 분포를 보이지 않는다는 점을 EDA를 통해 확인하였다. 해당 분석 결과 데이터 증강 시에는 Label의 불균형을 보정하기 위해 0에 가까운 값을 Undersampling하는 방식을 활용하였다.

[결측치 확인]



Train Dataset 내의 9,324개 데이터를 검증한 결과, 모든 column에 대하여 결측치가 존재하지 않았다. 따라서 결측치 처리는 별도로 수행하지 않았다.

3. 데이터 전처리

2. 데이터 분석 결과를 활용하여, 데이터에 필요로 하는 전처리를 수행하였다.

[불용어(Stopwords) 제거]

텍스트의 노이즈를 줄이기 위한 목적으로 **Stopwords-ko** 를 활용하여 문장의 의미에 큰 영향을 미치지 않는 단어(불용어)를 제거하였다.

Before	After
스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요~	스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요~
앗 제가 접근권한이 없다고 뜬니다;;	제가 접근권한이 없다고 뜬니다;;
...	...

[텍스트 클리닝 (Text Cleaning)]

텍스트 내에서 불필요한 정보인 특수문자, 숫자, HTML 태그 등을 제거하였다.

Before	After
스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요~	스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요
앗 제가 접근권한이 없다고 뜬니다;;	제가 접근권한이 없다고 뜬니다;;
주택청약조건 변경해주세요.	주택청약조건 변경해주세요
...	...

[정규화 (Normalization)]

자주 사용된 단어를 표준어로 바꾸거나, 숫자와 같은 특정 패턴을 일반화하였다.

- 어휘 정규화(Lemmatization) → 단어를 기본형으로 변환
- 숫자 정규화 'NUM' → 숫자 텍스트를 NUM으로 대체

Before	After
크 오늘 날씨 너무 좋았는데, 대박!!	크 오늘 날씨 너무 좋았는데, 대박!!
북한 연락사무소 리모델링 97억8천만원???? 시급한 일이라 사후 계산한다고//	북한 연락사무소 리모델링 NUM억NUM천만원???? 시급한 일이라 사후 계산한다고//
...	...

불용어 제거, 텍스트 클리닝, 정규화 등 다양한 시도를 했지만, 주어진 데이터의 경우 텍스트 전처리를 수행한 경우 오히려 모델의 성능을 저하시키는 결과를 확인할 수 있었다. 따라서, 최종 모델에 쓰인 학습 데이터는 텍스트 전처리를 반영하지 않은, 원래 데이터 자체를 대부분 활용하였다.

3.1. 데이터 증강

데이터의 최초 전처리를 수행한 이후, 데이터를 증강하기 위해 아래와 같은 작업을 시도하였다.

[KoEDA(Korean Easy Data Augmentation)]

일반화 성능을 향상시키고자 동의어 교체, 단어 순서 변경, 랜덤 삭제 등을 시도하였다.

[Data Swap]

- Sentence_1, Sentence_2의 순서를 변경해서 original train 데이터셋의 길이와 같은 SWAP 데이터셋을 만들고 학습을 진행하였다.
- SWAP 데이터셋을 original train 데이터셋에 병합하여 기존의 길이보다 2배 길어지게 만든 후 학습을 진행하였다.
- SWAP 데이터셋을 **snunlp/KR-ELECTRA-discriminator** 로 label 점수를 재평가한 후에, original train 데이터셋과 병합하여 학습을 진행하였다.

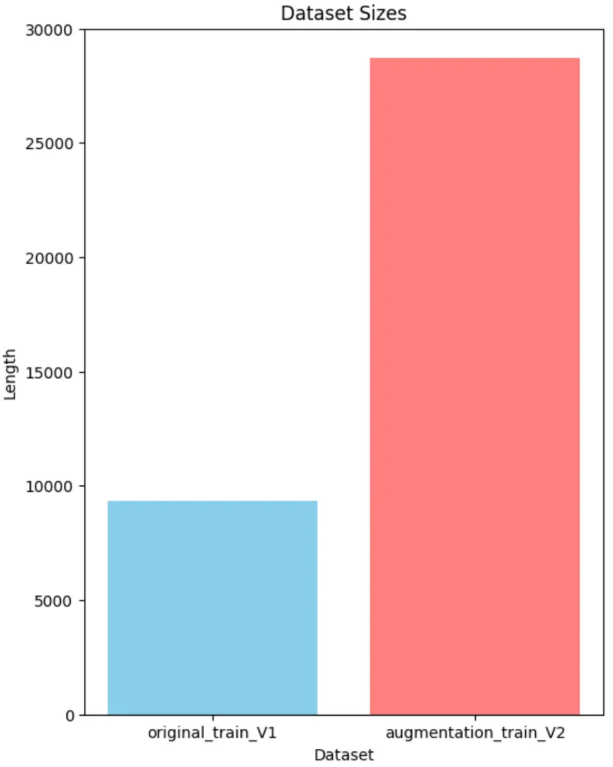
위 세 가지 방식으로 각각 Data Swap을 시도하였고, 위 방법론 모두 Original Train 데이터셋으로 동일 모델을 학습시켰을 때 보다 성능이 낮게 나왔다.

[언더 샘플링, 오버 샘플링]

앞선 데이터 분석 (EDA) 수행 결과, label 0으로 분포되어 있는 데이터가 많아서 균형을 위해 임의로 0에 가까운 데이터의 비중을 줄여서 학습을 수행하였다. 한편, label 0인 데이터에 비해서 label 5의 데이터가 현저히 적었기 때문에 균형을 위한 증강 시도가 있었다.

3.1.1. 최종 활용 데이터셋

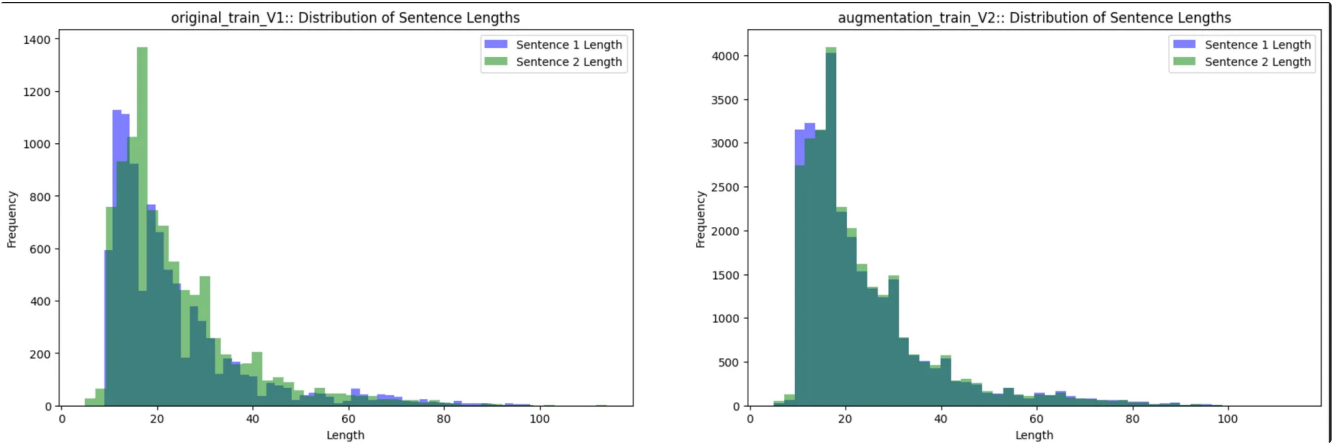
Version	Description	Size
original_train_V1	원래 훈련 데이터셋	9324
augmentation_train_V2	SWAP, label 0 언더샘플링 + label 5 오버샘플링	28722



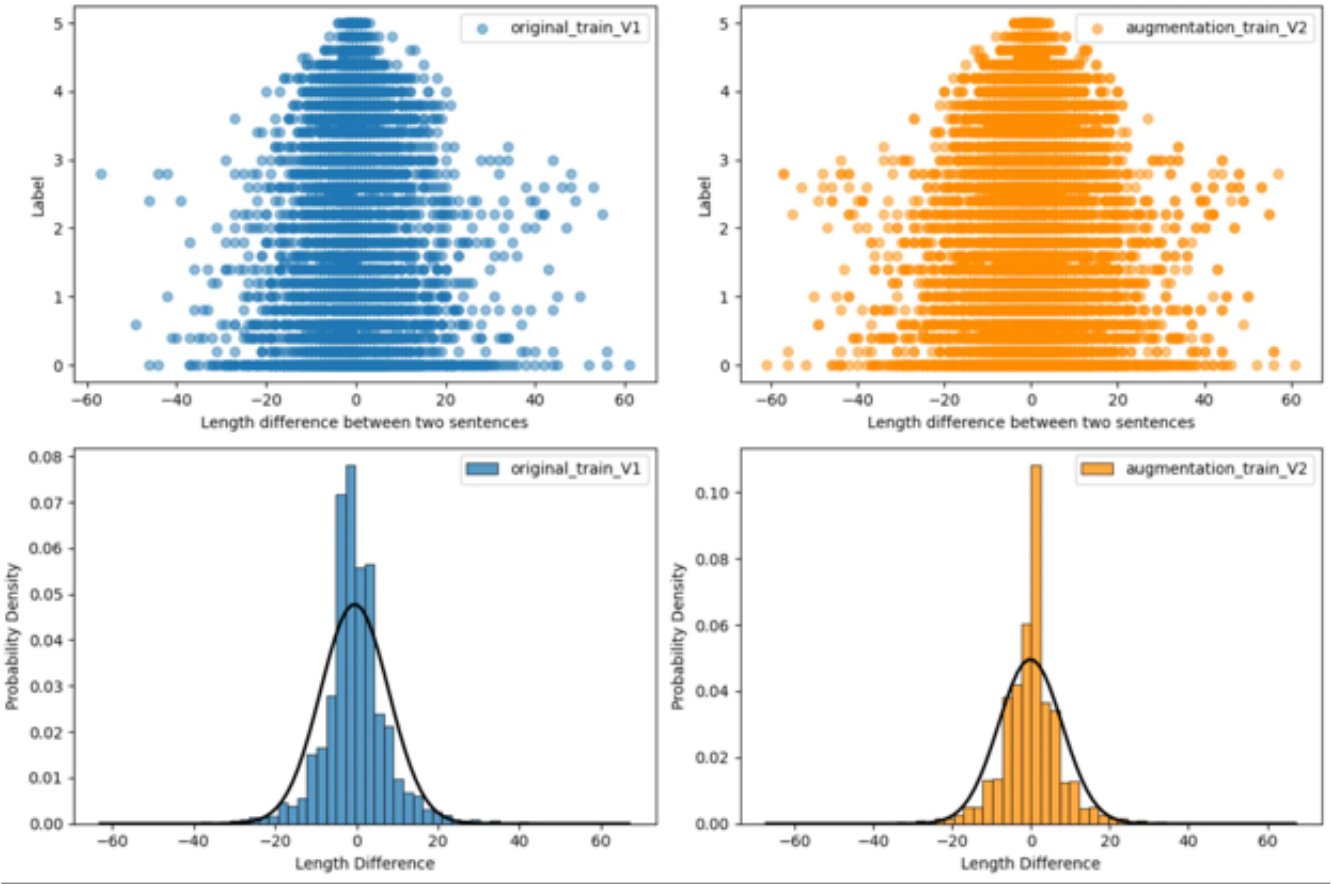
데이터 증강 전후 데이터셋의 길이

[데이터 증강 후의 문장 길이관련 EDA]

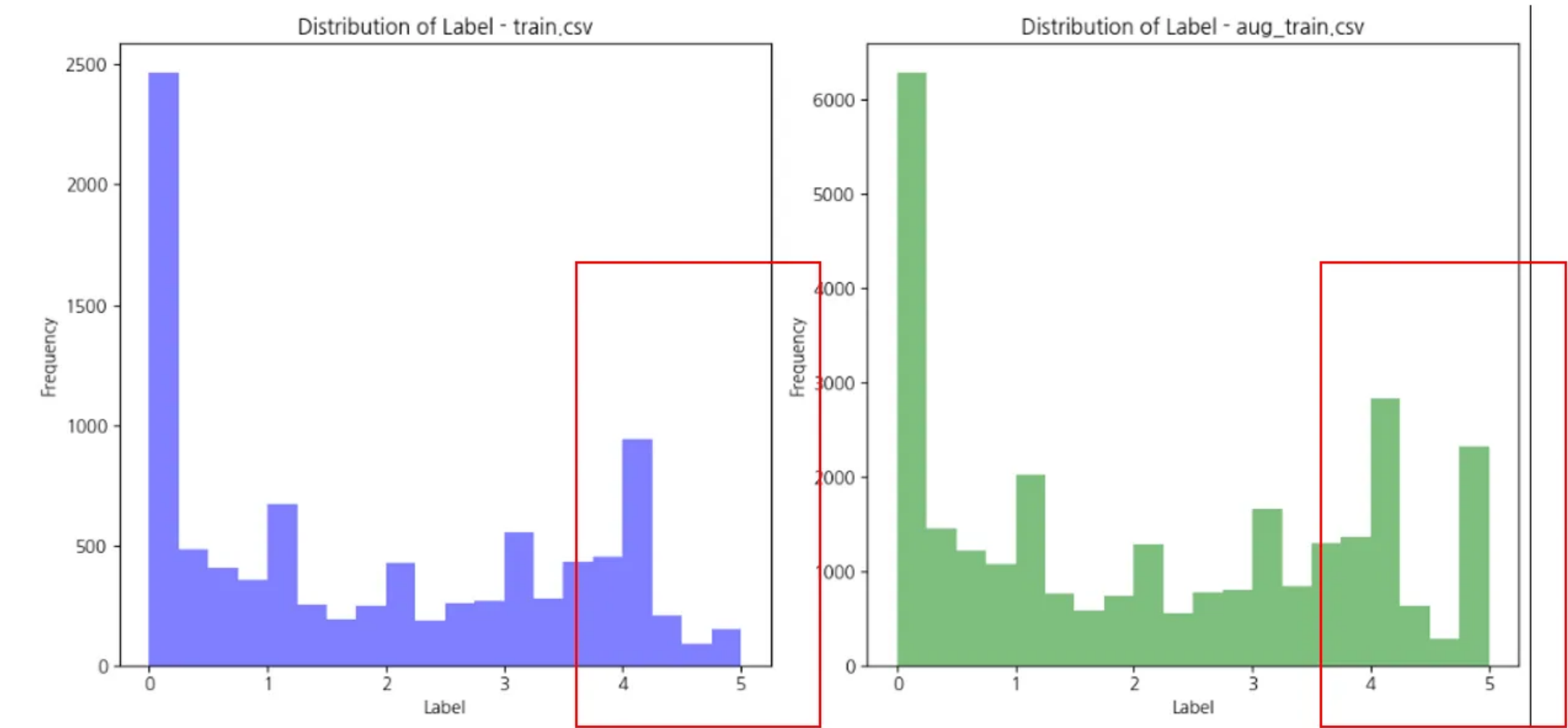
증강으로 인해서 본래 데이터셋과 비해서 두 문장의 유사도와 문장 쌍의 길이 차이에 대한 분포 변화가 확인된다.



원본 데이터에서는 문장 쌍 간의 문장 길이 차이가 꽤 존재하는 편이지만 증강 후에는 문장 쌍 간의 길이 차가 원본에 비해서 현저히 감소한 걸 볼 수 있다. 즉, 원본 데이터에 비해서 증강된 데이터의 분산과 변동성이 낮다.



original_train_V1는 다양한 길이 차이 값에 고르게 분포되어 있으며, 길이 차이가 큰 데이터도 일정량 포함되어 있다. 반면, augmentation_train_V2 는 중앙값 주변의 좁은 범위에 대부분의 데이터가 몰려 있는 경향을 보인다. 이를 통해서 증강된 데이터는 원본 데이터에 비해서 적은 변동성과 분산으로 과적합 방지, 일반화 능력 향상, 모델의 정확성 증가 등의 효과를 기대할 수 있다.



Copied Label 0 Sentence_A → Pasted Sentence_B → Edited label score → 5

왼쪽 그래프처럼 원래 훈련데이터는 라벨 0에 비해서 라벨 5 데이터가 현저히 적었다. 반면에, 증강 후 라벨 5 데이터가 많아짐으로써 오른쪽 그래프처럼 비교적 분포가 균형이 잡혀감을 알 수 있다.

그 결과 V2로 학습된 모델은 V1로 학습된 모델에 비해서 성능이 좋아질 것이라고 예상했다.

4. 모델 탐색

4.1. 모델 선정 기준

모델은 Hugging Face의 사전 학습 모델에서 아래의 기준을 따져 적합한 모델 후보군을 찾았다.

- 파라미터 개수를 1B를 넘지 않는 모델
 - 시간이 너무 많이 걸려서 실험하기 어려움
- NLP, STS 벤치마크 점수가 높은 Base 모델을 fine-tuning 한 모델
 - BLEU, KLEU 등의 벤치마크 고려
 - RoBERTa 계열
 - DeBERTa 계열
 - ELECTRA 계열
- 한국어 전용 도메인이 학습된 모델
- 한국어 전용이 아니지만 다국어로 학습된 LLM 모델
- 최대한 많은 Param을 사용하는 모델
- 최대한 적은 Param 대비 높은 Score를 보이는 모델

4.2. RoBERTa 계열 PLM

RoBERTa는 BERT를 개선한 모델이다.

4.2.1. 동적 마스크

Dynamic Masking은 매 epoch마다 하나의 시퀀스에 MASK 토큰을 적용할 위치를 계속 바꾸어주는 기법이다.

- 일반적인 BERT는 MLM에서 1개의 시퀀스에 N개 epoch에서 동일한 위치에 MASK 토큰을 정한다.
- 하지만 RoBERTa는 N개의 시퀀스로 복제하여 각각에 MASK 토큰 위치를 같거나 다르게 조정하고 각 epoch 마다 다른 마스크가 된 시퀀스를 사전 학습에 사용한다.
- 매번 masking을 새로 하니 학습 시간은 늘어나지만, 매우 폭넓은 성능 향상을 보였다.

4.2.2. FacebookAI/xlm-roberta-large

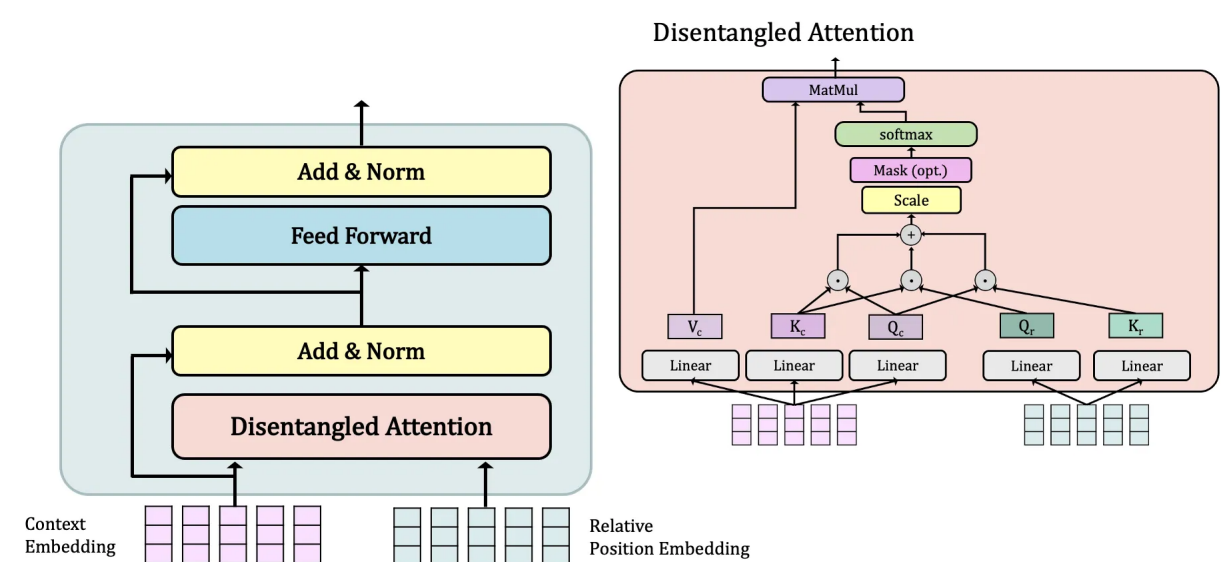
Model	MNLI	QNLI	QQP	RTE	SST-2
roberta.base	87.6	92.8	91.9	78.7	94.8
roberta.large	90.2	94.7	92.2	86.6	96.4
roberta.large.mnli	90.2				

- FacebookAI/xlm-roberta-large는 Facebook AI에서 개발한 XLM-RoBERTa의 대형 버전이다.
- 다국어 언어 모델로, 100개 이상의 언어를 이해하고 처리할 수 있다.
- RoBERTa 계열의 모델보다 더 효율적인 학습이 가능한 XLM 구조를 선택하였다.
 - Zero-Shot, Few-Shot을 지원한다.
- 2.5TB의 큰 학습 데이터를 활용하였다.
- 파라미터 개수 : 561M

4.3. DeBERTa 계열 PLM

DeBERTa는 MS에서 2020년에 BERT와 RoBERTa의 성능을 개선한 모델이다.

4.3.1. Disentangled Attention



- 단어 임베딩 된 토큰과, 위치 임베딩이 된 토큰을 분리한다.
- 위치 임베딩은 각 토큰의 절대 위치가 아닌, 토큰 별로 상대별 위치 인코딩을 사용한다.
- 이 각 토큰을 각각의 트랜스포머 인코더에 집어넣는다.
- 이에 따라, 2개의 Attention Score 벡터를 계산하여 답을 구한다.

4.3.2. 높은 벤치마크 점수

Model	SQuAD 1.1	SQuAD 2.0	MNLI-m/mm	SST-2	QNLI	CoLA	RTE	MRPC	QQP
	F1/EM	F1/EM	Acc	Acc	Acc	MCC	Acc	Acc/F1	Acc/F1
BERT-Large	90.9/84.1	81.8/79.0	86.6/-	93.2	92.3	60.6	70.4	88.0/-	91.3/-
RoBERTa-Large	94.6/88.9	89.4/86.5	90.2/-	96.4	93.9	68.0	86.6	90.9/-	92.2/-
XLNet-Large	95.1/89.7	90.6/87.9	90.8/-	97.0	94.9	69.0	85.9	90.8/-	92.3/-
DeBERTa-Large	95.5/90.7	90.7/88.0	91.3/91.1	96.5	95.3	69.5	91.0	92.6/94.6	92.3/-
DeBERTa-XLarge	-/-	-/-	91.5/91.2	97.0	-	-	93.1	92.1/94.3	-
DeBERTa-V2-XLarge	95.8/90.8	91.4/88.9	91.7/91.6	97.5	95.8	71.1	93.9	92.0/94.2	92.3/89.8
DeBERTa-V2-XXLarge	96.1/91.4	92.2/89.7	91.7/91.9	97.2	96.0	72.0	93.5	93.1/94.9	92.7/90.3

- DeBERTa 계열은 RoBERTa 모델보다 GLUE와 SQuAD에서 높은 점수를 받았다.
- 모델의 크기는 RoBERTa와 비슷하다.
- BERT와 같은 파라미터를 가져도 Kor-STS에서 더 높은 점수를 받았다.

4.3.3. team-lucid/deberta-v3-xlarge-korean

Model	SQuAD 1.1	SQuAD 2.0	MNLI-m/mm	SST-2	QNLI	CoLA	RTE	MRPC	QQP	STS-B
	F1/EM	F1/EM	Acc	Acc	Acc	MCC	Acc	Acc/F1	Acc/F1	P/S
DeBERTa-V2-XLarge	95.8/90.8	91.4/88.9	91.7/91.6	97.5	95.8	71.1	93.9	92.0/94.2	92.3/89.8	92.9/92.9
DeBERTa-V2-XXLarge	96.1/91.4	92.2/89.7	91.7/91.9	97.2	96.0	72.0	93.5	93.1/94.9	92.7/90.3	93.2/93.1
DeBERTa-V3-Large	-/-	91.5/89.0	91.8/91.9	96.9	96.0	75.3	93.1	92.1/94.3	93.0/-	93.0/-

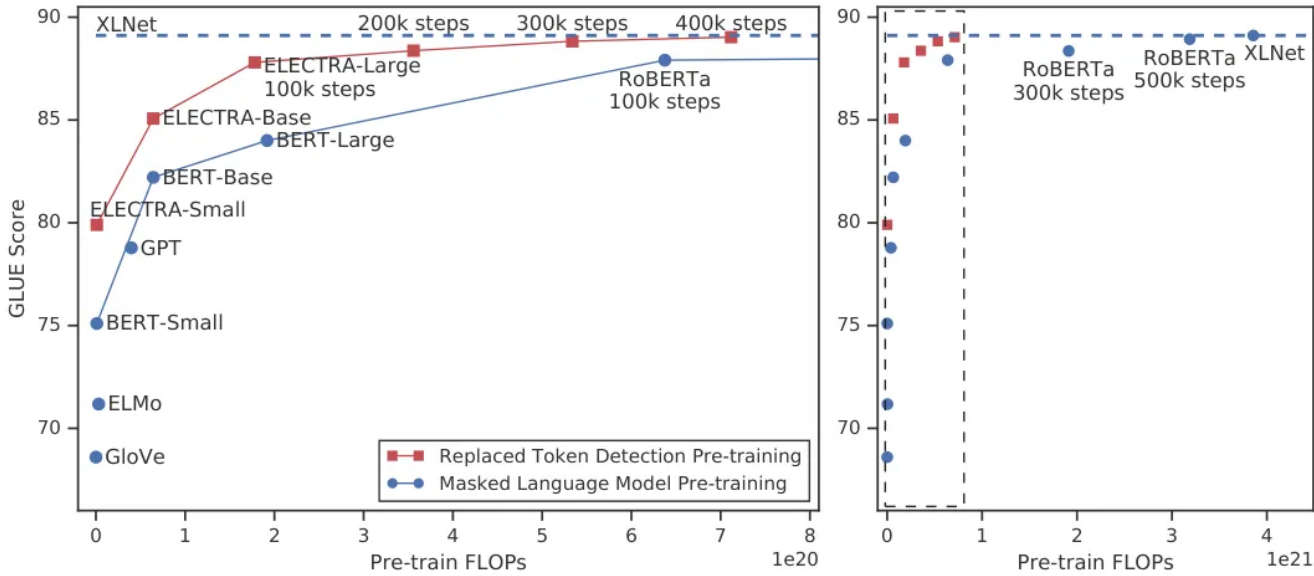
- DeBERTa의 V3 버전을 사용한 PLM이다.
 - MLM 모델링에 사용하여 MASK 토큰을 만드는 것 대신 ELECTRA와 비슷하게 RTD Task를 수행한다.
- 파라미터 개수 : 782M
- 벤치마크를 제공하지 않지만 DeBERTa V3 Large의 304M 파라미터의 비해 큰 파라미터로 학습하였다.

4.3.4. deliciouscat/kf-deberta-base-cross-sts

DeBERTa-Base를 기반으로, STS 도메인 전용의 PLM이다.

- kf-deberta-base는 카카오뱅크에서 제작한 금융, Kor-STS 전용 PLM이다.
- KLUE, Kor-NIL, Kor-STS 데이터셋을 활용함
- STS 문제를 위한 Data Augmentation 적용
- 파라미터 개수 : 186M

4.4. ELECTRA 계열 PLM



ELECTRA(Efficiently Learning an Encoder that Classifies Token Replacements Accurately)는 BERT계열을 대체할 수 있는 2020년에 소개된 트랜스포머 기반 모델이다.

- ELECTRA는 GAN과 비슷한 Generator와 Discriminator를 이용하여 NLP Task를 수행한다.
- 다양한 계산 용량을 수용할 수 있도록 다양한 크기(Small, Base, Large)로 제공한다.

4.4.1. Generator

- Generator는 실제 입력 토큰의 일부를 토큰의 역할을 비슷하게 수행하는 가짜 토큰으로 치환한다.
 - ELECTRA-Generator는 치환하는 토큰 비율이 정해져 있지 않다.
 - 즉 모든 토큰이 가짜 토큰으로도 변할 수 있다.
- 이 기능은 기존 BERT와 같은 MLM 모델이 수행할 수 있다.

4.4.2. Discriminator

- Discriminator는 Generator가 생성한 시퀀스에서 가짜 토큰을 찾아내는 이진 분류를 통해 학습한다.
 - N2N Problem
- BERT는 15%만 마스킹되지만, ELECTRA는 100% 토큰이 마스킹 될 수 있어 더 효과적으로 학습한다.

4.4.3. snunlp/KR-ELECTRA-discriminator

Model	NSMC (acc)	Naver NER (F1)	PAWS (acc)	KorNLI (acc)	KorSTS (spear man)	Questi on Pair (acc)	KorQuaD (Dev) (EM/F1)	Korean Hate- Speech (Dev) (F1)
KoBERT	89.59	87.92	81.25	79.62	81.59	94.85	51.75 /79.15	66.21
XLM- Roberta- Base	89.03	86.65	82.20	80.23	78.45	93.80	64.70 /88.94	64.06
HanBERT	90.06	87.70	82.95	80.32	82.73	94.72	78.74 /92.02	68.32
KoELECTRA- Base	90.33	87.18	81.70	80.64	82.00	93.54	60.86 /89.28	66.09
KoELECTRA- Base-v2	89.56	87.16	80.70	80.72	82.30	94.85	84.01 /92.40	67.45
KoELECTRA- Base-v3	90.63	88.11	84.45	82.24	85.53	95.25	94.83 /93.45	67.61
KR- ELECTRA	91.168	87.90	82.05	82.51	85.41	95.51	84.93 /93.04	74.05

이 모델은 ELECTRA-base의 아키텍처를 기반으로 한국어 도메인에 맞춘 ELECTRA 기반 모델이다.

- 파라미터 개수 : 110M
- 데이터셋 : 위키피디아, 뉴스, 법률 문서, 댓글 등을 포함한 34GB의 한국어 데이터
- KoELECTRA 계열과 마찬가지로 KorSTS Task에서 높은 점수를 보이면서, 다른 Task도 점수가 준수하다.

5. 실험

모델 선택을 완료한 이후, 각 모델에 앞서 조사한 적합한 전처리 및 증강 기법들을 적용하였으며 실험 결과 다음의 특징을 발견하였다.

1. 많은 모델에 점수 향상을 이끌어낼 수 있는 기법
2. 해당 기법 자체로는 변경점이 없지만, 다른 기법의 조합으로 점수 향상이 가능한 기법
3. 많은 모델에 변경점이 없거나, 점수가 오히려 감소한 기법

이 중 1, 2에 해당하는 기법만 모델에 결합하여 체크포인트를 생성하였고, 이것을 적절히 앙상블하여 최상의 결과를 이끌어 내고자 하였다. 다음 기법들은 조사한 것들에서 효과가 좋았거나, 반대로 기대보다 효과가 좋지 못한 몇 개의 기법들에 대한 결과를 소개한다.

5.1. 실험 환경 및 모델

실험은 Validation Dataset에 대한 Pearson을 측정하였다. 앞선 모델 선정에서 우리는 여러 가지 모델을 선정하였고, 많은 모델은 대상으로 다양한 전처리 및 증강 기법을 섞어서, 그렇지 못한 대조군과의 비교를 통해 좋은 기법들을 선정하였다.

그 중 실험에 가장 많이 사용된 모델을 기준으로 기법을 설명할 예정이다. 많이 사용된 모델들은 다음과 같다.

- *snunlp/KR-ELECTRA-discriminator*
- *team-lucid/deberta-v3-xlarge-korean*
- *deliciouscat/kf-deberta-base-cross-sts*

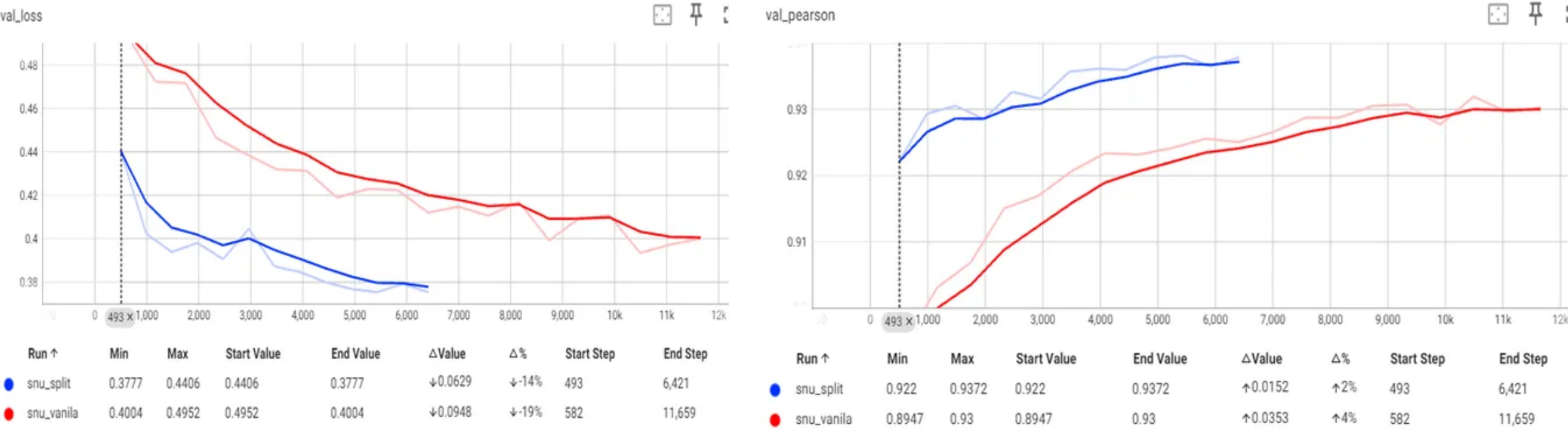
5.2. 점수 향상

5.2.1. 데이터 분할

큰 모델과 다양한 증강기법을 적용한 결과 0.935 이상으로 오르지 않아 과적합 임계점임을 가정하였다.

데이터 분석에 의해 Train 데이터 개수와, Validation(Dev) 데이터 개수의 비율에 많은 차이를 보이는 것을 판단하였다. 이것에 대하여 Train, Dev를 K-Fold Validation을 이용하여 과적합을 최대한 방지하거나, 임의적으로 Train, Dev 데이터 셋을 분리하여 새로운 학습, 검증용 데이터를 생성하는 것이 좋을 것 같다고 판단하였다. 이 실험에서는 후자를 선택하였으며, Train과 Dev csv를 합치고, 이것을 8:2 비율로 나누어 실험에 사용해 보았다.

[8:2 Train-Dev Split]



분할 기법을 통해 다음 모델에 수행한 결과 Validation pearson 점수는 다음과 같다.

- **snunlp/KR-ELECTRA-discriminator 사용** (하이퍼 파라미터는 동일하게 적용)
- Vanilla 모델보다 증강 모델이 Loss와 Pearson 점수가 모두 상승한 것을 확인할 수 있다.

5.2.2. 전처리 조합

[Copied Sentence]

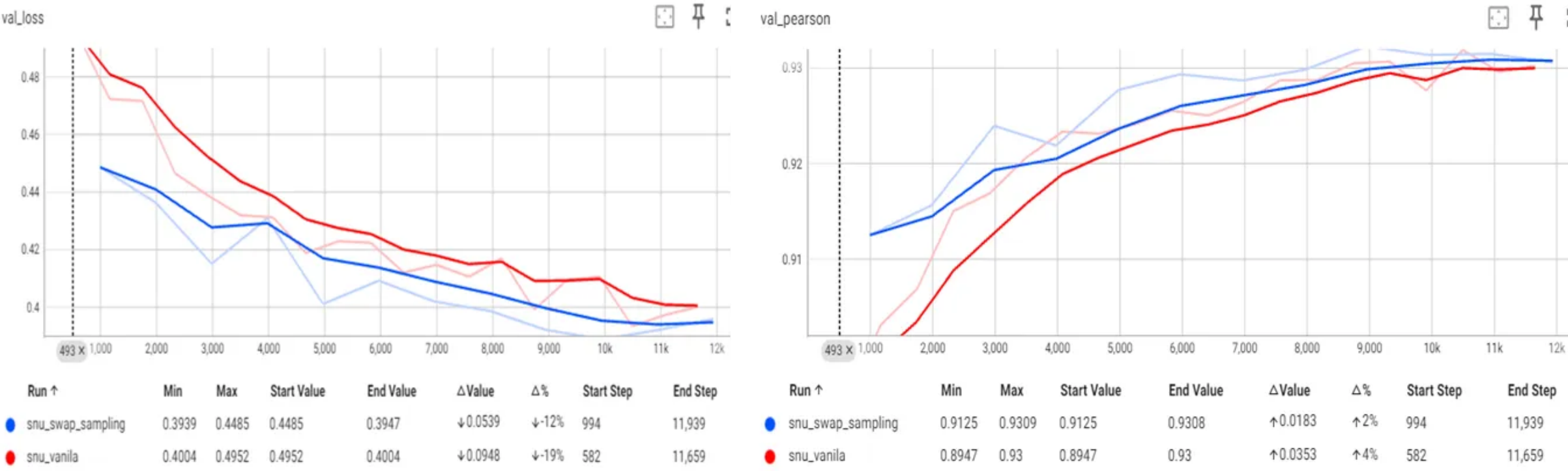
Copied Translation은 데이터 셋의 Label 결과가 특정 분포에 치우쳐 있을 때, 분포가 적은 쪽의 데이터를 복제하여 라벨 분포를 정규화 하는 방법론 중 하나이다.

특히 데이터 분석에서 살펴본 바와 같이, 이 데이터 셋은 라벨 5에 근접할수록 분포가 현격하게 줄어드는 것을 알 수 있고, 라벨 5에 근접하는 데이터는 Sentence 1만 이용하여 생성할 수 있기 때문에 손쉽게 증강이 가능하였다.

[Undersampling]

언더샘플링은 데이터의 분포가 고르지 않을 때 분포가 큰 라벨을 가지고 있는 데이터의 개수를 줄임으로써 분포를 고르게 만드는 기법이다.

Undersampling은 다른 기법과 조합했을 때 더 좋은 점수를 가지는 것을 발견하였다. **snunlp/KR-ELECTRA-discriminator 사용**

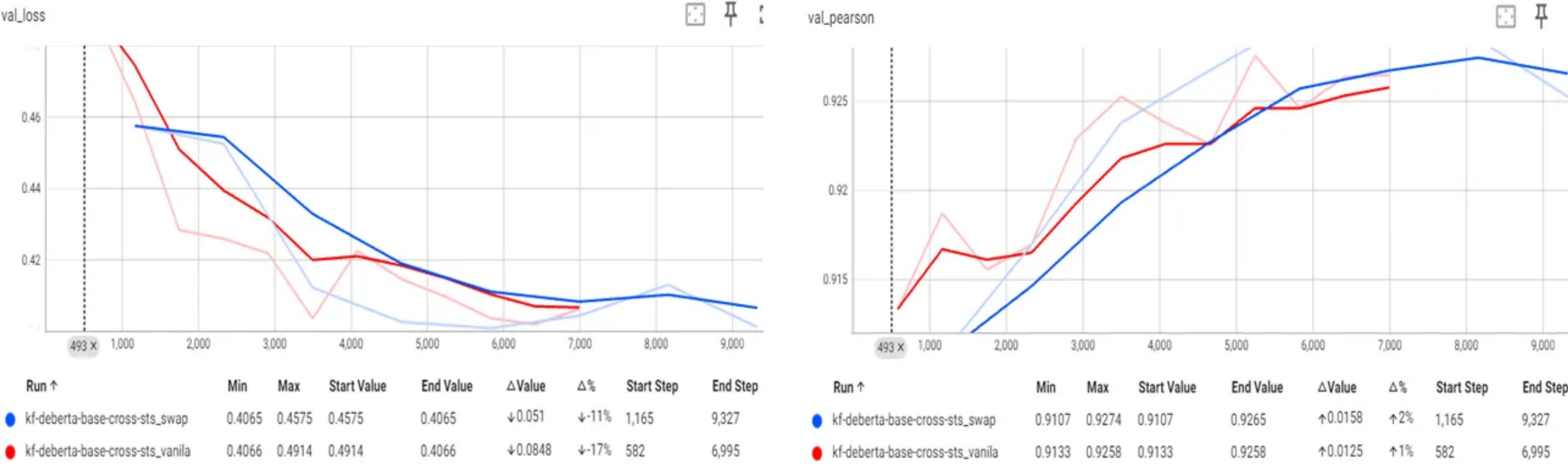


[Swap]

Swap은 STS 문제에서 자주 사용되는 증강 기법 중 하나로, 2개의 문장 위치를 서로 다르게 주어 데이터 개수를 복제한다.

이 기법의 가장 큰 장점은 현 프로젝트에서 존재하는 9300여개의 데이터를 온전하게 2배로 개수를 증강할 수 있는 것이다. 데이터 개수를 Swap을 통해 2배로 늘리고, 실험 결과 과적합되어 있지 않다면 이 복제된 데이터들은 온전하게 학습에 사용할 수 있기 때문이다.

- **deliciouscat/kf-deberta-base-cross-sts 사용**



다만 Swap 기법은 특이하게도 어떠한 모델에서는 오히려 점수가 떨어지는 경향을 보이기도 하였다.

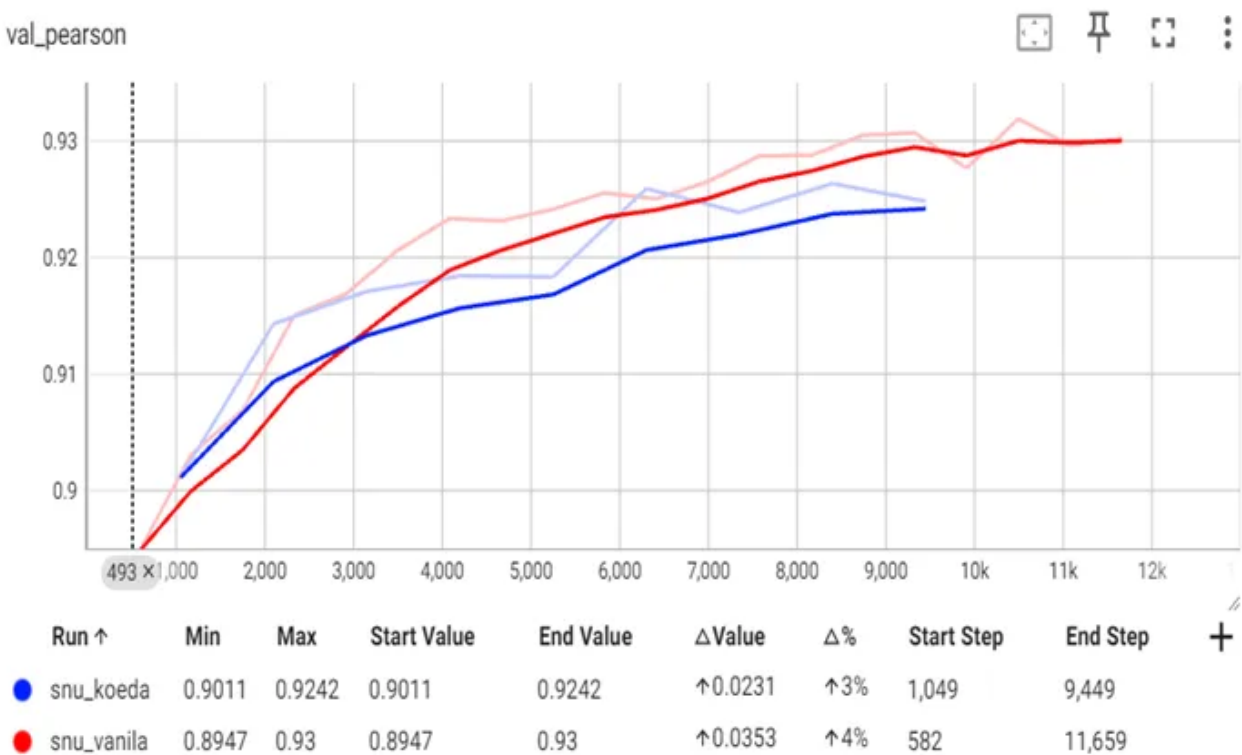
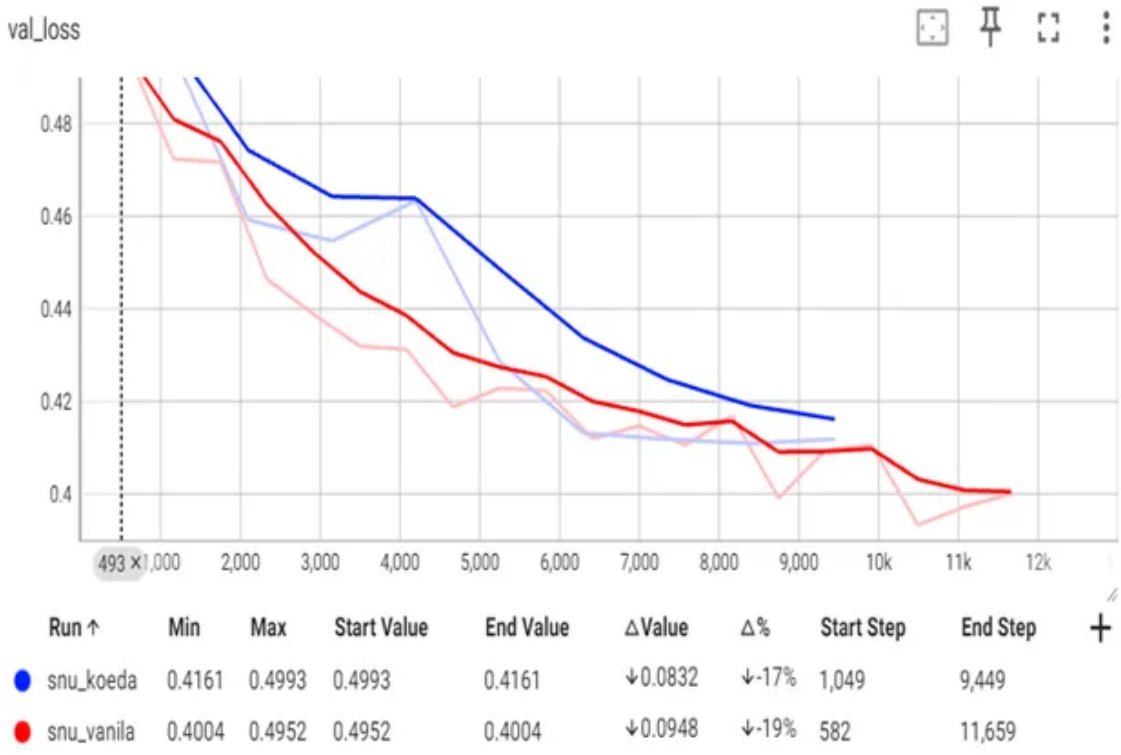
- **snunlp/KR-ELECTRA-discriminator** 의 경우 0.02점 정도 떨어지는 결과를 보였다.

따라서 이 기법은 점수가 높게 나오는 모델들을 실험하여 해당 모델에만 사용하였다.

5.3. 점수 저하

5.3.1. KoEDA

KoEDA는 한국어 문장에 대한 입력이 주어졌을 때 이를 어법적으로 유사한 새로운 문장을 생성해내는 EDA 기법 중 하나이다.



- snunlp/KR-ELECTRA-discriminator 사용
- 하이퍼 파라미터는 동일하게 적용하고, EDA 유무만 다름

증강을 통해 데이터 개수를 키웠지만 오히려 정확도가 감소되는 것을 확인할 수 있었다. 그리고 이 EDA는 위 모델 뿐만 아니라 앞서 소개한 모델들에게도 부정적이나, 영향이 없는 수준으로 작용하였다.

또한 시작 점수는 높았지만, KoEDA를 적용하지 않은 기법보다 빠르게 임계점에 도달하였고, 그 결과 Early Stopping이 적용되어 중지 못한 점수를 보였다.

5.3.2. Bagging

Bagging 은 Ensemble 기법 중 하나이다. 보통 학습할 데이터를 일정 분량으로 분할해서 같은 모델로 학습한 답에 결과를 Soft Voting 해서 진행하고 일반화 능력이 좋다고 알려져 있어 이를 시도해보았다.

[Why?]

새로운 문장을 비슷한 어법에 맞추어 내는 가정이 들어간 것은 좋았지만, 실제 EDA 결과를 대조하면 어법에 유사하지 않고, 문맥에 전혀 상관이 없는 vocab이 들어가는 것을 볼 수 있다.

이렇게 된다면 변형된 Sentence는 기존 STS 점수와 완전히 다른 새로운 점수를 가져야 하는 것으로 보였다.

또한 실험에 불리한 점은 KoEDA 처리가 매번 처리마다 새롭게 생겨나거나 없어지는 토큰이 바뀐다는 점이다. 따라서 운에 의해 실험 점수의 편차가 무작위로 판단하기가 어려웠다.

이 기법을 제대로 확인하기 위해서는 vocab과 문맥 유사도를 서로 학습한 모델을 이용하여 특정 vocab이 들어갔을 때 label 점수를 특정 범위만큼 증감시키는 알고리즘을 추가로 사 용해야 하는 것으로 판단하였다.

6. 모델 선정

model	val_pearson	lr	batch_size	가중치	사용 데이터
upskyy/kf-deberta-multitask	0.9289	1e-5	16	1	augmentation_train_V2
team-lucid/deberta-v3-xlarge-korean	0.9378	1e-5	16	1	augmentation_train_V2
team-lucid/deberta-v3-xlarge-korean	0.9377	1e-5	16	1	original_train_V1
snunlp/KR-ELECTRA-discriminator	0.9325	1e-5	16	1	original_train_V1
snunlp/KR-ELECTRA-discriminator	0.9313	1e-5	32	1	original_train_V1
kykim/electra-kor-base	0.9255	1e-5	16	1	original_train_V1
monologg/ko-electra-base-v3-discriminator	0.9252	1e-5	16	1	original_train_V1
kykim/electra-kor-base	0.9252	1e-5	16	1	augmentation_train_V2
jhgan/ko-sroberta-multitask	0.9249	1e-5	16	1	original_train_V1
FacebookAI/roberta-large-rtt	0.9249	1e-5	16	1	original_train_V1
snunlp/KR-ELECTRA-discriminator	0.9223	1e-5	16	1	augmentation_train_V2

6.1. 1차 모델 실험 - 모델의 기본 성능 테스트

증강과 전처리 없이 기본 데이터 셋에 약 20개의 모델들을 배치사이즈 변화만 주고 pearson 계수를 측정하였다. pearson 계수가 0.92이상인 모델들을 따로 추출하여 6개의 모델로 7개의 후보군을 선정하였다.(모델 1-7) 위 모델들로 앙상블을 진행하여 모델의 개수와 비율에 따라 해당 앙상블 모델의 최고 성능이 어떻게 될지 측정해보았다.

리더보드 제출 기준 27까지 모델 개수를 늘려감에 따라 앙상블을 진행해보았고, 모델의 개수가 늘어남에 따라 pearson 계수가 늘어남을 확인하였습니다. 28번 제출로 모델 7개(1-7) 후보군을 앙상블한 결과 0.9351의 pearson 계수를 얻으며 종료하였다.

해당 실험에서 세운 가설이 두가지가 있었고, 2차 모델 실험에서 해당 가설들이 통하는지 확인했다. 아래 가정은 후보군이 pearson계수가 0.92이상인 모델을 활용하여 앙상블함을 알린다.

- pearson 계수가 좋은 모델들만 평균 앙상블하기보다 pearson 계수의 모델이 살짝 떨어지는 모델들을 섞어서 앙상블 하는 것이 pearson 계수 상승에 더 도움이 될 것이다.
(예- 배치사이즈를 16, 32로 측정한 snunlp/KR-ELECTRA-discriminator을 학습한 모델 1, 2)
모델 1(0.9325)을 2의 가중평균을 두어 측정하기보다 모델 1(0.9325)과 모델 2(0.9313)의 1:1 비율로 측정하는 것이 더 높은 pearson 계수가 산출되었음
- 앙상블의 모델의 개수가 늘어날수록 최종 pearson 계수가 상승할 것이다.
모델 4개를 앙상블한 pearson 계수의 범위 : (0.9080-0.9322)
모델 5개를 앙상블한 pearson 계수의 범위 : (0.9308-0.9346)
모델 7개를 앙상블한 pearson 계수 : 0.9351

6.2. 2차 모델 실험 - 전처리와 증강 진행 후 모델의 성능 테스트

데이터를 증강하여 불균형을 해소하여 기존 20개 이상의 모델들과 새로운 모델들을 탐색하여 pearson 계수가 0.92 이상인 모델들을 추가로 추출하였습니다. 모델 후보군들을 활용하여 다양한 조합의 앙상블을 진행하였다.

1. 모델 1, 2, 16 : snunlp/KR-ELECTRA-discriminator / 모델 4, 14 : kykim/electra-kor-base / 모델 3, 18번 : team-lucid/deberta-v3-xlarge-korean을 통해 같은 모델이지만 다른 pearson 계수를 가진 모델들을 평균 앙상블하였더니 높은 pearson의 모델을 가중평균두어 앙상블 한 것보다 더 높은 pearson 계수를 산출했다.
연역적으로 다양한 분포를 가진 모델을 앙상블하면 성능이 높아지는 경향이 있음을 알 수 있었다.
2. 모델 7개에서 12개까지의 앙상블을 진행해보았다.

모델의 개수가 늘어난다고 무조건적으로 pearson 계수가 늘어나지는 않음을 파악했고 모델 7개 이후 부터는 pearson 계수가 0.934~0.936사이의 값이 도출되었다. 다양한 조합으로 제출한 결과 어느정도 임계점이 있음을 파악했고 37번 제출에 모델1,2,3,4,5,6,7,14,16,18,22 총 11개의 평균 앙상블로 0.9361의 pearson 계수가 2차 모델 실험의 best 결과였다.

7. 앙상블

앙상블 학습은 다양한 모델의 예측을 결합하여 정확한 예측을 도출하는 기법이다. 일반적으로 앙상블을 수행한 경우, 그렇지 않은 경우보다 성능에 유의미한 개선이 이루어진다고 알려져 있다. 이러한 점에 착안하여 앞서 선정된 모델을 기반으로 아래 모델들의 앙상블을 진행하였다.

사용된 모델은 위와 같은 데이터 및 하이퍼파라미터를 사용하여 학습하였으며, dev dataset 기준 피어슨 상관계수를 기록해두었다. Voting을 할 때에는 csv 출력값을 Soft Voting하는 방식으로 사용했다. 주어진 Task가 텍스트 유사도에 대한 점수를 회귀하는 작업에 가까웠기 때문에 각 값을 평균하는 Soft Voting 방식을 활용하였다.

각 모델의 비율은 동일하게 1로 선정하여 가중평균을 계산한 결과, Public Test Dataset 리더보드 기준 0.9361점에 도달하였다.

8. 중첩 앙상블

앞선 앙상블의 성능을 더욱 향상시키기 위해 방법을 고민하던 중, ‘중첩 앙상블’도 충분한 효과를 볼러올 수 있지 않을까에 대한 궁금증이 생기게 되었고, 앙상블된 결과값도 포함하여 우수한 성능을 낸 결과를 다시 앙상블하는 과정을 수행하였다.

이 방법에서는 앞서 8번에서 언급한 앙상블 결과값도 활용하였으며, 그 외 아래에 정리된 앙상블 결과를 활용하여 결과를 도출하였다. 앙상블하는 결과값의 수, 가중치도 조정하며 다양한 실험을 수행하였고, 그 결과 최고 성능을 도출하는 조합을 발견하였다.

🔗 6. 모델 선정 의 표에서 언급한 11개의 모델 외에도 아래 모델을 추가로 활용하였다.

model	val_pearson	lr	batch_size	사용 데이터
sorryhyun-sentence-embedding-klue-large	0.9301	1e-5	16	augmentation_train_V2
FacebookAI/xlm-roberta-large	0.9287	1e-5	16	augmentation_train_V2
deliciouscat/kf-deberta-base-cross-sts	0.929	1e-5	16	augmentation_train_V2
team-lucid-deberta-v3-xlarge-korean	0.9399	1e-5	16	augmentation_train_V2
snunlp-KR-ELECTRA-discriminator	0.9336	1e-5	16	augmentation_train_V2





추가 모델도 함께 활용한 앙상블로 도출된 Public Test Dataset 성능 기준 0.9375 ~ 0.9360 결과를 활용하여 각각 가중치 1씩 부여하고 중첩 앙상블을 진행하였다. 활용한 모델은 매우 유사하지만, 최초 앙상블을 수행하던 당시에 지정한 가중치가 달랐기 때문에 성능이 다양한 결과값에 동일한 가중치를 주고 앙상블을 수행하였다.

중첩 앙상블 결과 Public Test Dataset 리더보드 기준 0.9378점에 도달하였다.





9. 결론

🔗 8. 중첩 앙상블 을 통해, 최종적으로 Public Test Dataset 기준 0.9378 / Private Test Dataset 기준 0.9417의 성능에 도달하였다.

Public 리더보드 기준 전체 16개 참여 팀 중 2위를 달성하였다.

Rank	Team Name	Team Member	pearson	Entries	Final
My Rank 2	NLP_15조		0.9378	113	1d
1	NLP_03조 🏆		0.9413	87	18h
2	NLP_15조 🥈		0.9378	113	1d
3	NLP_16조 🏆		0.9374	41	19h

Private 리더보드 기준 전체 16개 참여 팀 중 3위를 달성하였다.

Rank	Team Name	Team Member	pearson	Entries	Final
My Rank 3	NLP_15조		0.9417	113	1d
1	NLP_16조 🏆		0.9461	41	20h
2	NLP_03조 🥈		0.9446	87	18h
3	NLP_15조 🏆		0.9417	113	1d

10. 프로젝트 회고

10.1 김진재

프로젝트 수행 역할

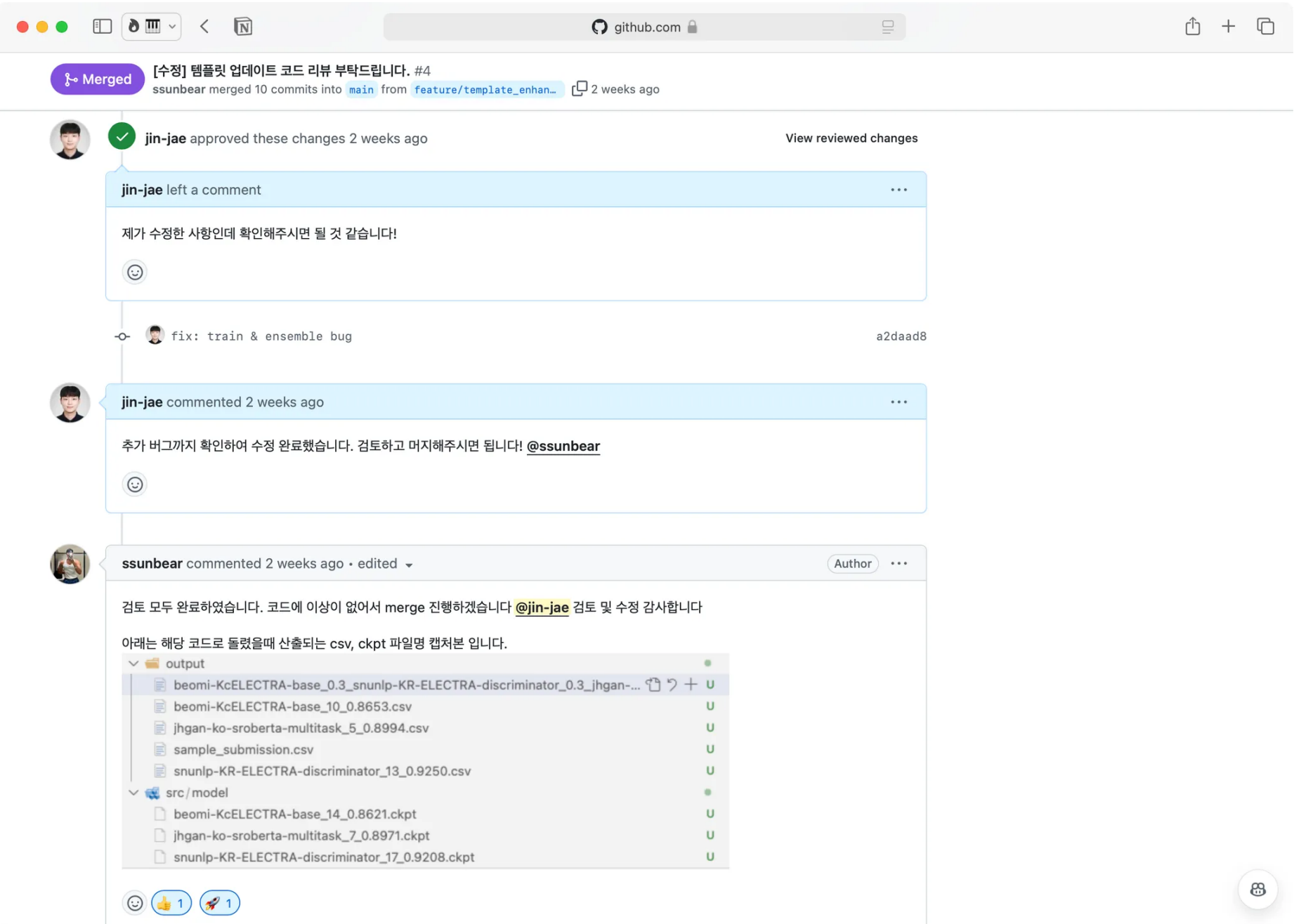
프로젝트에서 수행해볼 수 있는 다양한 방법론을 제안하였다. 리더보드 성능을 개선할 수 있도록 영어로 번역 후 영문 SOTA STS 모델을 사용하는 방법, 임베딩 모델을 찾아 두 벡터 간 차이를 계산하여 이를 반영할 수 있는 모델을 만들 것을 제안하기도 하였다.

또한, 협업 환경 및 베이스라인을 관리하였다. 코드 실행 과정에서 생긴 문제나 어려움을 이슈를 통해 직접 해결하기 위해 다양한 개선점을 고민하였고, 해결한 내용을 공유하였다. 또한, 코드 실행 중 생긴 이슈나 문제에 대해 면밀히 찾아보고 답변하였다. 일부 모델을 탐색하거나 전처리를 수행하고 직접 구동하며 독립변수의 영향력을 확인하였다. 기존 모델 대비 어느 정도의 성능이 나오는지 체크하며 리더보드의 점수를 높이고자 노력하였다.

양상블을 특히 많이 수행하였는데, 이전 모델 대비 어떤 조합과 비율로 모델을 합치는 경우 성능 개선이 가장 크게 이루어지는지에 대해 많이 고민하였다.

성과

다양한 방법론은 시간과 자원의 문제로 도전하지 못했지만, 새로운 접근법과 관점을 제시했다는 측면에서 충분한 의의가 있었다. 또한, 코드 내에서 발생할 수 있는 잠재적인 문제를 분석하고 해결하기 위해 꾸준히 노력하였고, 일부 버그를 수정하며 코드 실행 시 발생하는 오류를 해결하였다. 또한, 중간중간 팀 최고 성적을 경신하는 앙상블 조합을 계산을 통해 발견하여 제안하는 등 리더보드 성적을 올리기 위한 치열한 고민을 하였다.



마주한 한계

API를 사용하지 않기 위해 LLM을 이용하여 영어 번역을 수행하고자 하였는데, 서버의 VRAM 용량이 충분하지 않아 10B 파라미터의 모델을 올리는 것이 원활하게 이루어지지 않았다. 모델을 구축하고 돌리기 위해 외부 서버 등 다른 수단도 함께 활용해야 할 필요성을 느꼈다. 협업 과정에서는 많은 사람들의 동의와 활발한 소통이 이루어져야 했는데, 코드를 합치는 것에 리뷰어를 지정하는 과정에서 어려움이 있었다. 체계적으로 사람을 지정할 수 있었다면 훨씬 빠른 소통이 이루어질 것 같았지만, Private Repository 특성상 다인 지정에 어려움이 있었다. 앙상블을 제출 횟수 내에 무한정 수행하며 다양한 이론을 세우고 결과를 제출하였는데, 성능이 극소량 증가하거나 감소하는 경우가 대부분이었다.

다음 프로젝트에서 해보고 싶은 일

개인적으로 아쉬움이 많이 남아있는 프로젝트이다.

첫 번째로, 앙상블의 결과에 너무 집착하였다. 데이터 전처리와 모델링의 과정을 최소화하고, 모델을 최대한 많이 가져와 조합 비율을 고민하는 과정을 무한정 반복하였다. Private 리더보드가 나오고 각 결과를 분석한 결과, Public 리더보드의 결과는 중요도가 그렇게 높지 않았으며 중첩 앙상블을 수행하다 보니 오히려 답안이 답안 자체에 과적합되어 성적이 낮게 나온다는 점을 느꼈다. 리더보드의 결과에 집착하지 않고 데이터 EDA 및 전처리와 모델링에 많은 시간을 쏟을 것이다.

두 번째로, 협업 과정이 매끄럽게 이루어지지 못했다. 깃허브에서의 이슈나 Discussion이 존재하였음에도 급한 경우 다른 연락 수단 등을 활용하여 대화함으로써 매끄러운 협업 파이프라인이 부족했다는 느낌을 받았다. 다음 프로젝트에서는 조금 시간을 들이더라도 확실하게 프로젝트를 위한 발판을 마련해둔 이후 프로젝트를 진행해야겠다고 느꼈다.

마지막으로, 첫 번째 프로젝트였기에 나름의 부족함을 느꼈다. NLP 도메인에서 리더보드 경쟁에 참여하는 것은 이번이 처음으로, 노하우가 부족하여 프로젝트에 더 도움이 되는 방향을 제안하지 못했다는 점이 아쉽다. 다음 프로젝트에서는 시간을 더 투자하더라도 충분한 학습과 조사를 통해 좋은 결과로 이끌 것이다.

10.2 박규태

무엇을 하였나?

이번 프로젝트에서는 두 문장의 유사도를 측정하는 NLP 테스트를 수행하였다. 프로젝트는 주어진 Train, Dev, Test dataset을 활용하여서 모델을 학습하고 피어슨 상관계수를 메트릭으로 하여 모델의 성능을 측정하였다. 나는 이번 프로젝트에서 개인적으로 두 가지 방향으로 접근을 하였었다. 한 가지는 학습 데이터의 증강을 통한 모델 성능 향상을 지향했고 나머지 한 가지는 앙상블을 통한 결과의 최적화를 노렸다. 증강을 통한 기법은 주로 swapping 을 사용하였고 그 중 특히 Dev 데이터를 swapping 하여 원본 Dev 데이터는 제외하고 swapping 된 데이터만 마찬가지로 swapping 이 진행된 원 훈련 데이터에 추가하여서 훈련 데이터로 사용하여 모델을 학습하는 쪽을 진행하였다. 앙상블 기법에서는 다양한 기법을 시도하고 다른 제출물들과 비교를 해보면서 일부에서 과적합이 된 것 같아 일반화 성능이 좋다는 Bagging 기법쪽을 연구해보았다. 다만 내가 시도한 swapping 방식은 Dev 데이터셋을 단순 swapping 한 것을 훈련 데이터에 넣으면서 모델이 과적합이 되는 쪽으로 진행되었고 Bagging 방식 또한 획기적인 개선을 이끌어 내지 못해 아쉬움이 남는다.

좋았던 점

그 동안 AI 프로젝트에 몇 번 정도 참여한 적이 있긴 하였지만 이렇게 팀을 이뤄 AI 만을 위한 프로젝트에 집중하여 진행해본 것은 처음이었다. 이 과정에서 Git 을 활용한 협업을 해보며 백엔드나 프론트엔드 등이 아닌 AI 프로젝트에서도 Git Flow 을 통해서 Branch 별로 나누어서 진행해보며 AI 프로젝트에서 협업 프로세스가 어떠한 방식으로 진행되는지 알 수 있어서 좋았다.

다음으로 좋았던 점은 그 동안 부스트캠프에서 배웠던 pytorch lightning, matplotlib 등을 프로젝트를 진행을 하면서 자연스럽게 적용을 해볼 수 있다는 것이었다. EDA 를 진행해면서 데이터의 분포 등의 특징을 분석하여서 해당 결과에 따라서 성능을 개선시키고자 시도한 것이나 다른 팀원분들과 함께 기능별로 코드를 패키지를 분리/개선하여서 기능을 구현해보는 과정이 좋았다. 배운 과정을 실제로 구현을 하고 다른 팀원들이 적용한 코드와 기법 등을 나름대로 분석, 적용을 해보면서 실력이 늘었다고 생각한다.

아쉬웠던 점

Git 을 통한 branch 별로 나누어서 모델 별 , 진행한 기법 별 branch 에서 진행하고자 했으나 이 부분은 생각보다는 잘 진행되지 못한 것 같다. 초반에는 특징 별로 branch 을 파서 진행하고자 하였으나 팀원들 간의 Git flow 을 이용한 협업에 대한 익숙함에 차이가 있어서 생각보다 처음 계획했던 것보다는 잘 진행되지 못했고 후에는 대신 자기 개인의 branch를 파서 진행이 되었다.

다음으로는 이번 프로젝트 주간에 추석 기간이 1주일 가량 길게 존재하고 있었어서 이 기간 동안 프로젝트에 열심히 참여하지 못했던 것이 아쉬움으로 남는다. 처음 4일 정도 진행 후에 바로 추석이 끼어서 진행하던 흐름을 끊겨서 진행되지 않았나 생각이 된다.

마지막으로는 wandb이나 Tensorboard 등의 logger 라이브러리의 활용이 적극적이지 못했던 것이다. 이번에는 결과를 바로 바로 뽑아서 빠르게 진행하기 위해서 학습 과정과 평가 과정에서 터미널에 출력되는 validation 값만 보고 진행을 하였다. 이런 방식 대신에 logger 를 활용해 각 epoch 별 loss 와 validaiton 값의 로그를 뽑아보고 분석해보았으면 전체적인 흐름을 볼 수 있어 early stopping의 patience 값 등 하이퍼파라미터를 더욱 개선해볼 수 있지 않았을까 라는 아쉬움이 남는다.

10.3 윤선웅

팀에서의 나의 역할

이번 기초 프로젝트에서 맡은 저의 역할은 협업 환경 및 베이스라인 관리, 데이터 분포 및 재분할 총괄, 모델 탐색, 데이터 증강 실험, 앙상블 코드 작성 및 실험하는 역할을 맡았고, 전반적으로 모든 역할을 담당했습니다. 먼저 30개 이상의 모델들을 배치사이즈를 달리하여 기준 성능을 넘는 모델리스트를 팀 노션 페이지에 정리하여 앙상블에 용이하게 공유하였습니다. 대부분의 실험을 주도적으로 진행하였고, 데이터 증강을 적용하여 모델 성능 향상을 이끌었습니다. 이후 Nested Ensemble 방법으로 실험을 마무리하였습니다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

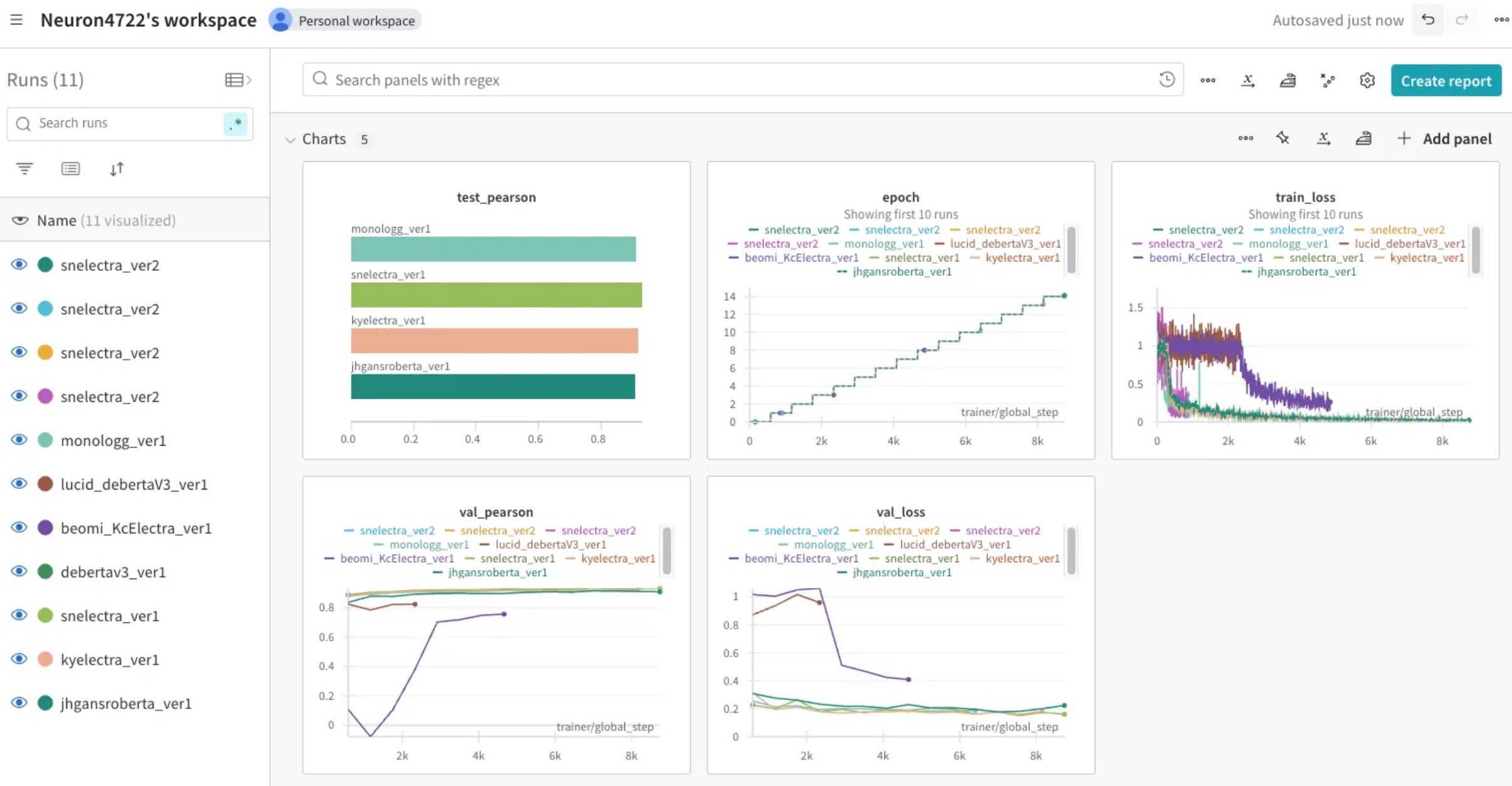
AI 엔지니어로 성장하기 위해서는 데이터와 모델을 다루는 기술이 필요하다고 느꼈습니다. 이번 프로젝트를 통해서 베이스코드 라인을 명확히 이해하고, 모델의 하이퍼파라미터 튜닝과 데이터를 분석하여 성능 향상을 이끄는 방법을 기르는 것을 목표로 삼았습니다. 템플릿화된 베이스코드를 익히고 사용하면서 ckpt의 저장파일명을 정지된 최대에폭수가 아닌 현재에폭수와 pearson 계수를 반영하도록 수정하고 pearson의 차이가 0.001이 3번이상 이뤄지지 않으면 Earlystop이 이뤄지도록 코드를 발전시켰습니다. 개인적으로 wandb를 활용하여 Train data에 최대한 fit하도록 학습을 진행하였고 overfitting을 감소키는 하이퍼 파라미터 튜닝 방식을 활용하였습니다. 이후 어느정도의 기본 모델의 성능을 판단한뒤 임계점에 도달하였고 이후 데이터에 대한 Label 분포를 분석하였습니다. train의 0 라벨의 데이터가 많고 5라벨의 데이터가 적은 데이터의 불균형을 발견하였고 dev의 데이터는 균형있게 이뤄진 것을 발견하였습니다. 라벨 0의 데이터 일부를 copy하여 라벨 5로 만들어 증강을 진행하였고, 라벨 0의 일부를 undersampling하여 증강을 진행하였습니다. 추가적으로 sentence 1과 2를 swap하여 데이터 셋의 개수를 늘려 학습 데이터셋에 활용하여 모델의 성능 향상을 이끌었습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

베이스코드를 익히고자 먼저 여러개의 모델의 학습을 통해 성능을 먼저 도출해보았습니다. 전처리 없는 기본 데이터 셋을 사용하여 30개 이상의 모델의 학습과 앙상블을 진행하여 0.935정도의 pearson 계수가 임계점임을 느끼고 첫 한계에 도달하였습니다.

이를 해결하기 위해 데이터 셋의 분포를 파악하였고 데이터의 증강을 진행하였습니다. 데이터 증강을 통해 다시 모델들을 학습시켜 증강모델과 증강하지 않은 모델들을 앙상블 시켜 0.936정도의 pearson 계수를 임계점으로 느꼈습니다.

두번째 한계를 타파하기 위하여 Nested Ensemble을 진행하였고 제가 세운 가설에 따라 성능향상을 0.9378까지 이끌었습니다. 다만 이 방법론이 맞는지에 대한 의문을 가졌지만 우직하게 80회 가까운 제출 횟수를 제가 사용하였습니다. 여러 csv를 앙상블하여 과적합을 방지할 것이라는 확신은 있었지만, 이 방법론이 엄청나게 pearson 계수가 향상되지 않은 것이라는 것은 알고 있었습니다. 알고 있었지만 마땅히 새로운 방법론을 찾지 못했고 결국 이 방법론을 활용하여 프로젝트를 마무리하였습니다. 1등조의 발표를 들으면서 데이터의 분석이 확실히 부족했다는 점을 알게되었고, 라벨링이 제대로 되어있는지 일일히 확인하지 않았고, 형태소 단위의 분석 또한 제대로 진행하지 않았다는 점이 아쉬웠다고 느꼈습니다. 추가적으로 개인적으로 학습을 판단할때 wandb를 사용하였는데, 베이스코드 템플릿 수정하면서 반영하지 못한점이 팀차원에서는 아쉬웠습니다.



한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

데이터 분석이 제일 중요하다고 느꼈습니다. 저희 팀 차원에서 EDA가 부족하다고 느꼈습니다. 다음 프로젝트에서는 이에 대한 시간을 많이 할애해 프로젝트에서 데이터 분석에 시간을 아끼지 않을 것입니다. 모델링과 하이퍼파라미터 튜닝에 시간을 많이 쏟은 것 같습니다. 다음 프로젝트에서는 모든 과정에 균형을 맞춰 시간을 할애할 것입니다.

협업의 차원에서 이번 프로젝트는 제대로 이뤄지지 않은 것 같습니다. 깃허브의 코드 버전관리에 어려움을 느꼈습니다. 다음에는 깃허브를 활용하여 더 효율적인 discussion이 이뤄질 필요가 있다고 생각합니다. 팀원 개인이 어떤 역할과 task를 하고 있는지 명확하게 알고 있지 않았기에 이에 대한 소통도 부족했다고 생각합니다. 다음에는 개인이 어떤 task를 하고있는지 discussion tab을 통해 효율적으로 소통하기로 하였습니다.

10.4 이정민

무엇을 하였나?

본 프로젝트에서 저는 다음과 같은 역할을 수행하였습니다. 팀장으로 전체 프로젝트 총괄, 모델 탐색, 모델에 대한 증강 기법 및 전처리 실험, KoEDA 증강 실험, K-Fold Validation 실험.

첫 프로젝트를 팀장으로 하면서 많은 부분들을 느꼈습니다. 가장 크게 반성할 점은 첫날에 프로젝트의 시스템적인 내용에 크게 집중한 점이었습니다. 프로젝트를 잘 수행하려면 프로젝트의 내용을 이해하는 것도 중요하지만, 팀장을 좀 더 넓은 Scope의 팀원 관리 및 프로젝트 Repo 관리와 같은 인적 자원을 정의하고 관리하는 것에 좀 더 집중해야 한다는 것을 느꼈습니다. 이 때문에 저희는 좋은 전처리, 모델 들을 발견할 수 있었지만, 이것들을 어떠한 파이프라인으로 테스트하고, 앙상블 한 결과들을 어떻게 정리하고, 최종적으로 Github에 어떤 구조와 Flow로 관리할 것인지에 대한 측면에서 좋지 못한 모습을 보였던 것 같습니다.

모델, 전처리, 증강 기법 탐색에 관해서는 많은 점을 배웠던 것 같습니다. Huggingface를 어떻게 잘 이용하는 지, 각 모델들의 Card에서 어떤 부분을 집중해서 보아야 하는 지 등을 배웠습니다. 또한 GLEU, KLEU와 같은 벤치마크 점수에 따른 모델을 평가하는 법을 배웠습니다. 증강 및 전처리 기법에서는 단순히 이론적인 부분이 아닌 실무적으로 코드를 어떻게 구현해야 하고 실제로 구현함으로써, 다양한 기법에 대한 저만의 BaseCode를 작성하였습니다. 이 기법들은 단순 STS Task가 아닌 다른 NLP Task에서도 사용할 예정입니다.

다만 실험에서는 더 많은 로그를 뽑아야 한다는 것을 배웠습니다. 본 프로젝트에서는 val_loss, val_pearson 점수 2개로 실험을 하였는데, 실제 점수와 이 2개의 로그로 상관관계가 명확하지 않은 실험들이 몇몇 존재했습니다. 따라서 다음에는 pytorch_lightning 및 다른 라이브러리를 통해 더 다양한 로그를 수집하여 실험을 진행 할 수 있도록 노력할 것입니다.

좋았던 점

가장 좋았던 점은 이론으로 배운 ML 사전 학습 및 적용 방법을 실제 프로젝트를 통해 Basecode를 구현하는 것이었습니다. 이 프로젝트의 난이도 자체는 크게 높지 않았으나 처음 프로젝트를 시작함에 있어 가장 중요한 것은 이론을 코드로 어떻게 구현하는 지라고 생각합니다. 첫 프로젝트에서 Basecode에 의존도와 결합도 같은 시스템을 얼마나 잘 구현하는가가, 이후 프로젝트에서 빠르게 초기 단계를 구축할 수 있게 하기 때문입니다.

두번째로는 pytorch lightning과 transformers 라이브러리 등에 친숙해질 수 있는 계기가 되었던 점입니다. 실제 이론에서는 torch를 사용하여 전처리 부터 모델 구현을 Low-Level 에서 구현하였지만, 실무에 가까운 코딩을 보니 전부 라이브러리로 클래스 상속 등을 받아, 원본 클래스에 기능을 추가하는 형태이고, 전처리와 증강과 같은 부분 역시 커스텀하게 작성하여 Call-Back을 하는 방식으로 동작하는 것을 발견하였습니다. 이러한 방식을 도입하여 이미 구현된 부분은 라이브러리로 깔끔하게 호출하고, 원하는 커스텀 기능만 따로 모듈로 관리하니 Clean-Code와 Efficiency를 동시에 챙길 수 있었습니다.

마지막으로는 Git Flow를 통해 Branch를 구분하여 협업을 본격적으로 시작할 수 있었다는 것입니다. 프로젝트는 PoC가 아니라면 결국 단체로 작업하는 것인데, 단체 작업에서 각 작업별 속도와 역량이 다른 시점에서 이것을 하나의 약속된 Basecode로 합쳐나가는 것은 초보자에게 매우 어려운 일입니다. 이러한 부분을 시작하고 팀원 스스로 배워나가기 시작할 수 있다는 것이 좋았다고 생각합니다.

아쉬웠던 점

제일 아쉬웠던 점은 AIStage 서버가 4개이고, test score에 대한 하루 팀별 제출 가능 횟수가 10회로 제한이 되었었던 점이었습니다. 저희 조는 첫날부터 데이터 분석과 동시에 많은 모델 후보군과 증강 후보군을 미리 선정하였습니다. 조합별로 좋은 점수를 가지는 후보군끼리 앙상블을 하려고 하였지만, 이 때 팀원 명수에 비해 작업 가능 서버가 제한되어있었고, 실제 점수를 받아보는 횟수도 제한되어있다 보니, Local 사양이 좋은 팀원은 작업을 Local로 수행하고 나머지는 서버를 이용하는 방식으로 작업하였습니다.

두 번째로 아쉬웠던 점은 파라미터 개수가 큰 PLM을 이용하지 못한 것이었습니다. 일반적으로 Task를 수행할 때 파라미터 개수가 크고 더 깊은 네트워크를 가지는 모델이 상대적으로 더 높은 Task Score를 보입니다. 하지만 실제 Local과 Remote로 모델 학습을 시켜보니 900M이 넘어가는 파라미터에 대해서 학습 속도가 매우 떨어지는 것을 발견하였고, 이 때문에 큰 모델에 대한 테스트를 진행하지 못하였습니다.

마지막으로는 실험에 대한 근거로 많은 로그 데이터를 Tensorboard와 같이 시각화 된 자료를 공유하지 않은 것입니다. 이번 프로젝트에서는 노선을 통해 각 실험에 대한 모델 정보, 증강 정보 및 정확도를 리스트-업 하였는데 이것이 시각적인 측면에서 큰 효과를 보지 못한 것 같습니다. 다음에는 lighthining logs를 하나의 서버에서 같이 공유를 하면서 팀원 전체가 하나의 결과를 토대로 개선해나가는 방법을 서로 모색하는 방향으로 진행하려고 합니다.

10.5 임한택

첫 번째 NLP 프로젝트를 진행하면서 학습 방면으로나 팀원 간의 협업 방식에서 많은 걸 배울 수 있었지만 몇 가지 아쉬운 점도 있었습니다. 아쉬운 점의 대부분은 NLP 강의를 듣고 팀원들과 마지막 논의를 하면서 생겼던 것으로서 다음 프로젝트에 반영시키고 더 좋은 성과를 낼 수 있을 것 같다는 생각이 들었습니다.

배운 점 및 아쉬운 점

1. 깃허브를 통한 협업

깃허브로 팀원들과 작업하는 것으로 효율적인 협업 방식을 경험해볼 수 있었습니다. 몇 주 전, 이고잉님의 깃허브 특강으로 기초적인 깃허브 협업 방식을 배울 수 있었는데 이걸 실제적으로 적용해볼 수 있었습니다. 다소 생소한 단어들과 프로세스가 있었지만 팀원들로부터 도움을 받으며 잘 진행할 수 있었습니다. 이를 통해서 혼자 작업하는 것과 여러 명에서 작업하는 것은 완전히 다른 차원의 작업이라는 걸 깨달았습니다. 특히, 협업 진행 시 디렉터리 구조와 파일명 등을 일관성 있게 구축하고 모든 팀원들이 효율적으로 작업했던 것이 기억에 남습니다. 이 과정에서 하나의 .py 파일에 여러 줄로 코드를 구성하기 보다는 모듈화 해서 파일을 가볍게 유지하는 것이 좋겠다고 생각했습니다.

다소 아쉬운 점도 있습니다. 아쉬운 점으로는 brunch와 feature의 개념을 정확히 이해하지 못하고, Pull과 Push만으로 작업했던 것입니다. 그리고 Issue와 Discussion을 잘 활용하지 못했던 것이 있습니다. 이 두 가지 아쉬운점을 다음 프로젝트에 반영해서, 더욱 효율적이고 활발한 소통이 깃허브 내에서 이루어질 수 있도록 노력할 것입니다.

2. NLP 프로젝트

NLP 프로젝트는 이번이 처음이었습니다. 이번 프로젝트를 계기로 그동안 배웠던 NLP 개념들을 실제 적용할 수 있었습니다. 이전에는 NLP 이론을 제대로 이해하지 못하고 있었지만 실제로 적용하며 학습한 덕분에 더욱 심화해서 이해할 수 있었습니다. 특히 허깅페이스를 배울 때 공유까지 실습할 수 있었습니다. 추후에 깃허브 포트폴리오 관리를 넘어서 허깅페이스 공유까지도 해봐야겠다는 다짐을 하기도 했습니다.

사실 수업에서는 데이터셋 로드, 전처리, 증강, 모델 선정 및 토큰나이저 확인, 평가, 하이퍼파라미터 최적화, streamlit 등 여러가지를 배웠지만 제대로 적용하지 못 했던 것 같아서 다소 아쉬웠습니다. 예를 들어서, 증강은 했으나 데이터 라벨 분포를 고려하지 않은채 증강을 시도했습니다. 전처리를 진행하지 않고 학습을 진행했습니다. WandB sweep을 사용해서 하이퍼파라미터 최적화를 시도하지 못했습니다. 증강 후 필터링을 통해서 데이터셋을 정교하게 다루지 못 했습니다. streamlit을 통해 모델의 예측과 데이터 시각화를 하지 못했습니다. 이렇게 아쉬움으로 남는 것들을 다음 프로젝트에 반영해서 더욱 완성도 있게 진행해보겠습니다.