Wrap Up Report

NLP - 14조 워드 마에스트로

김현서_T7330
단이열_T7333
안혜준_T7349
이재룡_T7417
장요한_T7430

1. 프로젝트 개요

문제 정의

중복된 문장은 글의 가독성을 떨어뜨리는 주요 원인 중 하나다. 문장을 작성할 때, 의미적으로 유사한 문장을 효과적으로 식별하고 수정하는 것이 매우 중요하지만, 이를 사람이 일일이 처리하기는 어렵다. 이번 프로젝트는 Semantic Text Similarity라는 Machine Learning Task를 통해 이러한 문제를 자동으로 해결하는 방법을 연구하는 데 목적이 있다

모델 선정

주어진 데이터셋의 양이 적기 때문에 Siamese Network 기반의 bi-encoder 모델보다는 cross-encoder 기반의 모델이 더 적합하다고 판단했다. 또한, 데이터셋이 한국어로 되어 있어, KR-ELECTRA, KcELECTRA, deberta-korean 등 한국어로 pretrained된 모델들을 초기 후보로 선정하여 실험을 진행했다.

성능 평가 방법

모델의 성능 평가는 문장의 유사도를 0~5점 사이로 예측한 후, 피어슨 상관 계수(Pearson Correlation Coefficient, PCC)를 사용하여 이루어졌다. 피어슨 상관 계수는 예측값과 실제값 간의 선형 관계를 측정하는 지표로, 예측값이 실제값과 얼마나 일관되게 변화하는지를 평가하는 데 중점을 둔다. 이 프로젝트는 정답을 정확히 맞추기보다는, 예측값과 실제값 간의 경향성을 잘 반영하는 것을 목표로 하기 때문에, 회귀 문제로 접근하는 것이 적절하다고 판단했다.

프로젝트 진행 과정

데이터 전처리, 증강, 모델 엔지니어링 등 세부적인 작업은 EDA와 inference 결과 분석을 바탕으로 이루어졌으며, 최종적으로는 모델 성능을 최대한 끌어올리는 방향으로 프로젝트를 진행했다.

프로젝트 구조 및 데이터셋 구조도

```
STS
├── data/                  # input/output
├── model/
│   └── model.py           # model 정의
├── utils/
│   ├── data_pipeline.py   # Dataloader
│   └── utils.py
├── train.py               # 모델 학습 및 저장
├── inference.py           # output.csv 출력
│
├── baselines/             # 실험 세팅
│   └── baseline_config.yml
├── experiments/           # 실험마다 모델 저장
├── lightning_logs/        # 자세한 실험 내역
└── README.md
```

```
data
├── raw/           # Raw dataset
│   └── *
├── custom/        # Customized dataset
│   └── *
├── inference/     # Inference results
│   └── *
├── analysis/      # Data Analysis
│   └── *
└── README.md
```

2. 프로젝트 팀 구성 및 역할

모든 팀원이 프로젝트의 전 과정을 경험할 수 있도록, 기본 구조를 설계한 후 각자 실험을 자유롭게 진행하였다.

이름	작업 내역
김현서	LR_scheduler, early_stopping 작성, 여러 모델 실험
단이열	프로젝트 Baseline 구성, 데이터 EDA 및 inference 결과 분석
안혜준	collate로 동적 패딩 적용, 하이퍼 파라미터 튜닝, 다양한 조합에서의 성능 비교
이재룡	BERT, Word2Vec 등 여러 모델에서의 데이터 증강
장요한	데이터 정제와 BERT를 활용한 데이터 증강

3. 프로젝트 수행 절차 및 방법

프로젝트는 기본적인 구조와 목표를 설정한 후, 팀원들이 개별적으로 다양한 실험을 자유롭게 진행하는 방식으로 이루어졌다. 각 팀원이 자신이 담당한 부분에서 필요한 기법들을 적용하고, 실험 결과를 분석함으로써 유연하게 진행되었다.

초기 단계: 프로젝트의 전반적인 방향과 목표를 설정한 후, 각자 EDA, 모델 구성, 데이터 증강 등 다양한 시도를 통해 프로젝트를 발전시켜 나갔다.

중간 단계: 각자 실험한 내용을 바탕으로 개별적으로 성능을 개선하기 위한 하이퍼파라미터 튜닝, 모델 최적화, 그리고 다양한 증강 기법 등을 적용하였다. 이 과정에서 도출된 결과는 팀 전체의 성능 향상에 기여하였다.

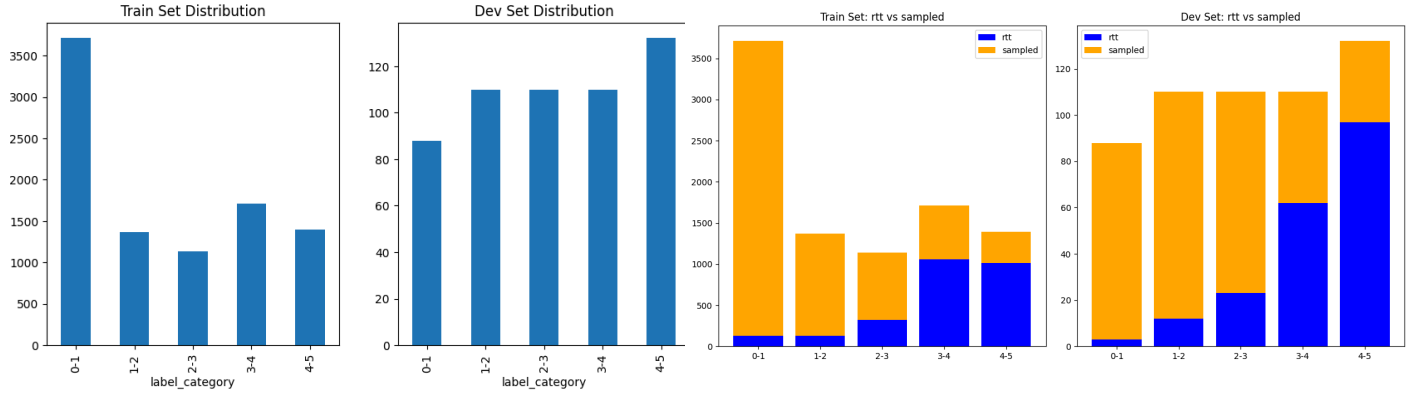
최종 단계: 마지막으로 각자의 결과물을 바탕으로, 여러 모델의 예측값을 앙상블하여 최종 결과를 도출하였다. 이를 통해 개별 모델보다 더 높은 성능을 달성할 수 있었다.

프로젝트 수행 과정에서는 통일된 절차보다는 다양한 접근과 실험이 핵심이었으며, 이를 통해 최적의 성과를 도출하는 데 성공하였다.

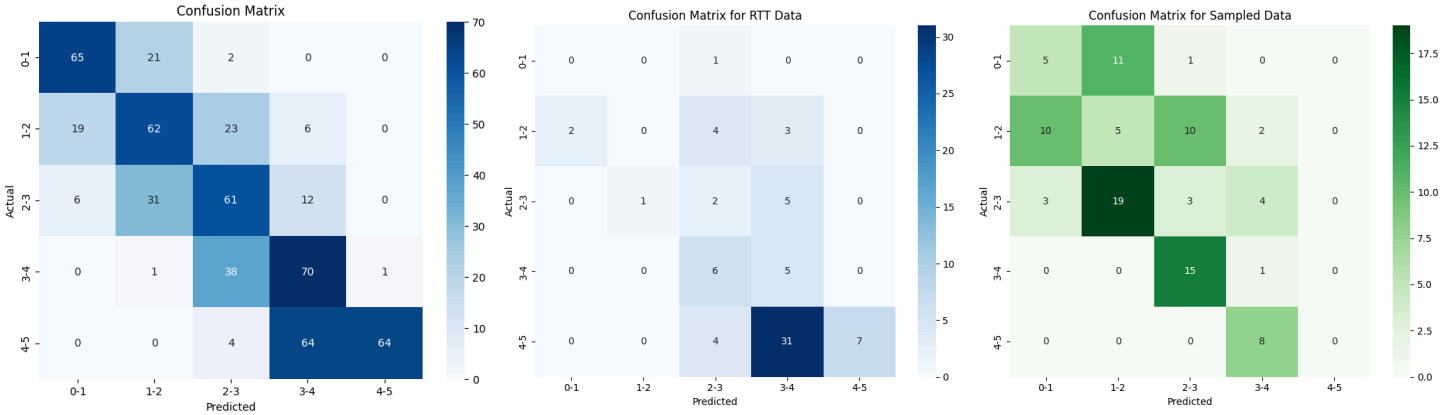
4. 프로젝트 수행 결과

데이터 분석

Train data와 dev data의 분포를 시각화하여 데이터셋 간의 차이를 분석하고, baseline 모델의 inference 결과를 바탕으로 confusion matrix를 도출하여 예측 성능 및 클래스 간 오차를 확인하였다.
각 분석 단계는 주어진 데이터를 보다 깊이 이해하고, 향후 모델 성능 향상을 위한 방향성을 찾는 데 중점을 두었다. 이러한 분석은 다음 단계에서 최적화 및 성능 개선을 위한 근거로 활용되었다.



학습 데이터에서 0-1 범주의 데이터가 상대적으로 많은 것을 관찰하였으나, 데이터의 불균형 문제를 바로 해결해야 한다고 단정짓기보다는, 각 데이터가 학습에 미치는 영향을 직접 확인하는 방향으로 진행하고자 하였다.
또한, 단순히 라벨별로 데이터를 나누는 것에 그치지 않고, source type에 따라 데이터를 더 세부적으로 분류하여 파란색과 노란색으로 시각화하였다. 이를 통해 데이터 분포를 보다 정밀하게 파악하고, 더 깊이 있는 분석을 시도하였다.



Baseline 모델의 inference 결과를 바탕으로 처음에는 source type 구분 없이 전체 데이터를 대상으로 confusion matrix를 생성하였다. 그러나, 단순히 라벨 범위로 만든 confusion matrix는 정확도를 충분히 반영하지 못한다고 판단하여, target과 predict의 차이가 0.5를 넘는 경우를 부정확한 예측으로 규정하였다. 이를 기준으로 부정확하게 예측된 데이터들을 source type(rtt/sampled)별로 나누어 각각의 분포를 확인한 것이 오른쪽 두 개의 matrix다.

이러한 분석 결과를 고려하며 전처리 및 증강 작업을 진행하려고 노력했다.

데이터 전처리 및 증강

앞선 데이터 분석에서 확인된 불균형 문제와 부정확한 예측 데이터의 특성을 바탕으로, 전처리와 증강 작업을 진행하였다. 특히, train data와 dev data의 분포에서 0-1 범주 데이터가 많음을 관찰하였고, source type(rtt/sampled)별로 분포를 나눠 더 깊이 있는 분석을 시도하였다. 이를 통해 불균형 문제를 바로 해결하기보다는, 각 데이터가 모델 학습에 미치는 영향을 분석하고 그 결과를 토대로 전처리와 증강을 적용하였다.

데이터 전처리

```
print(tokenized_origin[1])
print(tokenized_spaced[1])
```

```
['[CLS]', '앗', '제', '##가', '접근', '##권', '##한', '##이', '없', '##다고', '뜹', '##니다',
['[CLS]', '앗', '제', '##가', '접근', '권한', '##이', '없', '##다고', '뜹', '##니다', ';', ';'
```

띄어쓰기 차이에 따라 동일한 단어라도 다르게 토큰화되는 것을 확인하였고, 이는 임베딩 벡터를 부정확하게 만들어 학습에 악영향을 미칠 수 있다는 결론을 얻었다. 이를 통해 띄어쓰기를 포함한 전반적인 데이터 클리닝의 중요성을 인식하게 되었다. 이 과정에서 특수 문자, 이모티콘, 초성 문자와 같은 불필요한 요소들도 제거하는 것이 필요하다고 판단하였다.

데이터 정제를 적용한 후 초기 실험에서는 test pearson 점수가 0.9143에서 0.9182로 소폭 상승하는 긍정적인 결과를 얻었으나, epoch을 늘려 추가 실험을 진행한 결과, 성능 변화는 크게 다르지 않다는 점을 확인하였다. 이를 통해 데이터 정제의 효과가 초기에는 긍정적이었으나, 일정 수준에서는 그 효과가 제한적임을 알게 되었다.

model : kr-electra/discriminator
batch size : 16, lr : 0.00003, epoch : 1

	Test Pearson Score
raw data	0.9143
clean data	0.9182

<epoch를 늘렸을 때 val pearson>

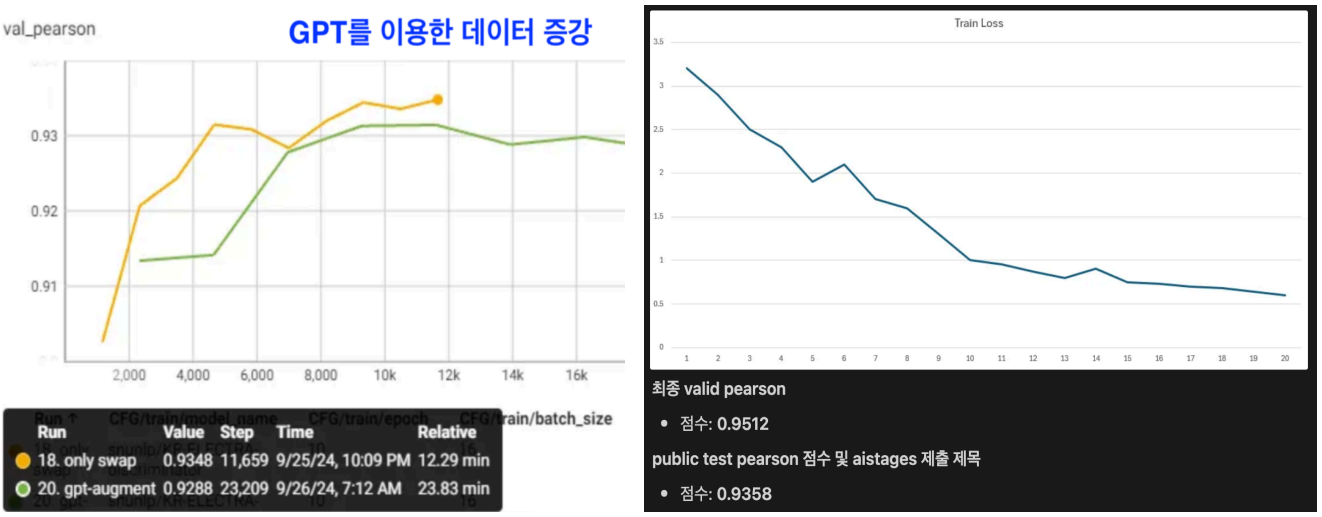
데이터 증강

이번 프로젝트에서는 다양한 방법론을 활용하여 데이터 증강을 시도하였다. BERT를 이용해 문장 내 특정 단어를 마스킹하고 적절한 단어를 예측하여 문장을 변형하는 방법과 Word2Vec을 통해 단어 간의 유사성을 바탕으로 문장을 변형하는 방법을 사용하였다. 또한, GPT와 Electra generator를 활용하여 추가적인 증강 실험을 진행하였다.

BERT/Word2Vec을 활용한 데이터 증강: BERT로 문장 내 특정 단어를 마스킹하고 동의어를 예측해 문장을 변형하거나, Word2Vec을 통해 유사한 벡터를 찾아 단어를 대체하는 방식으로 문장의 의미를 유지하면서 변형을 시도하였다.

GPT를 활용한 증강: GPT 모델을 사용해 새로운 문장을 생성하거나 변형된 문장을 만들어내는 방식을 실험하였다. 이 방법은 기존 데이터의 다양성을 높이고, 새로운 문장을 생성하는 데 중점을 두었다.

Electra generator를 활용한 증강: kr-electra/discriminator와 vocab을 공유하는 generator를 사용하여 학습 데이터에 새로운 문장을 생성하였고, 이를 통해 discriminator 모델이 이전보다 문맥을 잘 이해하게 될 것을 기대했다.



<val pearson이 감소하거나, val pearson은 올라도 test pearson이 크게 감소하였다>

각기 다른 방법론을 통해 데이터 증강을 진행했으나, 대부분의 방법론에서 validation 단계에서 피어슨 점수가 감소하는 문제가 발생하였다. 또한, validation 성능이 올랐어도 test 단계에서는 성능 문제가 지속적으로 나타났다. 이는 증강된 데이터셋으로 학습한 모델의 일반화 능력에 한계가 있음을 보여준다.

이로 인해 팀원들은 데이터 증강이 성능 향상에 항상 긍정적인 영향을 주는 것은 아니며, 특히 고품질 데이터를 생성하는게 어렵다는 한계를 공통적으로 느끼게 되었다. 증강된 데이터는 기존 데이터의 다양성을 높일 수는 있었지만, 그 품질이 기존 데이터만큼 높지 않아서 결과적으로는 모델의 성능 저하로 이어졌다.

모델 개요

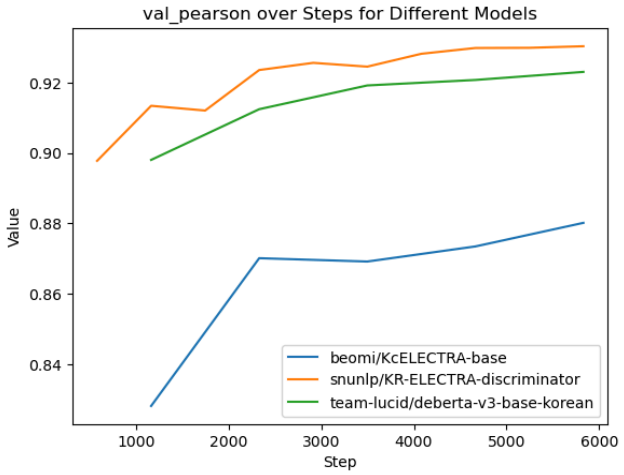
이번 프로젝트에서는 문장 유사도 측정을 위해 다양한 pretrained 모델들이 활용되었다. 모델의 성능을 극대화하기 위해, 각 모델의 특성과 한국어 데이터셋에 대한 적응성을 중점적으로 고려하였다. 주로 다뤄진 부분은 다음과 같다.

- **모델 튜닝 및 최적화**: 각 모델의 특성에 맞는 하이퍼파라미터 튜닝이 이루어졌으며, 학습 속도와 성능을 최적화하기 위한 LR 스케줄러 및 early stopping 등의 기법이 적용되었다. 이를 통해 모델의 과적합을 방지하고, 효율적으로 학습할 수 있는 환경을 구축하였다.
- **데이터 전처리 및 증강**: 모델의 성능을 높이기 위해 데이터 전처리 과정에서 동적 패딩(collate function)이 적용되었으며, 다양한 증강 기법을 사용하여 데이터의 다양성을 확보하였다. 이를 통해 모델들이 불균형한 데이터에서도 안정적인 성능을 유지할 수 있도록 했다.
- **실험 결과 분석**: 여러 차례의 실험을 통해 각 모델의 성능을 평가하고 비교하였다. 특히, 문장 유사도 예측에서 정확도와 일관성을 유지하는 것이 중요한 목표였으며, 이를 위해 각 모델의 예측값 분포와 실제값 간의 차이를 상세히 분석하였다.

각 모델의 특성을 고려한 튜닝과 데이터 처리 과정을 통해 최종적으로 성능을 극대화할 수 있었으며, 실험 결과는 프로젝트 목표에 부합하는 성과를 도출하는 데 기여하였다.

모델 선정 및 분석

일부 후보 모델인 KcELECTRA base, KR-ELECTRA discriminator, debertav3-base-korean에 대해 성능 비교를 진행하였다.



ELECTRA 모델들은 각각 109M의 파라미터를 가지고 있으며, deberta-v3-base-korean 모델은 139M의 파라미터를 보유하고 있다. Validation 단계에서 피어슨 상관 계수 그래프를 분석한 결과, 단일 모델에서는 KR-ELECTRA 모델이 이번 대회의 과제에 대해 가장 우수한 성능을 보여주었다.

이외에도 검증 단계에서 피어슨 상관계수가 높은 모델들을 선정하여 장단점을 분석하였다.

1. ELECTRA Discriminator
장점: 마스킹된 문장을 예측하는 방식으로, 문장이 얼마나 자연스럽게 구성되었는지 확인할 수 있다. 즉, 문장의 자연스러움을 잘 학습한다.
단점: Generator가 의미가 유사한 단어를 제시해도 Discriminator가 틀렸다고 판단할 수 있으며, Generator가 제대로 학습되지 않으면 학습 과정에서 단어들 사이의 유사도를 효과적으로 학습하기 어렵다.

2. DeBERTa
장점: MLM(Masked Language Model)을 통해 단어 사이의 정보를 학습하면서, Disentangled attention을 활용한 positional embedding으로 위치 정보까지 학습한다. 이에 따라 토큰 간 관계를 파악하는 성능이 뛰어나다.
단점: 위치 정보까지 학습하므로 학습량이 많고, 문장보다는 단어 사이의 관계에 집중하기 때문에 문장의 전체적인 구성보다는 단어 간 관계에 중점을 둔다.

3. T5 Encoder
장점: 인코더만 사용하기 때문에 가볍고, STS(문장 유사도)를 분류 문제로 접근했을 때 유용한 정보를 얻을 수 있다. 즉, 유사도를 분류 문제로 해석하는 새로운 관점을 제공한다.
단점: 디코더를 제외한 상태로 학습하므로, 완전한 분류 문제로 해석하지 못해 성능이 상대적으로 저하된다.

모델 평가 및 개선

단일 모델만으로는 성능 향상에 한계가 있다고 판단하여 앙상블 기법을 시도하였다. 분석한 3가지 모델들이 상호보완적인 역할을 할 수 있을 것이라 생각했다. ELECTRA는 문장의 자연스러움을 잘 보장해주고, DeBERTa는 문장 내 토큰들 사이의 관계를 더 정교하게 파악할

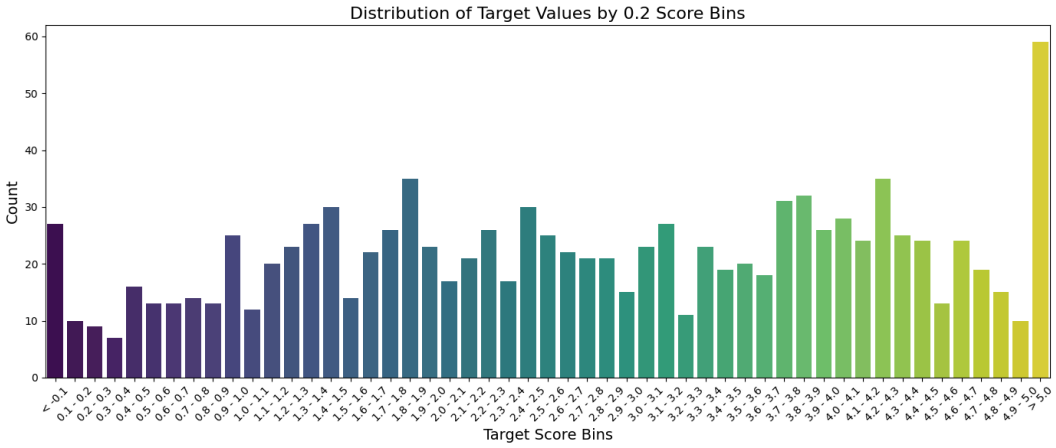
수 있어 상호보완적인 특성을 가지고 있다고 판단하였다. 또한, T5는 STS 작업을 분류 문제로 학습했기 때문에, ELECTRA와 DeBERTa 같은 회귀 기반 모델이 잘 맞추지 못하는 문장에서 더 나은 성능을 보일 가능성이 있다고 판단해 추가하였다.

특히 ELECTRA와 DeBERTa는 성능이 우수하므로, 이 두 모델에 더 높은 가중치를 부여하였고, 비교적 성능이 낮은 T5에는 적은 가중치를 부여하였다.

T5encoder,h6...emble	현서	0.9350 0.9386

각각의 모델 성능은 0.9281, 0.9150, 0.9203이었으나, 앙상블을 통해 최종 성능이 0.9350으로 향상되었다. 특히, ELECTRA와 DeBERTa 모델이 높은 가중치를 부여받아 성능 향상에 큰 기여를 했으며, T5는 적은 가중치를 부여받았지만 상호보완적인 역할을 하여 최종 점수에 기여하였다.

최종 제출



예측값 분포

프로젝트의 마지막 단계에서, 팀원들이 제출한 output.csv 파일의 예측값들을 모아 앙상블을 진행하였다. 각 모델의 피어슨 점수를 기반으로 성능이 높은 모델을 선별하고 예측값의 분포를 확인하여 가중합을 적용하였다.

선별된 모델들 중에서 가장 높은 피어슨 점수를 기록한 모델의 분포를 기준으로, 성능이 유사한 모델들을 가중치로 반영하여 최종 앙상블 모델을 구성하였다. 이를 통해 최종 피어슨 점수 0.9389를 달성하였고, 팀 내에서 가장 높은 성능을 기록할 수 있었다.

모델명	제출자	pearson pearson (최종)
ensemble-team	y	0.9365 0.9389

5.자체 평가 의견

잘했던 점

- 팀원들이 각자 엔드 투 엔드로 프로젝트 진행하여 프로젝트의 구조를 이해했던 점
- EDA를 통해 데이터의 분포를 시각적으로 확인했던 점
- 각각의 모델마다 토크나이저가 어떤 방식으로 작동하는지 확인했던 점
- 프로젝트 베이스 라인 모듈화를 통해 모델의 개선을 공유했던 점
- 각각의 모델들이 STS 작업에 대해서 어떠한 강점을 가질지 고민했던 점
- CLS 토큰만 사용한 것이 아니라, 전체 토큰의 평균 or 가중 평균도 사용해 보려했던 점
- 데이터 증강과 데이터 전처리를 하는 근거를 가지고 시도했던 점
- hugging face에서 모델의 구체적인 코드를 확인했던 점
- 학습 과정에서 loss와 pearson을 시각화했던 점

시도 했으나 잘 되지 않았던 점

- 다양한 데이터 증강 방법을 시도했으나 성능 향상에 기여하지 못했던 점
- 데이터 정제를 통한 데이터 품질 향상 과정에서 성과가 낮았던 점
- Notion을 활용한 일정 조율이 원활히 이루어지지 않았던 점
- Wandb나 Optuna 같은 하이퍼파리미터 최적화 툴 잘 사용하지 못했던 점
- 언더샘플링, 오버샘플링 등의 방법을 사용하여 데이터의 불균형 문제를 해결하려 했으나 성능 향상에 기여하지 못했던 점
- 과적합을 간과하여 학습 데이터에서 매우 높은 성능을 얻었으나 테스트 데이터에서 성능 저하를 일으켰던 점

아쉬웠던 점

- 팀원 간의 코드 공유 및 개선이 잘 이루어지지 못했던 점
- 실험을 진행함에 있어 예측했던 성능을 얻을 수 없었던 점
- 프로젝트 진행에 있어 일정관리가 잘 이루어지지 않았던 점
- 이론적 기반을 바탕으로 체계적인 실험을 진행하지 못했던 점
- 실험의 설계부터 구현까지의 과정이 매끄럽지 못했던 점
- 실험이 구조 변경 없이 표면적으로만 이루어졌던 점
- 여러 하이퍼파라미터의 탐색 시 로깅이 제대로 이루어지지 않았던 점
- 실험 간 데이터 전처리 과정에서 일관성이 부족했던 점
- 모델 선택 및 조합에 있어 충분한 데이터가 확보되지 못했던 점
- 성능 개선을 하기 위한 시간이 전체적으로 부족했던 점

프로젝트를 통해 배운 점 또는 시사점

- pytorch lightning, hugging face의 transformer 라이브러리 등
  AI 관련 라이브러리의 사용법을 알게 되었음
- 여러 레퍼런스 코드를 읽어보면서 코드 읽는 실력이 증가하였음
- 베이스 라인 모듈화를 통해 프로젝트에서 코드를 효율적으로 사용하는 법을 알게 되었음
- 전반적인 AI 프로젝트 과정과 문제 해결 방법에 대해 알게 되었고
  과제 해결 방안에 대한 전체적인 이해를 배웠음
- Github 사용을 통해 프로젝트 내에서 팀원과 협업하는 방법을 알게 되었음

5. 개인회고

개인회고 - 김현서

먼저 STS 작업에 대한 모델을 설정하기 위해서 SOTA에 올라와 있는 모델들의 성능을 확인하였다.
그 다음엔 SOTA의 성능 좋은 모델들에 대해서, hugging face에 한국어 dataset으로 pretrained 되어있는지 확인하였고, 모델들의 논문을 간략하게 읽어보았다.
마지막으로 각각의 모델이 가지는 방법론에 따라서 서로 다른 모델들이 어떻게 상호작용할지 고민해 보았고, 최종적으로 사용하기 위한 모델들을 선정하였다.

이렇게 선정된 pretrained model에 대해서, 가지고 있는 train data를 통해 잘 fine-tuning 시키기 위해 여러 시도를 해보았다.

데이터 증강에 대한 방법론으로는 swap sentence와 swap sentence를 통해 증강시킨 train_data를 단순하게 두배로 증강시킨 방법을 사용하였다..
단순하게 train_data를 두배로 증가시킨 이유는 batch_size를 작게 부여한다면, 하나의 batch 내에 같은 문장이 들어올 확률 자체가 낮아지기 때문에, 이상치로 작용할 확률이 낮아진다고 생각했기 때문이었다.
위와 같은 방법론만을 사용한 근거는 문장 내의 단어가 변경되는 방식으로 데이터를 증강시키게 된다면, 아무리 유사한 단어로 변경시켜도 유사도 점수의 신뢰도가 떨어질 확률이 높다고 생각했기 때문이었다.
데이터 증강은 전체적인 모델의 성능증가를 불러왔지만, 단순하게 증강을 시켰기 때문에 쉬운 문제는 잘 맞추는 대신 어려운 문제는 잘 맞추지 못하는 결과를 불러오게 되었다.

데이터의 전처리는 진행하지 않았는데, 그 이유는 데이터의 전처리가 train data의 숨겨진 고유 정보를 잃을 수 있다고 생각했기 때문이었다.
그러나 이러한 방식은 train data가 매우 많은 self-supervised learning에만 적합한 방식이었고, pretrained model에 대해서 supervised learning을 통해 특정 task에 fine-tuning 시킬때는, 데이터의 성질을 잘 이해할 수 있도록 이상치에 대한 전처리를 해줘야 한다는 것을 뒤늦게 깨닫게 되었다.

그 다음엔 모델의 훈련과정에서 사전학습된 모델을 불러올때, cls토큰만을 이용한것이 아니라, 전체 토큰에 대한 평균과, cls 토큰의 attention으로 전체토큰을 가중평균낸것과, cls 토큰과 전체 토큰에 대한 평균을 concat해보는등 모델에 대해서 여러가지 실험을 해보았다.
그러나 모델의 성능 자체는 크게 증가한 경향이 보이지 않았다.

이번 프로젝트에 대해서 여러가지 방법론을 시도해 보았지만, 각각의 실험에 대해서 기록을 제대로 해두지 않아서 가설검증에 대한 어려움을 많이 겪었기 때문에,
실험에 대한 기록의 중요성을 크게 느끼게 되었다.
특히 처음 프로젝트를 진행했을때의 방향성을, pretrained model의 구조를 변경해 보는것으로 설정하였는데,
pretrained model 을 fine-tuning할때 가장 중요한 부분이 전처리라는것을 깨닫지 못하고, 끝까지 틀린 방향성을 유지했던것이 매우 아쉽게 느껴졌다.

다음 프로젝트에는 데이터의 전처리 부분에 큰 비중을 두고, 여러가지 실험들에 대한 기록을 철저하게 할 것이다.

개인회고 - 단이열

프로젝트 직전에 했던 멘토링이 전체적인 방향을 결정하는데 큰 영향을 미쳤다. 앞서 이전 기수들의 프로젝트 회고를 봤을 때, train data의 imbalance를 지적하는 내용이 많았는데 멘토님께서는 단순히 데이터 통계를 보고 imbalance를 문제로 규정하기보다, 모델이 예측한 결과를 직접 확인해보고 어디에서 문제가 생겼는지를 확인해보라고 조언해주셨다.
Next Step을 정할 때 근거가 뭔지, 그 근거를 찾기 위해서는 분석이 중요하기 때문에 현업에서는 라벨 분석부터 한다고 말씀해주셨는데 그대로 실천하려고 노력한 것 같다. 단순하게 라벨별 데이터 분포만 확인하는게 아니라 source type별로 분포와 그 비율은 어떻게 되는지도 같이 확인하고, inference 결과를 토대로 confusion matrix를 만들어서 오차가 높은 데이터들의 분포가 어떻게 되는지 확인했다. 특히 rtt 데이터들은 그 양상이 라벨과 무관하게 비슷한데 무슨 차이로 정확도가 갈리는지 확인하려는 시도를 많이 했다. rtt 중에서 라벨이 0-1인 데이터들만 spacing해서 dataset을 바꿔보기도 하고, spacing을 했을 때 토큰이 어떻게 변화하는지 직접 관찰했다. 사실 임베딩 벡터의 변화와 attention map의 구성까지 찍어보고 싶었지만, 이 부분은 내 실력이 부족해서 구현하지는 못했다. 추가적으로 sampled의 데이터 중 target 라벨이 4 이상인 데이터들은 rtt와 비슷한 양상을 띤다는 생각도 들었지만, sampled 데이터 중에서 4점 이상의 데이터들은 대체적으로 정확도가 높은 편이어서 우선순위에서 밀린다는 생각에 확인 과정을 미루다가 결국 제대로 테스트해보지는 못했다.
Spacing 이후에 별다른 아이디어를 못내다가 다른 팀원이 w2v으로 동의어(또는 유의어) 대체를 통해 데이터 증강한 것을 모방하여 electra generator 모델로 데이터 증강을 시도했다. 주로 실험에 쓰는 모델이 kr-electra/discriminator였는데 해당 모델은 원래 ElectraTokenizer로 학습된 모델이지만, 한국어로 재학습할 때는 mecab-ko라는 형태소 분석기로 tokenize를 진행한 것 같았다. 그래서 원래의 electra 모델과 다른 독자적인 vocab을 사용하고 있었고, 이와 같은 vocab을 사용하는 kr-electra/generator는 문장의 문맥을 이해하는 방식이 discriminator와 비슷할 거라는 생각이 들었다. 어떤 문장이 주어졌을 때 동일하게 tokenize되고 동일한 vocab을 기반으로 임베딩 벡터를 뽑아내기 때문에 다른 모델들에 비해 임베딩 스페이스가 유사할 것이라는 가정이다.
그래서 일단 아무것도 학습시키지 않은 kr-electra/generator 모델을 가져와서 유의어를 대체하는 방식으로 데이터 증강을 하고 discriminator에 넣어봤는데, 성능면에서 차이가 발생하지 않았다. 증강한 데이터들을 직접 살펴보니 특정 토큰들이 유의어가 아니라 #이나 , 같은 문장의 원래 의미를 해치는 토큰으로 바뀌어 있었다. Generator가 train set 문장들에 익숙해지면 유의어 생성 능력이 높아지지 않을까 하는 생각으로 train data의 sentence_1과 sentence_2를 하나의 train set으로 만들어서, MaskedLM 형태로 불러온 generator에 학습시키고 다시 증강 데이터셋을 생성했다. 이때 valid pearson이 0.924에서 0.93까지 오르는 것을 확인했고, 처음으로 성능 향상을 봐서 아주 행복했다. 증강된 데이터셋에 Sentence Swap까지 적용했을 때, 0.9345까지 성능 향상을 이루었는데, 기분이 좋긴 했지만 앞서 진행한 데이터 EDA 및 예측 결과 분석이 의미 있게 쓰이지 못한 부분이 아쉬웠다. Sentence Swap을 진행할 때, 남들이 해봤는데 좋다더라 정도의 근거 밖에 없었기 때문에 더더욱. 그리고 test pearson을 확인하기 위해 리더보드에 제출해봤을 때 점수가 0.9137까지 내려가는걸 확인했는데, 결국 증강된 데이터셋이 일반화 성능에는 악영향을 끼쳤다는 뜻이다. Train set과 dev set의 임베딩 분포가 비슷한 것 같다는 생각에 test set까지 비슷할 거라고 행복 회로를 돌린게 문제였다.
이번 프로젝트에서는 모델을 트랜스포머 단에서 엔지니어링하지 못한 내 기술적인 한계를 크게 느꼈다. 분명히 토큰의 변화 과정을 보고 임베딩 벡터의 변화까지 트래킹했으면 유의미한 분석을 할 수 있었을텐데, 이것저것 고민하고 우유부단하게 행동한게 독이 되었다. 협업 과정도 미숙했던게 아쉽다. 다음 프로젝트에서는 협업, 실험 기록, 기술적인 도전, 체계적인 분석까지 완성도 있게 해내고 싶다.

개인회고 - 안혜준

이번 프로젝트에서 많은 실패를 겪었다. 다양한 pretrained Model을 테스트하고 성능 비교를 한 것 까진 좋았으나 성능을 높이기 위해 시도한 것들 대부분이 실패를 하였다. 시도했던 항목들은 다음과 같다.

1. pretrained model 위의 head에 hidden layer 추가
   [CLS] 토큰의 768차원의 출력을 바로 1차원으로 보내지 않고 256차원의 layer를 거쳐 예측할 수 있도록 해보았다. 하지만 이경우 모델이 충분하게 학습할 데이터가 부족한 이유에서인지 variance가 더 커졌고 쉽게 overfitting이 일어났다.

2. 데이터 전처리 - 띄워쓰기 처리
   dataset을 확인했을 때, 띄워쓰기가 제대로 안돼있는 문장이 다수 존재하였다. 이 부분을 처리하면 성능이 좋아지지 않을까 하였지만, 오히려 성능이 내려가는 상황이 발생했다. 전처리에 따라 tokenizer가 변환한 token의 형태가 달라지게 되었을 것이고 token의 입장에서 좀 더 분석 하였다면 성능을 올릴 수 있지 않았을까 생각한다.

3. 데이터 증강 - gpt augmentation
   variance가 높은 이유로 학습할 충분한 데이터가 존재하지 않는 점이 아닐까 생각하여 진행하였다. 전체 데이터를 대상으로 의미가 같지만 다른 문장을 만들게 하여 데이터를 2배로 늘렸다. 하지만 성능이 유의미하게 내려갔고, 도입하지 않게 되었다. 증강된 문장들을 확인해 보았을 때, 사람이 보기에는 같은 의미의 문장이지만 dataset 분포에서의 점수까지 동일하지않을 수 있다고 생각했다. 증강할 때 두개의 문장 모두 변화를 주었는데 한 문장은 고정하고 다른 하나의 문장만 변화를 주는게 어땠을 까 생각이 든다.

4. 하이퍼파라미터 튜닝 - weight decay
   training loss는 계속 낮아지지만 validation에서 쉽게 벽을 만나 성능이 개선되지 않는 점에 착안하여 모델의 varinace를 낮추기 위해 고심하였다. weight decay 값을 약간 높이자 미약하게 나마 성능이 개선되었다.

5. 동적 padding 추가 - collate
   baseline을 따라 개발하였을 때 기본적으로 collate가 설정되지 않아 max_length 160으로 일괄적으로 padding이 들어갔었다. 이부분을 collate function으로 batch 단위로 동적 padding을 추가하게 개선하였고, 학습 시간을 80% 감소 시킬 수 있었다.

6. attention_mask 및 token_type_id 추가
   마찬가지로 baseline에서는 token input id만을 사용하고 있어 attention mask와 token type id를 함께 encoder 모델에 넣도록 개선하였다. 성능이 높아질 것을 기대하였지만 큰 변화는 없었고, 오히려 성능이 내려가는 경우도 발생하였었다.

다양하게 시도하고 많은 실패를 겪은 프로젝트였다. 한정된 시간안에 실패 원인에 대한 많은 분석을 하지 못한 점이 아쉽다. 이번 프로젝트는 협업과 커뮤니케이션이 잘 이루어지지 않았다고 생각한다. 분명 한정된 시간이지만 5명의 팀이 함께 결과를 분석하고 새로운 시도를 역할을 분담하여 도전하고 피드백을 다함께 할 수 있었다면 충분히 좋은 성과를 얻을 수 있을 것이라 생각한다. 이번에는 팀이 함께 움직이지 않았는데 다음에는 한몸처럼 움직여 한정된 시간안에 많은 성과를 얻을 수 있다면 좋을 것 같다.

개인회고 - 이재룡

프로젝트 목표 달성을 위해 학습 데이터 증강을 통한 모델 성능 향상을 시도하였다. 이를 위해 BERT와 Word2Vec 모델을 활용하여 문장 내 단어를 동의어로 대체하는 방식으로 실험을 진행하였다. 구체적으로, BERT 모델을 사용하여 문장 내 특정 단어를 마스킹한 후 문맥에 맞는 동의어를 예측하였고, Word2Vec을 활용해 유사한 벡터 공간 내 단어를 찾아 동의어 후보군을 생성하였다. 또한, 형태소 분석기를 사용해 명사나 조사와 같은 특정 품사만 대체하여 문장의 의미를 최대한 유지하면서도 데이터의 다양성을 확보하고자 하였다.

모델을 개선한 방식은 기본 데이터 외에도 증강된 데이터를 추가하여 학습 데이터의 양을 늘림으로써, 모델이 다양한 표현을 학습할 수 있도록 유도하는 것이었다. 특히, BERT와 Word2Vec의 문맥 정보를 활용하여 동의어를 예측함으로써, 단순한 단어 치환이 아닌 문맥에 적합한 대체를 시도하였다. 이를 통해 데이터의 풍부함을 높이는 것을 목표로 하였다.

이러한 실험의 결과로, 데이터 증강을 통한 학습 데이터의 양적 증가는 이루었으나, 모델 성능 향상에는 기여하지 못했다. 이로 인해, 단순히 데이터 양을 늘리는 것이 성능에 긍정적 영향을 주지 않는다는 점을 깨달았다. 특히, 기존 데이터가 충분할 때 새로 증강된 데이터의 신뢰도와 품질이 낮다면, 모델 성능이 오히려 저하될 수 있음을 경험했다. 실험에서 설계한 데이터 증강은 동의어 대체를 기반으로 한 다양한 증강 방법을 활용한 것이었으나, 이러한 변화가 성능 향상으로 이어지지 않았다는 점에서 기대한 효과를 얻지 못했다.

마주한 한계는 증강된 데이터의 품질 관리가 어려웠다는 점이다. 동의어 대체 방식은 문맥에 맞지 않는 단어를 생성하는 경우가 있었고, 이러한 데이터가 학습에 부정적 영향을 미쳤다. 또한, 형태소 분석을 통해 특정 품사를 대체하였음에도 문장 전체의 의미가 왜곡되는 문제가 발생하였다. 아쉬웠던 점은 증강된 데이터의 품질을 충분히 검증하지 못하고 모델 학습에 반영했다는 것이다. 이 과정에서, 증강 데이터가 오히려 모델의 혼란을 야기하며 성능 저하를 가져왔다는 점이 확인되었다.

이러한 한계를 바탕으로 다음 프로젝트에서는 데이터 증강 방법에 더 신중을 기할 계획이다. 예를 들어, 증강된 데이터를 필터링하거나, 의미적 일관성을 더 잘 유지하는 다른 기법을 적용해볼 것이다. 또한, 기존 데이터의 품질이 높은 상황에서는 불필요한 증강보다는 모델 자체의 구조 개선이나 앙상블, 하이퍼파라미터 최적화에 집중하는 방식으로 접근할 계획이다. 이 과정에서 증강 데이터의 품질을 평가하고, 모델에 미치는 영향을 면밀히 분석할 것이다. 이를 통해 데이터 양이 아닌 데이터 질을 높이는 방향으로 나아가고자 한다.

개인회고 - 장요한

이번 프로젝트를 통해서 지난 기간 배웠던 AI의 지식들에 대한 점검과 처음으로 AI프로젝트를 처음부터 마지막까지 혼자서 해볼 수 있는 기회라고 생각해 최대한 열심히 참여하려고 노력했다. 하지만 생각보다 모델 구조를 변경한다거나 모델에 대한 이해를 바탕으로 수정하기에는 개인적인 배움이 많이 부족하며 그 한계를 느낄 수 있었던 프로젝트였다. 따라서 모델을 직접적으로 수정하는 방식을 선택하기 보다는 데이터에 대한 접근을 통해 문제를 해결하려고 노력했다.

그래서 먼저 데이터를 직접 확인하면서 어떤 부분을 개선하면 괜찮아질지 데이터를 하나씩 확인하였다. 확인한 결과로 데이터들에 이모티콘, 특수 기호, 초성, 지나치게 많은 마침표 및 물음표 들이 있다는 것을 알게되었다. 또 맞춤법은 괜찮았지만 띄어쓰기가 지켜지지 않은 문장들이 많아 파이썬의 정규 표현식을 활용하여 문장에서 불필요한 부분을 없애고 한국어 단어의 간격을 자동으로 조절해주는 pykospacing 패키지를 사용하여 문장의 띄어쓰기를 진행했다.

해당 방법을 통해서 기존 실험 대비 피어슨 점수가 약 0.001 정도 상승하는 효과를 볼 수 있었다. 이를 통해서 데이터를 수정하는 방식도 충분히 효과가 있을 것이라는 생각을 했고 다음 시도로 데이터를 증강 하는 방식을 사용해보았다. 구글링을 통해서 BERT를 이용해 문장 속 랜덤한 토큰에 mask 토큰을 통해 가리고 해당 토큰 부분을 모델이 예측 하는 방식으로 새로운 문장을 만들어 내는 방식을 알게 되었다. 또 한국어에서 부사의 경우 문장에서 핵심적인 의미를 담고 있지 않으므로 문장의 부사를 변경해 새로운 문장을 만들어 내는 방식도 알게 되었다. 이 두가지 방법을 이용해 데이터 증강을 시도하게 되었다. 데이터 증강을 통해 기존 약 9000여개의 데이터를 약 2만개의 데이터로 증강을 하였다.

증강한 데이터를 통해 모델을 학습하였지만 결과가 그리 좋지 못했다. 정제 데이터를 사용하여 학습한 최고 피어슨 계수는 0.9243으로 꽤 높은 점수를 달성했다. 하지만 증강을 이용한 학습에서 피어슨 계수가 0.9195로 오히려 소폭 하락하는 결과가 도출되었다. 해당 부분을 분석해보니 검증데이터의 피어슨 계수가 약 3에폭정도 학습이 진행될 때는 내려갔지만 이후 오히려 피어슨 계수가 상승하는 모습을 보여주었다. 이를 통해 증강한 데이터가 오히려 부정확하여 학습이 오래 진행될 수록 학습에 오류가 발생하고 있다는 것을 알게 되었다.

그 다음으로 데이터의 라벨 데이터를 확인했을때 라벨 값이 0인 데이터가 많다는 것을 알게 되었고 데이터 편향으로 인해 피어슨 계수가 적게 나올 수 있겠다는 가능성을 확인했다. 따라서 라벨 값이 0인 일부 데이터를 활용하여 라벨 값이 5인 데이터로 변경하여 학습을 진행했다. 약 100개 가량의 데이터만 실험적으로 적용하였지만 결과는 0.9211로 준수한 편이지만 정제한 데이터만 모델을 돌렸을 때와 달라지지 못했다.

이후 더 많은 시도를 진행하고 싶었지만 시간 문제상 더 실험을 진행할 수 없었다. 이번 프로젝트를 진행하면서 많이 아쉬운 점들이 많았다. 특히 추석 기간에 프로젝트를 진행하지 못해 많은 시도를 해볼 수 있는 기회를 날린 것 같아 후회를 많이 했다. 그리고 프로젝트를 진행하는 동안 아직 실력이나 지식이 많이 부족하다는 것을 깨닫게 되어 부족한 부분을 보충해야겠다는 생각을 많이 하게 되었다. 제일 부족하다고 느낀 점은 모델의 구조에 대해 아직 익숙하지 않아 코드를 통해 모델 구조가 한눈에 들어오지 않는다는 것이 가장 문제라고 생각이 들어 앞으로 여러 모델들의 코드와 논문을 읽어보면서 모델들의 구조를 확인하는 공부를 가장 많이 해야겠다는 생각을 했다.