

문장 간 유사도 측정 Wrap-up Report

NLP-02조(아이다섯)

서가은 서보성 오원택 이승우 정효정

1. 프로젝트 개요

A. 개요

Semantic Text Similarity(STS) Task는 정보 추출, 질문-답변 및 요약과 같은 NLP 작업에 널리 활용되고 있다. 대회의 목표는 STS 데이터셋을 활용해 두 문장의 유사도를 0과 5 사이의 점수로 예측하는 것이다.

B. 프로젝트 구조

○ 프로젝트 구조도

```
root/
|
|-- module/
|   |-- preprocessing.py
|   |-- augmentation.py
|   |-- freeze_params.py
|   |-- memo.py
|   |-- rdrop.py
|   |-- seed.py
|   |-- sweep.yaml
|   |-- translation.py
|   |-- visualization.py
|
|-- code/
|   |-- train.py
|   |-- inference.py
|
|-- baseline/
|   |-- train.py
|   |-- inference.py
```

○ 데이터

id	source	sentence_1	sentence_2	label	binary-label
boostcamp-sts-v1-train-000	nsmc-sampled	스릴도있고 반전도 있고 여느 한국영화 스테기틀하고는 차원이 다르네요~	반전도 있고,사람도 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	루듯루듯 하네요!!	고옥 실제로 한번 봐어요 루루루~!~!	0.0	0.0
boostcamp-sts-v1-train-005	nsmc-rtt	오마이가뜨지저스크와리스트헛	오 마이 갓 지저스 스크온 이스트 맨	2.6	1.0
boostcamp-sts-v1-train-006	slack-rtt	전 암만 짝어도 까만 하늘.. πππ	암만 짝어도 하늘은 까맣다.. πππ	3.6	1.0
boostcamp-sts-v1-train-007	nsmc-sampled	이렇게 귀여운 쥐들은 처음이네요.***	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 있도록해주세요	4.2	1.0
boostcamp-sts-v1-train-010	slack-rtt	그 책부터 연봉 꺼내봐야 겠어요!	책에서 꺼내야겠어요!	2.4	0.0

data: 총 10,974 문장 쌍 (train/valid/test 각각 85/5/10의 비율)

feature: source(국민청원 게시판, 네이버 영화, 슬랙) / 문장 쌍 / label / binary-label

label: 0~5점 사이의 유사도 점수 (소수 첫째 자리)

binary-label: 2.5점 이상이면 1, 아니면 0으로 변환한 label

Baseline code: PyTorch-Lightning 사용

평가 지표: Pearson 상관 계수

C. 프로젝트 환경

컴퓨팅 환경: 인당 V100 서버 할당, VSCode와 SSH 연결해 사용

모듈: PyTorch, PyTorch-Lightning, Hugging Face

협업 환경: Notion, GitHub

의사 소통: 카카오톡, Zoom

2. 프로젝트 팀 구성 및 역할

- 서보성: 팀장(PM), 데이터 전처리
- 오원택: 모델 리서치
- 이승우: 서기, 데이터 전처리
- 정효정: 데이터 전처리
- 서가은: 모델 리서치

3. 프로젝트 수행 절차 및 방법

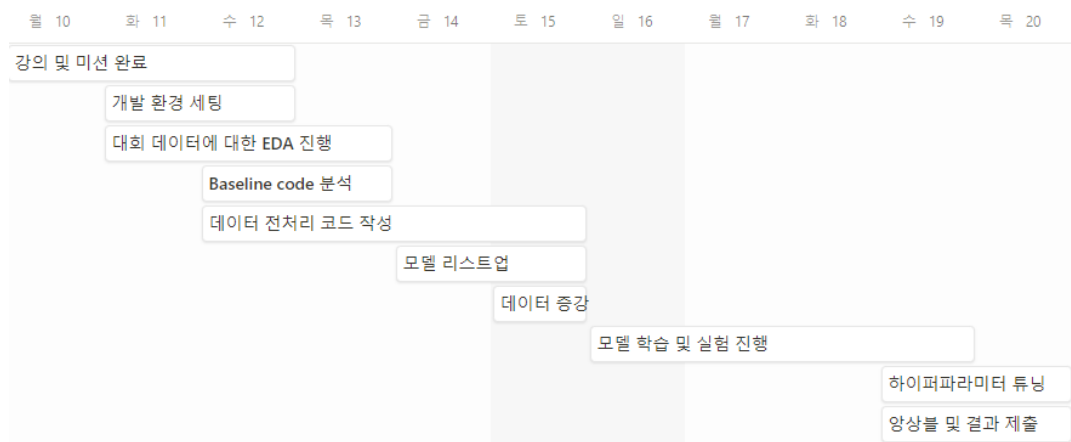
A. 팀 목표 설정

(1주차) 강의 전부 듣기, EDA 진행, Baseline code 분석, 데이터 전처리, 모델 리서치

(2주차) 모델 실험, 결과 도출 및 비교, 최종 제출

B. 프로젝트 사전기획

(1) Time Line 수립



(2) 협업 문화

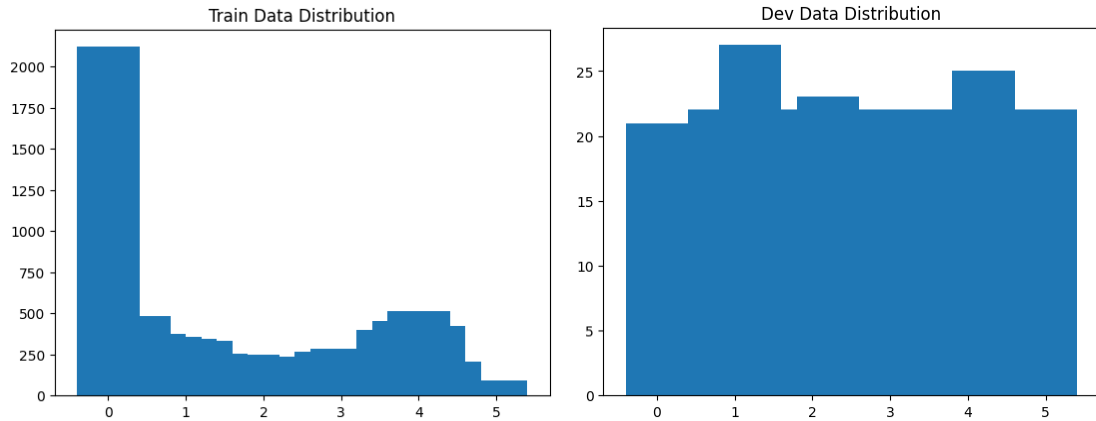
[1] Peer Session 및 정기적인 회의를 통해 역할 분배, code merge 및 실험 결과, 의견 공유

[2] Github를 이용한 code 공유

4. 프로젝트 수행 결과

A. Data analysis and Hypothesis

(1) 데이터 분포



Train data의 label 값의 평균값은 약 1.85, Dev data의 label 값의 평균값은 약 2.58로 두 데이터 간의 분포가 다르다. 특히 Train data의 경우 0.0인 label 값이 많아 쏠려있기 때문에 모델의 훈련이 한쪽으로 편향될 수 있다.

(2) Tokenize

Tokenizer마다 차이는 있지만, 'klue/roberta-base tokenizer'로 tokenizing한 결과 모든 문장들은 평균적으로 약 14.7개의 token을 갖고 있고, 가장 많은 token을 가지고 있는 문장도 90개 이상의 token을 가지지는 않았다. 따라서 모델의 최대 token 길이를 조절해 학습 시간을 줄일 수 있을 것으로 예상된다. 또한 [UNK] token의 비율이 문장 당 약 0.02개로 학습의 큰 영향을 줄 정도는 아니라고 판단되었다.

(3) 특수문자 분석

특수 문자의 비율을 분석한 결과, 괄호, 하이픈 등 연결자가 자주 쓰였고, 웃음을 뜻하는 '^^', ';;' 등의 이모티콘이나 사람 이름을 마스킹한 '<PERSON>'이 자주 등장했다. 또한 'ㅋㅋ/ㅎㅎ/ㅠㅠ' 등과 같이 자음과 모음이 3개 이상 연속으로 나오는 경우가 많았다.

(4) 가설

1. 데이터 증강으로 데이터 분포를 조절하면 일반화 성능이 오를 것이다.
2. 모델에 따라 특수문자가 포함된 pre-trained된 모델의 경우에는 특수문자도 학습데이터로 사용되었기 때문에 이 경우에는 제거 할 필요가 없을 것이다.
3. 중복 문자열을 토큰화하면 'ㅋㅋ' '###ㅋㅋ'와 같은 결과가 나오므로, 길이가 2인 문자열로 축약해도 의미상 문제가 없을 것이다.
4. 맞춤법 검사 API를 이용해 [UNK] 토큰을 일부 제거해 학습에 도움이 될만한 토큰을 늘릴 수 있을 것이다.
5. 주어진 baseline code에서 사용하지 않은 hugging face의 'attention_mask'와 'token_type_ids'를 모델이 사용하도록 처리하면 성능이 향상될 것이다.

B. Preprocessing Data

(1) 문장 정제

2가지 방법을 이용하여 문장을 정제했다. 먼저 가설 2의 증명을 위해 RegExp를 사용하여 한글, 영어, 숫자, 초성(ㅋ 등)만 남기고 모든 특수문자를 제거했으며, 또 가설 3에 따라 'ㅋㅋㅋㅋ'와 같이 2번 이상 반복되는 초성은 모두 두 글자로 치환했다. 또 실험을 위해 괄호 및 하이픈 등 유의미한 특수문자는 유지하는 옵션을 추가해 사용했다.

두 번째로, 가설 4의 증명을 위해 맞춤법 검사 API를 사용했다. 기존 408개였던 [UNK] 토큰이 포함된 문장을 'hanspell' 맞춤법 검사 라이브러리를 이용하여 287개까지 줄일 수 있었다. 이로 인해 해당 방법을 적용하면 성능이 올라갈 것이라고 예측했다.

(2) Data Augmentation

- 역번역

기존에 Google Trans 라이브러리를 이용했으나 배치화를 제공하지 않아 이를 보완하기 위해 사전 학습된 번역 모델들을 사용했다. 'Helsinki-NLP'와

'facebook/mbart-large-50-many-to-many-mmt'의 경우 번역 결과가 좋지 않았고, 따라서 카카오 브레인의 'Pororo' 기반 'Dooly' 번역 라이브러리로 배치화하여 역번역을 진행했다. 총 번역시간이 약 5시간 30분으로 다소 오랜 시간이 걸렸으나 원의미가 어느 정도 유지되었다고 판단해 증강에 이용했다.

- Easy Data Augmentation

EDA 관련 논문¹을 참고하여 한국어에 대해서 EDA 기법을 적용할 수 있는 KoEDA, KorEDA를 사용해 증강을 진행했다. 또한 증강 진행 전 label 값의 분포를 맞추기 위해 label 값을 정수 단위로 범주화하여 확인한 뒤 각 범주가 비슷한 양의 데이터를 가질 수 있도록 했다.

- Token masking & Changing sequence order

Dacon의 코드 유사성 판단대회²를 참고하여 토큰화된 입력값의 일부를 [MASK] token으로 치환하는 방법과 입력하는 문장의 순서를 바꾸어 진행하는 방법으로 데이터의 증강을 진행했다.

C. Model & Learning Method

(1) 모델 선정

한국어 데이터를 이용하므로 한국어 데이터로 학습된 BERT 계열의 모델들을 사용하면 좋을 것이라 생각했다. 따라서 기존에 코드에 사용된 'klue/roberta-small' 외 'beomi/KcELECTRA-base-v2022', 'klue/bert-base', 'kykim/electra-kor-base', 'klue/roberta-base', 'klue/roberta-large' 등 다수의 모델을 실험에 사용했다. 이 중 'beomi/KcELECTRA-base-v2022'³의 tokenizer는 오타나 신조어 등을 잘 tokenize하는 특징을 가지고 있어 성능이 높을 것이라고 예상했고, 실제로 여러 모델들로 실험을 한 결과 일반적인 경우에서 가장 성능이 좋았다.

(2) attention_mask 및 token_type_ids

Baseline Code에서는 모델 학습에 hugging face tokenizer에서 제공하는 3가지 값 중 input_ids만 사용하고 있다. 그러나 attention_mask를 사용하지 않으면 [PAD] token까지 attention되고, 일부 모델에서 token_type_ids를 사용하지 않으면 두 문장을 구분할 수 없어 학습에

¹ J.Wei, and K.Zou, "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), China, Hong Kong, pp. 6383-6389, 2019. DOI:10.18653/v1/D19-1670

² 월간 데이콘 코드 유사성 판단 AI 경진대회, 2022

³ <https://github.com/Beomi/KcBERT>

지장이 갈 것이라고 판단했다. 따라서 가설 5에 따라 두 값을 학습에 사용할 수 있도록 코드 변경 후 진행했다.

(3) R-Drop

Dacon의 코드 유사성 판단대회⁴를 참고하여 **Regularized Dropout**을 진행했다. 하나의 입력에 대해서 순서만 바꾼 두 입력값을 모델에 통과시킨 후, 도출된 두 결과값을 **KL-Divergence Loss**로 최소화 시키면 성능 향상이 있을 것으로 예상했다. 해당 대회는 회귀 **task**이기 때문에 **CERoss**가 아닌 **MSELoss**를 이용하여 적용했다.

(4) K-fold

데이터의 개수가 약 1만 개로 그 개수가 적다고 판단해 **K-fold**를 사용하면 좋을 것이라 생각했다. 강의에서 학습한 **K-fold Cross Validation**을 참고해 학습을 진행했으며, 학습하는 시간이 오래 걸리기 때문에 **K**의 값을 최대 5로 설정했다.

(5) Model Parameter Freezing

Pre-trained 모델은 일반적으로 전반부 레이어에서 **token**의 의미를 포착하고, 후반부 레이어에서 원하는 **task**에 대한 예측을 수행하므로 전반부를 **freezing**한 채로 **fine-tuning** 하는 것이 좋다고 판단했다. 학습시간을 줄이기에도 좋은 방법이었다.

(6) Label Smoothing

Target label의 기준이 애매해 실제 **label** 뿐 아니라 그 근처의 값에도 가중치를 주어 더욱 안정적으로 수렴할 수 있도록 하는 것이 좋겠다고 판단했다. 따라서 **label** 값을 평균으로 하는 정규분포에서의 **log** 확률을 **loss**로 처리하는 **label smoothing**을 적용했다.

(7) Add modules to Pre-Trained Model

보통 **BERT**구조의 모델은 **<CLS>**토큰을 사용하거나, 전체 토큰을 **Mean Pooling**하는 하는 방식으로 활용한다. **<CLS>**토큰처럼 특징적인 토큰만을 더 활용해보고자 **<CLS>**, **<SEP>**, **<EOS>**토큰을 적절히 합쳐서 **pre-trained** 모델 뒷단에 **FCN**, **LSTM**, **CNN** 등 다양한 모델을 추가해보며 실험을 진행했다.

(8) Sentence Bert⁵

기존의 방법과 다르게 두 개의 모델에 각각 하나의 문장만을 넣어서 각 문장에 대한 **Sentence Embedding**을 얻어내고 유사도를 판단하는 방법으로, 각 문장에 대한 의미정보를 담는 방향으로 훈련하기 때문에 실험해 볼 필요가 있다고 느꼈다.

(9) Dropout 및 L2 Regularization 값 강화

여러가지 실험을 진행하면서 **train**, **validation loss**가 언제나 큰 차이의 일정한 수치로 수렴했기 때문에, 그 간극을 줄이고자 **regularization** 방법을 적용했다.

(10) Model Ensemble

다양한 모델의 예측값을 비교하니 한 문장 쌍에 대한 여러 예측값 중 한 두 개 정도만 차이가 큰 경우가 많았다. 따라서 앙상블을 통해 일반적인 값을 최종 예측값으로 정해 분산을 줄이는 것이 성능 향상에 큰 도움이 될 것이라 판단했다. 회귀 **task** 이기에 **soft voting** 방법을 적용했으며, 차이가 큰 경우를 배제하기 위해 각 문장 쌍마다 가장 큰 예측값과 가장 작은 예측값 한 개씩을 제외한 뒤 진행했다.

⁴ 월간 데이콘 코드 유사성 판단 AI 경진대회, 2022

⁵ Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, 2019

D. Hyperparameter

(1) Batch Size

RoBERTa 논문⁶에서 **batch size**의 증가를 통해 더 좋은 성능을 얻은 점을 참고해 **batch size**를 16, 32, 64까지 증가시켜 실험을 진행하였다. 그러나 논문에서는 **document-level**의 데이터를 사용해 큰 **batch size**가 필요했지만, 우리가 가진 데이터는 짧은 문장 단위 데이터였기 때문에 큰 변화를 주지 못했다.

(2) Max Length

BERT의 기본 값은 512이지만, EDA 결과 대화 데이터는 100 token 이하의 데이터로만 이루어져있어 256, 128 등으로 값을 줄였다. 이로 인해 학습 시간이 단축되고 메모리 사용량이 줄어 **batch size**를 늘리거나 **large** 모델을 사용하는게 가능하게 되었다.

(3) Learning Rate & LR Scheduler

여러 **scheduler**를 실험해본 결과 **Step**, **Cosine**, **Onecycle** (작은 값부터 커졌다가 다시 작아지는 **scheduler**) 세 가지의 스케줄러의 성능이 가장 좋았기에 주로 사용했다. **5e-6 ~ 5e-5** 범위에서 **lr**을 변경하였으며 세 경우 다 일반적으로 **3e-5**를 기준으로 훈련할 때 결과가 가장 좋았다.

E. Model Performance

- B-(1) 문장 정제

가설 3, 4에서의 예상대로 일반적인 상황에서 적용했을 때가 그러지 않은 경우보다 성능이 올랐다.

- B-(2) Easy Data Augmentation & Back translation

EDA의 경우 문장이 **noise**를 발생시키고, 문장이 같은 의미를 가지면서 약간만 변형이 되다보니 다른 데이터에 대한 일반화 성능이 개선되지는 않았다. 역번역또한 문장의 원의미가 어느 정도 유지되었으나 같은 의미를 가지는 문장들이 늘어나 일반화 성능 개선에 큰 효과 있지는 않았다.

- B-(3) Token masking & Changing sequence order

EDA에 비해서 두 방법 모두 **noise**를 많이 발생시키지 않으면서 증강이 될 수 있었고, 그로 인해 다른 증강 방법에 비해서 성능이 개선되었다.

- C-(2) attention_mask 및 token_type_ids

가설 5에서의 예상대로 **baseline code**에서 이 부분만 추가해도 성능이 크게 올랐고, 본격적인 실험 전에 추가함에 따라 일반적으로 반드시 성능에 좋은 영향을 줄 것이라 판단해 이후 모든 실험에서 적용했다.

- C-(3) R-Drop

일반화 성능을 향상시키기 위해서 **Rdrop**을 적용했는데, **Rdrop**의 경우 회귀 문제보다는 분류 문제에 더 적합한 방법이다보니 약간의 성능 향상이 되거나 경우에 따라서는 성능이 감소하기도 했다.

- C-(4) K-fold

일반화 성능을 향상시키기 위해서 적용한 방법으로 데이터의 개수가 적을 때 유용하게 쓸 수 있는 방법이다. 하지만 주어진 데이터가 그렇게 적지는 않았기 때문에 약간의 성능 향상만 얻을 수 있었다.

⁶ RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019

- C-(5) Model Parameter Freezing

모델의 앞 부분을 동결했을 때 학습 시간이 크게 줄었지만 **loss**가 비슷한 수치로 수렴했다.

- C-(6) Label Smoothing

대부분의 경우 기존 성능에 영향을 주지 못했고, 별개로 **validation loss**가 심하게 오르는 문제가 있어 사용하지 않았다.

- C-(7) Add modules to Pre-Trained Model

초기화 파라미터 갯수가 늘어나 더 많은 GPU 메모리와 학습이 필요해서, 시간 문제로 사용하지 못했다.

- C-(8) Sentence Bert

문장 임베딩이 어떤 의미를 가지는지 알 수 없어 제대로 검증 할 수가 없었고, 유사도가 매우 낮아서 더이상 사용할 수 없었다.



- C-(9) Dropout 및 L2 Regularization 값 강화

의도대로 **training loss**와 **validation loss**간의 차이를 극적으로 줄이진 못했지만, 실험 결과 **dropout**을 비교적 크게 적용했을 때 조금의 성능 향상을 기대할 수 있었다.

- C-(10) Model Ensemble

여러 모델들의 결과값을 고려했기 때문에 단일 모델만으로 결과를 냈을 때보다 앙상블을 진행했을 때의 결과가 크게 높았다.

- 최종 결과

7 (-)	NLP_02조	Public		0.9268	35	4d
8 (1 ▾)	NLP_02조	Private		0.9328	35	4d

5. 자체 평가 의견

A. 잘한 점들

- Pytorch lightning 기반의 **baseline code**에서 크게 벗어나지 않고 전체 코드를 구성했다.
- 데이터 **EDA**에 다같이 참여하며 데이터를 파악했다.
- 처음에 프로젝트 진행 계획을 자세히 수립하고 진행했으며 이를 잘 지켰다.
- 다양한 방향성을 가지고 다양한 시도들을 연구 및 진행했다.
- 프로젝트 중 주기적으로 코드 **merge** 및 구현 기능 설명 시간을 가지면서 사용하는 코드를 정리했다.
- 가설을 세우며 논리적으로 접근하려고 노력했다.

B. 시도했으나 잘 되지 않았던 것들

- **Code review** 방식을 미리 계획했으나, 모두 처음 진행하는 대회에 익숙치 않아 **code review**를 진행하지 못했다.
- **Code convention** 역시 정해두었으나 **code review**를 하지 않으면서 역시 지켜지지 않았다.

C. 아쉬웠던 점들

- 시도할 방법이 다양하다 보니 그 과정에서 성능이 좋지 않다고 판단되면 바로 포기했다.

- 각자의 진행상황을 실시간으로 알 수 없어 소통이 조금만 덜 돼도 진행하는 실험이 겹치는 경우가 있었다.
- 앙상블에 대한 이해도가 높지 않은 상태에서 진행하다보니 오히려 최종 제출한 앙상블 모델보다 제출하지 않은 앙상블 모델이 훨씬 더 좋은 결과를 냈던 점이 아쉽다.

D. 프로젝트를 통해 배운 점 또는 시사점

- 사전에 실험 계획을 구체적으로 세우고 실험을 시작하지 않으면 실험이 원활하게 진행되기 어렵다는 점을 알게되었다.
- 서로의 작업이 겹치는 경우를 겪으면서 **Code review** 작업의 필요성을 깨닫게 되었다.

6. 개인 회고

5-1. 서가은

우리 팀의 학습 목표

각자 맡은 부분에서 다양한 시도들을 해보며 모델의 성능을 올리는 것.

나의 학습 목표

다양한 모델과 방법들을 리서칭해서 팀원들에게 소개하고, 프로젝트에 적용하며 확인해보는 것.

학습 목표를 달성하기 위해 시도했던 것들(모델 개선법들)

데이터 불균형을 해결하기 위해 데이터 증강과 샘플링을 적용했었다. 증강 방법으로는 2가지를 적용했었는데, 첫번째로 문장의 앞뒤 순서를 바꿔서 데이터를 추가하는 방법을 적용했었다. 해당 방법은 내가 진행했을 때는 오히려 모델의 성능을 저하시켰는데, 다른 팀의 발표에서 해당 방법을 적용했는데 성능이 나쁘지 않았다고 했어서 의아했다.

두번째로, 라벨이 0.0인 데이터가 지나치게 많았기 때문에 해당 데이터들의 일부를 떼어내서 각각의 문장 1과 문장 2를 **paraphrasing task**(비슷한 문장 생성)에 적용해서 문장 쌍을 늘리고, 새로 생긴 문장 쌍들의 유사도 값을 구해서 추가하는 방식의 증강법을 적용했다. 그러나 실험 결과, 원본 데이터만 넣었을 때보다 결과가 좋지 않았다. 실패한 이유는 라벨링 방식이 원본 데이터의 라벨링 방식과 달라서 라벨링이 잘못된 데이터들이 추가되었기 때문이라고 추측했다.

데이터 샘플링의 경우에는 오버샘플링을 진행했다. 데이터가 회귀 문제에 적합한 데이터이다 보니 특정 라벨에 속하는 데이터는 아예 존재하지 않는 경우도 빈번했어서 0~1, 1~2, 2~3, 3~4, 4~5으로 구간을 정하고, 구간에 속하는 데이터의 양을 고려하여 가중치를 설정하는 방식을 택했다. 실험 결과, 샘플링을 진행하지 않았을 때보다 성능 향상이 있었다.

마주한 한계&아쉬웠던 점

가장 아쉬웠던 부분은 제대로 검증하지 않고 성급하게 버려진 많은 가설들이다. 처음에는 여러 방법들을 찾아와서 팀원들과 항상 공유하고, 실제로 프로젝트에 적용하려고 노력도 했었는데, 해당 방법들을 한두번만 해보고 결과가 나오지 않자 바로 실망하고 버리다보니 새로운 방법을 찾아도 팀원들과 공유하고, 적용할 의욕이 이전보다 많이 떨어졌다. 이는 곧 근거있는 가설과 충분한 검증으로 구성되는 실험이 아닌, 이 방법 저 방법 내키는대로 가져다 붙이는 무논리적인 실험으로 이어졌다.

구현 부분에서 개인적으로 많이 부족했던 것 같다. 내가 맡았던 부분 중에 결과적으로 내가 구현을 못하고, 정말 감사하게도 다른 팀원분들께서 구현을 해주셨던 부분들이 있었는데, 이러한 부분들이 팀에 도움이 되지 못한 것 같아 아쉬웠다.

한계&교훈을 바탕으로 다음 프로젝트에서 시도해볼 것들

첫번째로, 내가 생각했을 때는 과연 정말 효과가 있을지 의문스러운 방법이라도, 일단 팀원들과 공유하며 더 좋은 방법을 찾기 위해 의논해봐야겠다는 생각이 들었다. 두번째로 가설을 세우고, 이를 체계적으로 다양한 환경에서 확실히 검증한 후에 해당 가설을 적용할지 말지 명확히 정하고 다음 단계로 넘어가야겠다는 생각이 들었다.

5-2. 서보성

- 내 학습 목표와 달성을 위한 방법

모델의 좋은 성능을 내는 것도 중요하지만 프로젝트 이전까지 배웠던 것들을 생각하면서 활용해보는 것을 목표로 잡았습니다. 모델의 성능을 올리기 위해서 다양한 방법이 있지만, 이번 프로젝트에서는 배운 것들을 활용하기 위해서 몇 가지 방법들에만 집중하여 진행했습니다. 특히 **Data Augmentation**과 모델의 학습 방법에 집중했습니다.

- 진행한 작업

- Data Augmentation

- EDA (Easy Data Augmentation)

- Token masking & Changing sequence order

- Learning Method

- R-drop

- K-fold

- 작업에 대한 성능 평가 및 한계

EDA를 진행했을 때, **Random Swap** 방법에 비해서 다른 방법들은 많은 노이즈를 발생시켜서 기존 문장의 의미를 변질시킬 수 있다는 것을 느꼈다. 그래서 해당 데이터로 학습했을 때, 모델의 성능 향상이 발생하지 않거나 오히려 떨어지는 경우도 발생했다. 반면에 **Token masking**과 **Changing sequence order**의 경우에는 성능의 상승이 있었다. 또한 **R-drop**의 경우에는 해당 **Dacon**에서 이진 분류에서 사용했던 방법으로 이번 프로젝트가 회귀 문제였기 때문에 생각했던 것보다 작은 성능 향상만 있었다. **K-fold**의 경우는 해당 프로젝트의 데이터가 적지도 않았고, 구현에 약간의 문제가 있어서 정확하게 구현을 하지는 못했기 때문에 많은 **R-drop**과 마찬가지로 약간의 성능 향상만 있었다.

- 이번 프로젝트를 진행할 때 잘된 점과 아쉬운 점

단순히 사전학습된 모델을 불러와서 학습을 진행한 것이 아니라 다른 방법들을 통해서 모델의 성능을 높이려고 시도해봤다는 점과 데이터 증강 등을 시도해본 것이 이번 프로젝트에서 잘된 점이라고 할 수 있다. 다만 정확히 구현을 못했던 부분과 왜 이것을 사용해야 하는지에 대한 정확한 이유를 가지고 있지 않고 구현을 하는 점은 아쉬웠던 부분이다.

- 이번 프로젝트를 통해서 배운 점

개념적으로 배우는 것과 실제로 프로젝트를 진행하는 것이 다르다는 것을 느끼게 되었으며, 이번 첫 프로젝트를 통해서 배운 것들을 토대로 다음 프로젝트에서는 구현하고자 하는 것을 제대로 구현하고 어떤 방법을 사용했을 때 해당 방법을 사용하는 이유에 대해서 확실하게 인지하고 사용할 수 있도록 프로젝트를 진행해야겠다는 것을 배웠다.

- 다음 프로젝트에서 시도해보고 싶은 점

이번 프로젝트에서 데이터를 증강하는 것에 많은 초점을 맞추어 진행을 해서, 모델의 하이퍼 파라미터를 튜닝하는 작업을 많이 진행하지 못했다. 이러한 부분을 다음 프로젝트에서 진행해보면서 발전시키고 싶다.

5-3. 오웬택

학습목표

처음으로 프로젝트에 참여하기 때문에 지금까지 학습한 이론을 코드로 구현하고 논리적으로 접근하는 것을 목표로 했다. 특히 모델 구조에 대해 학습한 부분이 많기 때문에, 데이터보다는 모델을 리서치하고 실험하는 역할을 맡아 더 깊게 공부하고자 했다. 처음부터 한계를 두기 보다는 여러 방법에 대한 가능성을 열어두고 확인해보는 시간으로 보내고 싶었다.

또 팀원들과의 협업을 통해 긍정적인 시너지가 나올 수 있는 구조를 경험하고 싶었다. 미리 정해진 룰을 지키며 함께 성장하는 팀 프로젝트를 진행하려고 했다. 적극적으로 커뮤니케이션에 가지며 서로의 부족한 부분을 채워주고 싶었다.

프로젝트에서 수행한 것

데이터를 살펴봤을 때 깔끔하지 않은 댓글 데이터가 많았고, 정규표현식과 **hanspell** 라이브러리를 가지고 데이터 정제를 했다. 그리고 비슷하게 한국어 댓글 데이터로 훈련한 모델을 리서치하고, 캐글이나 데이터콘을 살펴봄에 대회에서 활용할 만한 아이디어가 있는지 아이디어를 얻었다. 모델을 선별할 때 모델의 토큰라이저가 댓글데이터를 잘 인식하는지를 중점으로 탐색했다. 한정된 메모리와 시간때문에 토큰 길이 제한이나 파라미터 동결을 활용했다. 또 실험 과정에서 과적합이 일어난다고 생각해서, **dropout** 확률을 변경하거나, 모델 뒤에 **FC layer**를 추가하려고 여러 방법을 시도 했다. 그외 **Sentence Bert**를 구현해 **baseline**에서 벗어난 새로운 형태의 예측 모델로 성능을 비교해보고자 했다.

깨달음

팀적으로 다양한 시도를 하려고 여러 가설을 세웠지만, 막상 그 가설을 검증하는 과정에 대해서는 생각해보지 못했고, 명확한 답을 찾기 어려웠다. 프로젝트의 평가 지표인 **pearson** 상관계수를 위주로 성능을 평가하려 했지만, 다시 기회가 생긴다면 워드 임베딩이 어떤 방식으로 되었는지, 모델이 다른 **task**도 잘 수행하는지와 같은 다양한 방법으로 접근해보고 싶다.

배치사이즈, 토큰길이, **learning rate** 등등 하이퍼파라미터를 조절하는 과정에서 스스로가 정리 되지 않는 느낌이 들었으며 미리 결과를 예상하기도 힘든 부분이라 느꼈다. 다음 프로젝트에서는 이러한 실험을 할때만큼은 더 체계적인 계획을 세우고 싶었고, 모든 실험에 대해서 깔끔하게 기록을 통해 비교할 필요를 느꼈다.

코드를 구현하거나 깃허브를 다룰 때 어려운 점이 많아 팀원들의 도움을 많이 받았는데, 혼자 하는 것보다 훨씬 빠르고 효율적이라고 느꼈다. 다만 내가 설명할 기회가 있을 때 명확하게 설명하지 못하거나, 반대로 팀원의 설명을 잘 이해하지 못하는 경우가 있었는데, 팀에서 수행하고자 하는 기술의 개념에 대해 더 꼼꼼히 공부해야겠다고 다짐했다.

아쉬운점

Lightning, **Wandb**와 같이 익숙하지 않은 툴을 많이 사용하다보니, 코드 확장성에 한계가 있었다. 마찬가지로 계획한 모듈을 구현하는데 시간이 오래 걸려서 아쉬운 점이 있다. 또 파라미터 기록을 엑셀에 수기로 기록하려고 하다보니 어려움이 많았는데, 다음에는 이러한 부분도 미리 코드화 시켜놓고 싶었다. 협업 시스템이 잘 갖춰지지 않아, 자연스럽게 각자 역할을 수행하고 보고하는 느낌으로 진행 되는 면이 있었다. 온라인으로만 소통하다보니 양방향으로 피드백이 오고가는게 더 어렵기도 했다. 다양한 시도를 하는 것을 목표로 하다보니, 서로의 코드를 이해하는데 갖는 시간이 적어 더 힘들었다. 프로젝트 중간에도 부족한 점을 많이 느꼈지만 어떤 방식으로 개선 해야할지를 깨닫거나 토론하기는 힘든 상황이었다.

다음 프로젝트

이번 프로젝트에서는 양적으로 다양한 시도에 집중하고 결과를 얻는데 집중했다. 다만 그 과정에서 쉽게 포기하거나 가볍게 지나가는 것들이 많았는데, 다음에는 더 깊은 단계의 데이터, 모델 분석에 집중하며 프로젝트에 임하고 싶다. 추가적으로 구현한 코드의 경우 모듈화 시키지 않아 스스로 헷갈린 적이 많았는데, 다음에는 코드를 구현할 때 구조적으로도 더 깔끔하게 정리하고 싶다.

그리고 이번에는 각자 기술적인 부분에서 노력을 많이 했는데, 다음에는 팀원들과의 커뮤니케이션에 더 많은 시간을 투자해서 함께 고민하고 싶다. 온라인으로 하는 팀프로젝트의 한계점을 느낀적이 많았는데, 어떤 점을 개선할 수 있는지 생각해보려 한다.

5-4. 이승우

주로 이론 위주의 학습만 해오다 처음 도전해보는 대회였기에 최종 결과 보다는 대회의 전반적인 flow 파악 및 쌓아온 이론적 베이스를 직접 구현해보는 경험 자체에 큰 목표를 두었다. 따라서 EDA 및 baseline code 분석을 철저히한 후 문제 해결을 위해 어떤 방법을 적용해야 할지 배운 이론과 대입해보며 논리적으로 접근해보고, 가설을 세운 후 이를 코드로 직접 옮겨보고 그것이 잘 적용이 되는지, 효과가 있는지 확인하는 작업에 집중했다.

- 진행한 Task

- Baseline code에 attention_mask 및 token_type_ids 추가 적용
- Pre-trained model 뒤에 추가 layer 적용
- 특수문자 제거
- Label smoothing
- Regularization 연구
- Code refactoring 및 Module화

- 평가 및 한계, 느낀 점

- Baseline code를 분석하니 attention_mask 정보를 별다른 언급 없이 사용하지 않고 있다는 것을 발견해 이를 추가하고자 하였고, 실제로도 큰 성능 향상을 이루었으나, 처음 진행하는 프로젝트이다 보니 이 생각을 코드로 옮기는 데 시행착오가 많아 빠르게 해결하지 못했다.
- 특수문자를 학습에 활용한 pre-trained 모델이 많았기에 이를 제거하는 것이 큰 도움이 되지 않을 것이라 생각했고 이 가설을 증명하기 위해 특수문자를 제거하는 기능을 추가했다. 결과적으로 제거 후 대부분의 모델 및 상황에서 성능이 올라, 가설은 틀렸지만 성능 향상에 기여하는 경험을 했다. 하지 않아도 되는 작업이라 생각되더라도 실험을 통해 이를 증명해야 함을 느꼈다.
- 프로젝트 초반 조금의 소통의 부재 탓에 특수문자 제거 작업을 다른 팀원과 중복으로 진행하는 실수를 했다. 소통 및 공유가 특히나 대회 같은 단기적인 프로젝트에서 아주 중요함을 느꼈다.
- Label smoothing의 경우, 이론적으로 성능 향상이 보장될 것이라 생각해 작업을 자처해 많은 시간을 투자했다. 결과적으로 validation loss 값이 많이 높아지는 문제가 있었고 성능도 큰 변동이 없어 마무리 작업에서는 포함하지 않았다. 이론적으로 문제가 없다고 해도 실제 프로젝트에서는 마냥 좋지만은 않다는 것을 느꼈다. 하지만 한 번 구현해 본 경험이 앞으로 큰 도움이 될 것이라 생각한다.
- Validation loss가 Train loss와 비교해 너무 높다고 생각했고, 그 간극을 줄이기 위해 여러 가지 regularization 기법을 적용해 보았으나 극단적인 상황에서도 큰 효과가 없었고, 전처리 및 구조를 변경하지 않는 한 큰 효과가 없을 것이라고 결론지었다. 수치 정보에 대한 경험 및 상황 판단 능력이 부족했던 것 같다.
- 결론적으로, 문제에 논리적으로 접근해 해결하고자 했던 방식은 스스로 잘 지키도록 노력해서 만족하지만, 아직 소통 방식 등 프로젝트 경험 자체 및 문제 해결을 위한 적절한 대처 능력, 구현 능력이 많이 부족함을 느꼈다. 더욱 많은 경험을 쌓아야 함을 배웠으며, 따라서 다음 프로젝트에는 이번에 많이 경험해보지 못한 모델 리서치 및 fine-tuning 파트를 중점으로 맡아 진행해볼 생각이다.

5-5. 정효정

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

처음 대회에 참가해 보는 것이었기 때문에 대회 진행 방식을 익히는 것과 팀 내에서 맡은 일을 잘 해내는 것이 목표였다. 아직 나 자신이 부족하다고 느껴 경험이 있었던 EDA를 맡았고, 팀원들이 어떻게 실험하는지 참고해 실험 방식을 이해하면서 대회를 진행했다. 전처리 중에서 특수문자 및 불용어 제거, 역번역을 이용한 데이터 증강 부분을 맡았다. 라이브러리 및 모델을 이용하여 전처리를 진행했고, 이후 실험에서는 하나의 모델을 선택하여 전처리된 데이터를 이용하여 학습을 진행했다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

시간 분배에 어려움을 느꼈다. 코드를 merge한 후, 다른 사람 코드를 공부해 보고 싶었으나 시간에 쫓겨 못하고 넘어가 아쉬웠다. 시간 부족으로 인해 다양한 파라미터에서 실험해 보지 못한 점도 아쉬웠다. 전처리를 조금 더 빨리해서 부족한 점을 더 많이 파악하고, 더 다양한 실험을 해보고 멘토링 때 피드백을 받았으면 좋았을 것 같다. 결론적으로 코딩 구현 능력이 부족했던 것 같다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 시도해 보고 싶은 점은 무엇인가?

먼저 베이스라인 코드 분석과 EDA를 빠르게 마무리하고, 실험하면서 필요한 EDA를 추가로 진행하, 팀원들 간에 역할 분배와 할 일을 확실하게 정해 효율적으로 실험해 보면 좋을 것 같다. 그리고 다소 부족한 시간일지라도 다른 사람의 코드를 이해하고 다음 단계를 진행해야겠다.

○ 나는 어떤 방식으로 모델을 개선했는가?

처음에는 preprocessing 함수에서 역번역 함수를 호출하여, 학습 전에 데이터를 역번역을 했다. Google trans 라이브러리, 모델 'Helsinki-NLP'와 'facebook/mbart-large-50-many-to-many-mmt'를 이용하였으나 배치화가 안된다거나 번역 성능이 좋지 않아 'Dooly' 라이브러리를 이용하여 배치화하여 역번역했다. 미리 CSV 파일로 만들어 두고 학습을 진행하면 더 효율적이기 때문에 preprocessing 함수에서 호출하지 않는 방식을 택했다. 이 밖에 특수문자를 지우는 함수를 만들고, 여러 파라미터에 모델을 학습시켜 좋은 성능을 나타내는 파라미터를 찾고자 하였다.

○ 내가 해본 시도 중 어떠한 실패를 경험했는가? 실패의 과정에서 어떠한 교훈을 얻었는가?

'은','는','이','가','을','를'과 같은 불용어를 제거하려고 했으나 잘 안되었는데, 시간이 부족해 보완하지 못하고 마무리했다. 코딩 연습을 많이하고, 다른 사람들의 전처리를 많이 찾아보고 참고해야겠다.

○ 협업 과정에서 잘된 점/ 아쉬웠던 점은 어떤 점이 있는가?

대회 시작 전에 함께 EDA를 진행하고 Baseline Code를 분석한 후, 전처리와 모델 연구로 역할 분배 한 점을 잘한 것 같다. 또한 멘토링 후, 보완할 점들을 우선순위를 정해 역할을 나누고, 깃허브를 통해 코드 공유를 원활히 했던 점과 한 명이 merge를 담당해 충돌과 코드 오류를 최소화했던 점이 좋았다고 생각한다. 그러나 서로의 코드를 보고 리뷰해 주기로 했는데 못 해서 아쉬웠고, 각자 실험에서 사용한 하이퍼파라미터, 훈련 데이터, 결과를 깔끔하게 정리해서 공유했으면 좋았을 것 같다. 또한 같은 모델에 대해서 다른 하이퍼파라미터를 사용했다면 이를 WandB로 공유하여 비교하면 좋았을 것 같다. 실험을 체계적으로 진행하지 못해 아쉬웠다.