Semantic Text Similarity(STS) Report

1. 프로젝트 개요

A. 프로젝트 주제

컴퓨터로 언어를 이해하는 과정에서 문장에 대한 유사성 판단은 중요한 부분 중 하나이다. 이번 프로젝트는 한국어 문장에 대한 Semantic Text Similarity(이하 STS) 분석으로 한국어 문장의 유사도를 0과 5 사이의 값으로 예측해야 한다.

B. 데이터 셋

총 10974 문장 쌍이 데이터로 주어졌으며, 이 중 train data는 9324 쌍, test data는 1100 쌍, dev data는 550 쌍으로 구성되어 있다. Train data에는 2가지 label이 존재하며, 한 가지는 0에서 5 사이의 실수이고 나머지는 2.5점 이하는 0, 2.5 이상는 1인 binary label이다.

C. 협업 과정

원활한 협업을 진행하기 위해 필요시에는 ZOOM을 사용해 회의를 진행하고 코드 수정이 요구되는 경우 ZOOM 원격 조종을 활용했다. 간단한 논의 같은 것은 SLACK으로 진행했고, Notion에서 프로젝트를 전반적으로 관리했으며, 개발자다운 협업을 경험해보기 위해 Github를 필수적으로 사용했다.

2. 프로젝트 팀 구성 및 역할

이름	역할
김현수(팀원)	Data EDA, pytorch 코드 pytorch lightning으로 수정, 학습 방법 수정, earlystopping, unk token 확인, wandb sweep 기능 추가
김혜빈(팀원)	다양한 모델 실험, 데이터 증강 (oversampling), Notion 문서 정리
박승현(팀장)	baseline 코드 작성, k fold 코드 작성, 도메인별 학습된 모델 종합하는 코드 작성, 다양한 모델 실험
최동민(팀원)	KLUE 논문 리뷰, pytorch 코드 pytorch lightning으로 수정, prediction check, 학습 방법 수정
홍인희(팀원)	다양한 모델 실험, 데이터 EDA 및 전처리, tokenizer check, unk token vocab 코드 추가

3. 프로젝트 수행 절차 및 방법

A. 코드 작성

협업하는 과정에서 사람마다 다른 코드를 사용한다면 각자 다른 방식으로 코드를 수정하기 때문에 이론은 비슷하더라도 성능은 달라질 수 있는 문제가 있다. 모든 팀원이 같은 코드로 실험할 수 있도록 우리 조만의 baseline code를 작성하기로 했다. baseline code는 한 사람이 작성했으며, 모든 팀원이 쉽게 이해할 수 있도록 최대한 모듈화했고 코드 작성 완료한 후 팀원에게 코드 흐름에 대한 설명도 진행했다.

B. KLUE 논문

프로젝트를 진행하기에 앞서 STS task가 어떤 문제를 해결하길 원하고 과거 STS task를 해결할 때 어떤 접근법을 사용했는지 파악하기 위해 관련 논문인 KLUE 논문을 읽어보기로 했다. KLUE 논문에서는 STS 데이터를 수집하고 레이블링하는 방식에 대해 서술했다. 실제로 대회 STS와 KLUE STS의 labeling 방식이 유사하다는 점을 알 수 있었고, KLUE 논문에서는 어떤 방식으로 STS 문제를 해결했는지 파악할 수 있었다.

C. 모델 구조

모델 구조가 다르다면 파라미터 개수와 구조가 상이하므로 다른 결과가 도출될 것이라고 생각하여 다양한 모델 구조에 대해 실험을 진행해봤다.

a. Only Regression

MSE loss를 활용

b. Regression + Binary Classification

데이터에서 주어진 binary label을 활용하여 loss 함수를 regression loss(MSE) 뿐만 아니라 binary classification loss(BCE)도 추가하여 사용해보았다. 비율 역시 beta라는 하이퍼파라미터로서 sweep 등을 이용해 튜닝을 해보았는데 약 7:3(MSE: BCE) 정도 일때 성능이 가장 좋았다.

c. Regression + Binary Classification

대회의 학습 데이터 개요 및 KLUE 논문을 통해 사람이 데이터를 labeling 할 때 3.8, 4.2 등의 score로 바로 나누는것이 아니라 위와 같은 기준으로 0~5 사이의 정수로 labeling하고 여러 사람들의 label들을 평균내서 score를 계산했다는 것을 알 수 있었다.

따라서 모델 역시 바로 regression을 하는 방향 보다는 multi label(0,1,..,5)을 새로 추가하여 사람이 labeling한

방법과 비슷한 방법으로 적용해보려 하였다. binary classification loss(BCE) 대신 multi classification loss(CE)으로 regression loss와 섞어보았다. 아쉽게도 성능은 그닥 좋아지지 않았는데, 마스터님께서 말씀해주신 것 처럼 Binary classification과 multi classification에 사용되었던 각 label들의 정보량이 동일하기 때문인 것 같다.

d. Only Classification

score label의 개수가 31개였기 때문에, 이를 31개의 label에 대한 classification(only) 문제로 정의하여 접근해봤는데, 성능이 좋지않았다. 그 이유는 31개의 각 label들에 해당하는 데이터 개수가 많이 부족하였기 때문인 것 같다.

e. Add More Layers

layers를 깊게 쌓으면 성능이 좋아지지 않을까 하는 생각에 dropout을 적절히 섞어가며 layer를 많이 쌓아봤는데, 눈에 띄는 성능 향상은 없었다.

D. EDA

a. 결측치/데이터 중복 확인

결측치와 중복된 데이터는 없었다.

b. 최대 시퀀스 길이 확인

모델의 input token의 수는 제한되어 있기 때문에, 문장의 최대 시퀀스 길이를 확인하여 잘려서 들어가는 문장이 없도록 max sequence length를 256으로 설정했다.

c. binary label & label 별 데이터 분포 확인

0을 가리키는 label이 다른 label에 비해 비정상적으로 많은 양을 가지고 있다. 이를 통해 모델에 균일하게 학습시키는 방법을 찾아보았다.(oversampling, undersampling)

d. source(rtt, sampled) 분포 확인

rtt의 분포는 주로 3, 4, 5 label을 가리키고 있고, sampled의 분포는 주로 0, 1, 2 label을 가리키고 있다.

E. 데이터 전처리

a. 띄어쓰기

데이터를 살펴보면 띄어쓰기 되지 않은 문장들이 많이 있다. 이러한 데이터들을 띄어쓰기 해줌으로써 UNK token의 발생 비율을 줄일 수 있다. 띄어쓰기 API로는 딥러닝 모델 패키지인PyKoSpacing을 사용하였다.

b. 특수기호 삭제

문장이 유사함에도 불구하고 특수기호로 인해 유사하지 않다고 판단하는 경우가 있을 것이라고 판단하여 정규표현식을 활용하여 특수기호를 삭제하여 데이터를 추출하였다.

c. 맞춤법 교정

제대로 되지 않은 맞춤법으로 인해 문장의 의미가 다르다고 판단되는 경우가 많았다. 맞춤법 교정 API(py-hanspell)를 통해 맞춤법 교정을 시도했으나, 제대로 된 문장까지 다른 의미로 변질되는 경우 때문에 성능이 **오히려 낮게 나왔다.**

F. Tokenizer

a. UNK token을 tokenizer vocab에 추가

문장이 통째로, 혹은 중요한 의미를 지닌 단어가 통째로 [UNK] token으로 발생되는 경우가 있었다. 문장의 의미를 잘 나타낼 수 있게끔 [UNK] token 발생 비율을 줄여야 좋은 성능을 낼 수 있다고 생각하였다. 각각의 tokenizer마다 [UNK]로 나타나는 문장이 없게끔 train data에서 나오는 unk token을 vocab에 추가하여 모델이 학습시킬 때 문장의 의미를 보다 더 쉽게 알 수 있을 것이다.

G. Oversampling & Undersampling

a. Oversampling

데이터는 많으면 많을수록 유리하다. oversampling에는 AEDA, back-translation 등 다양한 방법이 존재하지만, back-translation만 진행하기로 결정했다. AEDA 등 문장에 특수 문자를 추가하거나 문장 내 단어 순서를 변경하는 방식으로 데이터를 증강하는 경우 문장의 의미를 변경시킬 수 있다. 실제로 실험한 결과, AEDA를 진행한 경우 많은 데이터의 의미가 변질되는 것을 확인할 수 있었다. STS의 경우 문장의 의미적 유사도를 예측하는 것으로 의미가 바뀐다면 이에 상응하는 label도 변동되야 한다. 따라서, AEDA와 같은 문장의 의미를 퇴색시키는 증강 기법은 사용하지 않았다. 최종적으로 back-translation만 남았는데 파파고로 back-translation(ko-en, en-ko)을 진행했다. back-translation을 진행한 이후의 데이터와 기존 데이터를 대조해본 결과 반 이상 데이터의 의미가 살짝 변경된 것을 발견할 수 있었다. 데이터의 의미가 변경됐음에도 불구하고 실험을 진행해봤는데, 제대로 예측하지 못한 데이터의

대부분이 back-translation 데이터여서 back-translation을 적용하지 않기로 했다.

전체 문장을 번역시에 질이 좋지 않게 번역되는 문장들이 섞여있어 성능 증가로 이어지진 않았다. 처음에는 Mecab을 이용한 back translation을 진행했지만, 서버 내에서 설치 문제가 존재해 google translation으로 진행했다. 일본어와 프랑스어를 이용한 재번역이 그나마 괜찮은 번역 결과를 보였지만 콩글리쉬처럼 재번역하기 까다로운 문장은 완전히 다른 번역이 되는 경우가 있었다.

b. Undersampling

Oversampling으로 데이터의 분포를 맞추기 어렵다고 판단해 데이터가 과하게 많다고 생각되는 부분을 줄여주기로 했다. 데이터는 문장으로 이뤄져 있기에 분포 기반으로 데이터를 sampling하기 어려웠다. 따라서, 가장 쉬운 방법인 random undersampling을 실시했다. 그 결과, 전체 데이터를 학습시키는 방식보다 성능이 향상됨을 알 수 있었다.

H. Learning by Domain

상대적으로 말투(구어체/문어체)가 비슷하다고 생각되는 도메인끼리 나누어 학습시켰다. petition/slack,nsmc 로 나누어 각각을 다르게 학습시켰다. 이 때, undersampling과 전처리된 데이터로 학습시켰다. KcELECTRA는 정제되지 않은 구어체 특징에 신조어가 많으며, 오탈자 등이 많이 나타나는 데이터셋에 상대적으로 강점을 갖고 있는 Pretrained ELECTRA 모델이다. 그렇기에 slack과 nsmc가 구어체에 가깝다고 판단하여 KcELECTRA를 적용해보았고, 문어체에 가까운 petition 데이터는 KoELECTRA을 적용했다. 아쉽게도 성능은 더 떨어지는 모습을 보여 실험에서 그쳤다.

I. Prediction Check

Fine tuning한 모델로 Validation data에 대한 예측치를 뽑아보고, 정답과 비교해보았다. label의 차이가 1.5이상 나는 것들은 차이가 크게 나는 이상치들로 봤고, 그 과정에서 오타가 있는 문장을 사람과 달리 잘 예측하지 못하는 점, 어투의 차이와 유의어 사용으로 인해 점수가 크게 바뀌는 점 등, 모델이 제대로 예측하지 못하는 것들의 경향성을 찾을 수 있었다.

예측을 제대로 하지 못한 데이터들에 대해서는 추가적인 학습을 진행하는 등의 방법을 적용해볼 수 있을 것이라 생각했지만, 그 기준이 모호하고 데이터의 양이 적어서 시도하지 않았다.

J. Hyperparameter Tuning

WandB의 sweep을 사용해 beta (regression : classification 비율), learning rate, eps, weight decay의 최적의 hyperparameter 조합을 찾았다.

beta 값의 경우 loss 함수를 결정짓는 중요한 요소이고, learning rate와 batch size 등 역시 최적의 하이퍼파라미터를 찾아 고정시켜놓고 실험하는 것이 실험하는데 있어서 효율적이라고 생각하였다.

다만 시간이 오래 걸려서 야간에는 15번의 test를, 주간에는 5번의 test를 반복적으로 해가며 각각의 하이퍼파라미터들이 어떤 경향일 때 최종 성능이 좋은지 비교해봤다.

K. K Fold

하나의 모델에 대해 train data를 5등분 하여 그 중 4개를 학습시키고 나머지 하나를 validation data로 활용하였고, 그것을 서로 다르게 총 5번 시행하였다. 이후 5번의 시행을 각각 거친 모델들을 종합하였다.

L. 앙상블

a. 성능이 좋았던 상위 7개 모델 앙상블

성능이 좋았던 모델들끼리의 앙상블은 시너지 효과를 낼 수 있을 것이라고 생각했다.

b. KoELECTRA를 사용한 모델끼리 앙상블

KoELECTRA를 사용한 단일 모델들이 안정적으로 성능이 높았기에 KoELECTRA로만 이루어진 앙상블을 실험해봤다. 결과는 미미했다. 같은 모델 하에 하이퍼 파라미터, 데이터만 조금씩 달랐기에 영향력이 크지않아 비슷한 결과가 나온 듯하다.

c. 다양한 모델 앙상블

모델마다 잘 예측하는 부분이 다르다고 생각하였고, 이것들을 종합한다면 일반화 측면에서 성능 향상을 기대할 수 있고, 실제로 성능 향상을 불러왔다.

d. klue 논문에서 데이터의 label을 부착할 때 사용했던 방법과 마찬가지로, 7개의 후보군을 뽑은 후 평균을 내어 평균값에서 가장 먼 2개를 제외한 나머지 5개의 결과를 평균내서 결정

평가 지표인 피어슨 상관계수가 이상치에 민감한 만큼 평균에서 제일 먼 2개의 후보를 제외함으로써 이상치를 없

애는 효과를 기대할 수 있고, 실제로 소폭의 성능 향상을 불러왔다.

4. 프로젝트 총 회고

베이스라인 모델(koelectra)을 기준으로 다양한 실험들을 해봤는데, 대조군이 없어 명확한 성능 향상의 이유를 찾지 못한 모델들도 꽤나 있었다. 깃허브와 노션을 조금 더 잘 활용했더라면 모델 구조, 데이터 전처리, Undersampling, Tokenizer unk vocab 추가, Hyperparameter Tuning, K Fold, 앙상블 등의 각각의 항목에 대해서 어느 정도의 성능 향상을 보였는지 정리 할 수 있었을 것 같다. 또한 한국어로 된 PLM이 한정되어 있고, 모델의 구조를 바꾸는 것 보단 좋은 데이터를 입력값으로 넣는 것이 훨씬 더 중요하다는 것을 깨달았다. 멘토님께서 말씀해주신 "Garbage in, Garbage out"이란 말의 의미를 실감할 수 있었고, 다음에는 데이터 전처리에 더욱 힘을 써보고 싶다.

5. 자체 평가 의견

a. 잘한 점들

1. 팀원들이 모두 빠르게 소통하며 서로의 의견을 존중해주는 진정한 협업을 진행했다.
2. 등수에 연연하지 않고 모두의 성장을 위해 팀원 모두가 다양한 시도를 해보려고 노력했다.
3. 실험하는 도중 잘 되지 않을 때, 팀원들이 다같이 해결방안을 제시해주거나 코드 수정을 다같이 하여 문제해결 속도가 빨랐다.

b. 시도 했으나 잘 되지 않았던 것들

Multi-classification, Over-sampling (back-translation), 맞춤법 교정(private에서 강점이 있던듯하다), 특수기호 제거, Learning by domain

c. 아쉬웠던 점들

1. 모델 실험 시, 기준을 정해놓고 그 동일한 조건 기준 하에서 실험을 진행해가며 분석을 했어야 했는데 그러지 못한 게 아쉽다.
2. 더 다양한 모델들을 찾아보지 못한 것이 아쉽다.
3. 깃허브를 활용한 협업이 아직 익숙치 않아 버전 관리, 코드 공유 등에서 어려움이 조금 있었다.

d. 프로젝트를 통해 배운 점 또는 시사점

1. 이번 프로젝트를 통해 앙상블의 한계를 느꼈고 정말 다양한 실험을 해봐야겠다고 생각했다.
2. 체계적인 실험과 효율 측면에서 잘 짜여진 프로젝트 템플릿의 중요성을 느꼈다

김현수B_T4060 Level 1 프로젝트 개인 회고

1. AI 프로젝트에서 고려해야 할 사항

1-1. 협업에 관하여

AI 뿐만 아니라 개발 관련 프로젝트를 진행하는데 가장 중요한 것은 원활한 협업이라고 생각한다.

프로젝트는 혼자 하는 것이 아니기 때문에, 팀원들과의 협업은 프로젝트의 성공을 좌지우지 할 만큼 중요한 요소라고 볼 수 있다.

1-1-1. 프로젝트의 목적은 무엇인가

부스트캠프의 특성상 그리고 다행히도 그 안에서 너무나도 좋은 팀원들을 만난 덕분에, 소통이나 적극적인 참여 등에 있어서 팀원들과 갈등이 있었던 적은 없었다. 그러나 프로젝트가 진행되면서 약간의 의견 차이를 느낄 수 있었는데, 아래와 같은 문제였다.

Q. 좋은 점수(등수)가 목적인가? 아니면 각자의 실력 향상이 목적인가?

아마 대부분의 사람이 그럴 것 같은데, 머리로는 실력 향상을 목적이라 하지만 좋은 점수 앞에서 욕심이 생기는 것도 인간의 어쩔 수 없는 본능인 것 같다. 실력 향상이라는 것에 대한 기준이 사람마다 다를 뿐더러 제출 횟수가 정해져있는 AI 대회 특성 상 후반으로 갈 수록 좋은 점수에 신경 쓸 수 밖에 없는 것 같다.

예를들어, 실력 향상에는 도움이 되지만 누가봐도 결과가 안 좋아보이는 모델을 제출하는 것이 의미가 있는가? 라는 질문의 경우 위 질문보다 조금 더 답변 내리기 어려울 것 같다.

내가 내린 결론은 여기는 부스트캠프이기 때문에 실력 향상을 주요 목적으로 해야한다는 것이다.

캐글과 같은 중요한 대회를 참여하거나, 실무에서 AI 엔지니어로 일하는 경우에는 상황이 조금 다를 수 있지만 부스트캠프에서 만큼은 실력 향상에 힘써야한다. 부캠의 목적을 생각해보니 꽤 쉽게 결론을 내릴 수 있었는데, 내가 부스트캠프에 참여한 이유는 어떤 프로젝트 혹은 대회에서 좋은 등수를 받기 위함이 아닌, AI 엔지니어로의 역량을 부스트하기 위함이다.

역량을 부스트한다. 라는 말은 지난 회고에서 변성윤 마스터님이 하셨던 "부스트캠프 AI Tech 4기가 끝나는 시기엔 어떤 모습이길 바라나요?" 질문에 대한 답변과 이어지는데, 부캠에 약 두달동안 참여하면서 느낀점은 부캠은 최고의 환경을 제공할 뿐 결국 얼마나 부스트하느냐는 자신의 마인드와 부캠에 임하는 자세에 따라 달라진다고 생각한다. 물론 모두의 시작점은 당연히 다르겠지만, 시작점 이외에 앞으로 나아갈 보폭, 나아가는 끈기, 나아가는 방향 등은 모두 자기가 정하는 것이다. 시작점이 다르다고 상황 탓을 하기 보단 앞으로 어디로, 어떻게 나아갈 것인지를 고민해봐야겠다.

다시 돌아와서, 결론은 **부캠의 환경이 좋은 점수보다는 실력 향상에 최적화 되어있다**는 것이다. 멘토님들, 마스터님들은 실력 향상을 위한 방법을 알려주시지 점수를 올리는 방법은 알려주시지 않는다. 따라서 앞으로도 나는 부캠을 하며 실력 향상을 최우선시 할 것이다. (가장 좋은것은 실력 향상과 동시에 좋은 점수까지 따라오는 것 ..)

LEVEL2, 3에서 하는 프로젝트들도 팀원들과 그런 부분에 있어서 생각을 맞추고 가면 좋을 것 같다.

1-1-2. 소통 및 버전 관리 (Git)

프로젝트를 도와주는 수많은 소통 및 버전 관리 툴들이 있는데, 대표적으로는 Slack, Notion, Git, WandB 심지어 카카오톡 등이 있는 것 같다. 가장 할 말이 많은 것은 Git인데, 아마 코드를 공유하고 버전을 관리해야하는 프로젝트인 경우 Git을 사용하는 것이 훨씬 편리할 것이다.

다만 나는 이번 프로젝트에서 Git이 편리하지 않았는데, 그것은 100퍼센트 나의 부족함 때문이다. 어떻게 보면 당연한 얘기지만 깃허브를 이용한 프로젝트를 이번에 처음 해봤기 때문에 아직 너무나도 낯설었고, 이고잉님의 깃허브 강의도 조만간 다시 볼 예정이다.

깃 말고 다른 툴들은 팀원들과 협의해서 적절하게 사용하면 될 것 같은데 WandB 같은 AI 프로젝트를 관리할 수 있는 툴의 경우 개인적인 공부가 조금 더 필요할 것 같다.

1-1-3. 역할 분담과 역할 공유

마지막은 역할 분담과 역할 공유라고 적었는데, 역할 분담보다는 역할 공유에 대해서 강조하고 싶었다.

프로젝트를 진행하면 자연스레 역할 분담을 하기 때문에 이 부분은

이번 프로젝트의 경우, 나는 pytorch 코드를 pytorch lightning으로 바꾸거나, earlystopping 코드 구현, 다양한 모델 실험, sweep을 활용한 하이퍼파라미터 튜닝 등을 맡아서 하였는데, data augmentation쪽은 아예 공유가 안돼서 어떤 식으로 진행되는지 꽤 오랬동안 알지 못했다. undersampling이나 oversampling 부분 역시 하지 않았기 때문에 잘 모르는 상태로 프로젝트를 진행했던 것 같다.

사실 역할 공유가 되지 않는 역할 분담은 **역할 분리**라고 생각한다.

역할 분리가 되지 않기 위해서는 팀원들끼리 나눈 역할에 대해서 내용을 공유하고, 서로의 지식 및 알고있는 내용을 공유해가며 기준 선을 맞추는 과정이 반드시 필요하다. 역할을 잘 나눈것은 좋았으나, 주기적인 회의 등을 통해 각자 역할에서 한 내용을 설명해주고 공유한다면 훨씬 더 의미있는 프로젝트가 되었을 것이다.

1-2. 모델링

대회를 시작하기 전 개인적인 목표는 다양한 모델을 써서 비교해보자는 것이였다.

물론 koelectra부터 roberta, kcelectra 등의 모델들을 써봤지만, 더 다양한 모델을 공부하고 비교해보는 공부하 필요할 것 같다. 사실 아직 ELECTRA에 대해서 잘 알지 못하고 그냥 썼던 것 같은데, 이번주에는 ELECTRA 공부를 하며 왜 koelectra가 klue/roberta 보다 전체적으로 좋은 성능이 나왔는지에 대해서도 고민을 해봐야겠다.

다른 팀 얘기를 들어보니 pre-training부터 구현하려한 팀도 있었고(실패했지만), RNN 이나 LSTM 등을 사용한 팀도 있었다. RNN이나 LSTM을 사용한 모델을 만들어보고, BERT 계열 모델들과 비교하면서 프로젝트를 진행했다면 조금 더 좋지 않았을까 하는 생각이 든다. 멘토님께서도 무작정 큰 모델을 사용할 것이 아니라 성능 차이가 거의 나지 않는다면 (실무 관점에서는) 작은 모델을 사용하는 것이 나쁘지 않을 수 있다고 말씀해주셨다.

다음 대회에서는 완전 기본 모델(RNN, LSTM)부터 훨씬 더 다양한 최신 모델(BART, T5, xlm roberta large 등)까지 공부해보고 비교 및 실험해보자.

1-3. Why? 계속 하기

성능이 왜 올라갔는지에 대한 고민을 계속 해야한다.

이를 위해서는 반드시 대조군이 필요한데, 대조군에 대한 정리 및 분석이 조금 부족했던 것 같다.

점수가 잘 나오면 왜 잘 나온지를 분석해서 더 나은 방향으로 발전시킬 방법에 대해 논의해야하는데, 대조군 없이 이것 저것 중구난방 식으로 실험을 했던 것 같다. baseline model(대조군)을 확실하게 잡고, 실험을 해가며 성능을 향상시키는 것이 프로젝트를 길게 보았을때, 또 훨씬 더 복잡하고 어려운 task가 주어졌을때 큰 효과가 있을 것이라 생각한다.

마스터님께서도 최종 1,2등 팀에게 질문하셨던 것이 모델의 성능이 왜 잘 나온것 같냐 라는 질문이였는데, attention 메커니즘 때문이라고 답변을 한 팀이 있었다. 그런데 그 팀은 attention 메커니즘을 적용하지 않은 모델은 사용하지 않았다. 즉 대조군이 없었다는 얘기인데, 명확한 분석을 위해서는 대조군이 반드시 필요하다는 것을 배울 수 있었다.

1-4. 기타 질문들 및 배웠던 것들

- V100이 아니라 작은 GPU 한장에 이 모델을 혹은 비슷한 성능을 내는 모델을 넣으려면 어떻게 해야하나?
    - 잘 학습하지 못한 데이터를 추가로 학습하는 boosting 방법 등 (조금 더 찾아볼 것)
- task가 주어졌을때, 어떤 분류기를 만들어서 1~1000번 데이터는 A 모델, 1001~2000번 데이터는 B 모델 등으로 데이터를 나눠서 학습도 가능
- 정보량이 같으면 추가적인 loss를 쓸 필요는 없다.
    - label1이 0~5까지의 score고 label2가 label1을 기준으로 다시 나타낸 binary label(0, 1)이면 두 label의 정보량은 같다.
- scatter plot을 활용해 최종 예측값의 분포를 확인해 볼 수 있다.
- 다양한 loss 함수에 대한 실험도 꼭 해보자
- Wandb 및 sweep을 잘 활용해보자
- lr scheduler도 적절하게 활용해보자 (어떤 것들이 있나 우선 찾아볼 것)

- 모델 저장할때 파이프라인을 잘 만들어놓자! (.pt파일을 관리하기 번거로움)

2. NLP 프로젝트에서 특히 더 고려해야 할 사항

2-1. Tokens

- 최대 토큰 길이를 살펴본 후 max_token을 256 → 128로 수정 (170과 130 제거)

2-2. Data augmentation

- BERT 계열 모델에 들어가는 input에서 sentence 1 [SEP] sentence 2 를 sentence 2 [SEP] sentence 1 형태로 바꿔서 augmentation 할 수 있다.
- 다양한 언어로의 backtranlation을 활용해 aumentation 할 수 있다.
  - 성능은 보장이 안 되므로 눈으로 번역 결과를 보자 → backtranslation은 결과가 오히려 안 좋았음
- MLM을 활용한 aumentation
- koeda
- source column
- 띄어쓰기 및 맞춤법 교정 : 삭제하는 것이 아닌 ..... → ... or ????????????? → ? 처럼 반복되는 특수문자는 한 두개로 치환

3. 앞으로는 어떻게 할 것인가

3-1. 이번 프로젝트의 교훈

pytorch를 활용해 이렇게 모듈화 하면서 체계적으로 nlp task를 수행해본 적은 처음이였는데, pytorch뿐만 아니라 pytorch lightning 코드의 흐름을 잘 이해할 수 있었다.

huggingface를 활용해 BERT 계열의 PLM(klue roberta, koelectra, kcelectra 등)도 사용해보며, 이론으로만 배웠던 내용들을 실제 코드로 적용시켜보니 이론과 실습 사이의 공백을 채울 수 있었다.

Git을 활용한 협업 역시 처음이였는데, Git이 정말 편리하지만 툴인 것은 맞지만 아직은 내가 많이 기능을 모른다는 생각이 들었고 추가적인 공부를 해야겠다는 생각이 들었다.

3-2. 앞으로는?

프로젝트를 시작하기 전에는 왜? 라는 질문을 많이 해야겠다고 다짐했었는데, 짧은 기간 동안 왜? 라는 질문을 계속 던지고 이에 대한 답변을 생각해가며 프로젝트를 진행하기에는 아직 내 실력이 부족했다. 예를들어, koelectra를 사용했을때 다른 모델보다 성능이 왜 잘 나왔는지 알고싶다면 우선 electra의 모델 구조 및 특징부터 이해하고 있어야 할 것이다. 대회 기간 동안 이러한 내용들을 공부하며 진행하기에는 뒤로 갈 수록 시간이 부족했고, 대회가 끝나고 다시 공부해볼 예정이다.

🔥 김혜빈 개인 회고

Tags

✍🏻 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

■ 우리 팀과 나의 학습목표는 무엇이었나?

🥕 우리 팀의 학습목표는 성능을 올리면서 다양한 실험을 해보는 것이 목표였다.

🥕 나의 학습목표는 대회의 흐름을 알게 되는 것과 실험하는 방법 알기, 코드 이해 등 AI 엔지니어가 되기 위한
기본적인 것들을 배우기가 목표였다. 첫 대회이자 이론이 아닌 실무에 관련된 일을 해보는 것이기에 욕심부리지 않았다.

■ 개인 학습 측면

나의 학습목표를 달성하기 위해 맡은 업무를 깔끔하게 처리하려고 노력했다.
데이터 증강 (Oversampling) 업무와 모델 실험을 진행했다.
데이터 증강에 대해 처음 들어봤고 시도해봤기 때문에 처음에는 어리바리했지만 자료를 많이 찾아보고 코드를 작성했다.
누구나 내 코드를 봤을 때 직관적으로 이해할 수 있게끔, 내가 코드를 작성하면서도 불필요한 작업을 최대한 하지 않게끔
함수화해가며 코드를 깔끔하게 짜려고 노력했다.
누군가 "데이터 증강 코드에 뭐 그리 노력을 쏟았나" 할 수 있겠지만 나는 쉬운 부분, 많이 중요하지 않은 부분이어도
깔끔하게 코드 짜는 연습을 하며 개발을 잘하는 개발자가 되고 싶었기 때문에 작은 부분에서도 열심히 했다.

또한, 이해가 가지 않는 부분에 대해서는 스스럼없이 팀원들에게 물어보았고 따로 개인적으로 설명을 듣기도 했다.
다른 팀원들도 본인 업무로 인해 바빴을텐데도 친절하게 설명해준 팀원들에게 큰 고마움을 느꼈다.

■ 공동 학습 측면

팀원들과 함께 소통을 끊임없이 했다. 혼자 결정하는 일은 최소화하고 팀원들의 의견을 묻고 진행하는 등,
서로를 배려하는 모습이 많이 보였다. 공동적으로 갖가지 모델 실험을 진행하면서 결과를 공유하고
그 외의 모든 일도 함께 진행했다. 다른 팀원이 이해를 못했거나 질문을 하면 친절하게 다시 설명을 해주면서
많이 가르쳐주고 배우면서 학습을 진행했다.

✍🏻 나는 어떤 방식으로 모델을 개선했는가?

■ 사용한 지식과 기술

아쉽게도 이론에서 배웠던 내용들이 막상 실무에 들어가니 잘 기억이 나지 않고, 생각이 나더라도 어떻게 사용하면 좋을지
이렇게 적용해도 되는건지 혼자서 판단이 서지 않아 모델 개선에 큰 도움이 되지 못했다.
그래도 하이퍼 파라미터를 조정해가면서 데이터 전처리에 힘쓰는 방향으로 생각을 많이 했던 것 같다.
모델도 중요하지만 결국 input인 데이터가 가장 중요하다고 생각하기 때문에 고품질의 데이터로 전처리 하려고
많은 고민을 해봤던 것 같다.

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

데이터 증강을 진행하면서 여러 증강 방법에 대해 찾아봤지만 (SMOTE, GAN…)

우리 데이터에 어떤 기법을 적용해서 증강하면 좋을지 판단이 서지 않았다.

모든 기법을 한번씩 다 사용해보고 싶었지만, 사용해보기에는 데이터 증강 업무에 주어진 시간이 많지 않았고

`코드를 직관적으로 작성하기 위해 코드에 노력을 기운 점 + 서버를 처음 이용해봐서 서버에 대한 어려움을 겪은 점` 으로 인해 가장 많이 사용한다는 back translation으로 데이터를 증강했다.

역시 모든 일은 처음하면 어렵기 마련이고 적응하면 괜찮아진다는 사실을 다시 한번 깨달았다.

데이터 증강에 예상보다 많은 시간을 쏟았는데, 다음 번에는 어떻게 하면 효율적일지 조금이나마 깨달았다.

아쉽게도 back translation 결과가 좋지 않아 데이터 후처리를 한 번 더 해줬어야 했는데

시간이 부족해서 진행하지 못했고 증강한 데이터는 쓰지 못하게 되었다.

그래도 데이터 증강이라는 것에 대해 배울 수 있었고 처음 시도한 뜻깊은 경험이 된 것 같아서 좋았다.

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

이번 대회가 첫 대회이자 AI 실무 첫 도전이었기에 비교군이 없지만,

새롭게 시도한 변화로써 `over sampling` 과 `model 실험` 이 존재한다.

또 다른 변화는 이전에는 AI 베이스라인 코드 흐름을 전혀 이해하지 못했는데 이번 대회를 통해

코드 흐름을 알게 되었고 코드가 무엇을 뜻하는지 이해할 수 있게 되었다. 이전부터 바라왔던 순간이라 즐거웠다.

over sampling은 back translation 결과가 좋지 않아서 폐기했지만 시간이 더 있었다면 후처리를 통해 양질의 데이터를 만들어낼 수 있었을 것 같다.

model 실험은 여러가지 하이퍼 파라미터 (batch, regression과 classification의 비율, learning rate…) 들을 조정하거나 다양한 모델을 써보는 기본적인 실험을 진행했다. 나의 실험에서 뛰어난 성능 향상은 아쉽게도 없었다.

또한 github 이용을 예전에 얕게나마 해본 적이 있어 어느정도의 지식이 있었다.

이번에 팀원들도 github 이용을 익숙치않아 했는데 내가 아는 부분에 대해서는 도와주고 설명해주면서 함께 진행했다.

모두 큰 문제 없이 github을 이용한 것 같아 다행이었다.

✍🏻 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

이번 대회를 진행하면서

1. 앙상블에는 한계가 존재한다.
2. 동일한 조건 하에서 1가지씩 요소를 바꾸어가면서 실험을 해봐야 어떤 것을 넣었을 때 성능이 좋은지 / 안 좋은지 알 수 있다.
3. 첫 대회여서 이전에 실무 경험이 없던 점

위 3가지의 한계, 깨달음 그리고 아쉬움을 얻었다.

신기했던 것으로, 다양한 모델을 앙상블 하면 성능이 꽤 올라가는 경험을 했다. 정확한 이유는 아직 잘 모르겠지만

다양하기 때문에 각 모델마다 특화된 부분이 달라 상호보완이 된게 아닐까 생각한다.

반대로, 비슷한 모델끼리 앙상블을 진행하면 성능이 미미한 것을 확인했다.

대회 1등, 2등한 팀이 어떻게 진행했는지 들어보니, 역시 앙상블은 한계가 존재하고 여러가지 방법으로 실험해보는 것이 성능을 올리는 요인이구나 깨달았다. 성능을 올리는 것뿐만 아니라 본인과 팀이 성장하기에도 굉장히 좋겠구나 싶었다.

- loss 계산 기법을 갖가지를 사용해보기
- 피어슨 상관계수 결과를 ploting해 결과값만 보는게 아닌, 현재 모델 결과의 방향성을 확인하기
- 더욱 다양한 모델을 사용해보기 등…

✍🏻 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

여러 깨달음을 바탕으로 다음 프로젝트에서는,

더욱 체계적으로 업무 분담을 하고 진행 상황을 살펴볼 것 같다.

모델 실험을 하려면 어떻게 진행하는 것이 가장 효율적일지도 깨달았으니 다음에는

모두 동일한 조건 하에서 하나씩 바꾸어 실험해보고 어떤 요소가 얼마의 성능을 내는지, 어떤 요소를 적용하면 데이터가 0쪽으로 치우치는지 등 제대로된 `분석` 을 진행하고 싶다.

이번 대회에서는 모델 실험을 자율적으로 진행했기에 팀원마다 다 다르고

뒤죽박죽 섞여 어떤 요소가 성능을 높이는지 분석을 할 수가 없었다.

이 아쉬움을 토대로 다음 프로젝트에서는 다르게 진행해볼 예정이다.

그리고 다양한 기법들과 모델을 조사해서 여러가지 실험을 다양하게 할 것이다.

✍🏻 진행 중 ISSUE

실험을 진행하면서 input의 size miss-match, 모델의 Embedding 크기와 현재 나의 모델의 Embedding 차이 issue, 마지막 batch의 사이즈가 다른 남은 부분(rest)으로 인해 생기는 오류 등 에러들이 있었다.

본인은 코드를 작성할 능력이 아직 부족하기에 코드를 작성할 수 있는 다른 팀원이 짠 코드를 바탕으로 진행했다.

코드를 내가 작성하지 않아서인지 오류가 떠도 이해할 수 없어서 아쉬웠다.

그래도 팀원들과 다같이 오류를 살펴보고 소통했기에 대부분 빠르게 해결했다. 하지만, 해결하지 못한 부분도 있어 아쉬움이 남는다.

Personal Report(박승현)

1. 프로젝트 목표

a. 개인 목표

프로젝트 결과가 좋다면 최고이지만, 점수에 연연하기보다 팀에서 마주만 문제들에 대해 끊임없이 고민하는 과정에서 많은 성장을 이룰 것이라고 기대했다. 따라서, 높은 점수보다는 다양한 가설을 세우고 데이터를 분석하면서 여러 실험을 진행하는 것에 초점을 맞췄다.

b. 팀 목표

살아온 배경이 다른 만큼 팀원 간의 지식 편차도 존재했다. 그러나, 편차에도 불구하고 팀원과 원활하게 소통하고 서로 모르는 부분을 상호 보완해주며 서로 최대한의 성장을 이루는 것에 초점을 맞췄다.

2. 실험 & 시도

a. Baseline code 작성

팀원과의 원활한 소통을 위해 코드를 통일했다. pytorch lightning보다 순수 pytorch를 사용해 본 인원이 더 많은 만큼 우선 pytorch로 기본 코드를 작성했으며, 모든 팀원이 손쉽게 이해하도록 코드를 작성했다.

b. Multi-Model Learning

Transformer의 tokenizer, 학습 방식, 학습 데이터 등 다양한 요소로 인해 task에 따라 상이한 성능을 보여주곤 한다. 데이터 자체에 대해 분석하기 전에 같은 hyperparameter와 모델 구조인 상황에서 어떤 모델이 전체 데이터를 잘 표현하는지에 대해 파악하기로 했다. 우선, Tokenizer 같은 경우 모델에 따라 tokenizer의 unknown token 개수와 종류가 달랐으며, 최종적으로 koelectra-base 모델이 가장 좋은 성능을 도출했다. 전체 데이터를 확인해봤을 때 구어체 같은 문장이 많아 구어체를 대상으로 pre-train한 kcelectra나 kpfbert가 우수할 것으로 예측했지만, 결과는 roberta-large보다 성능이 좋지 못했다. 이 결과는 아마 rtt 데이터가 sampled 데이터 보다 많아서 발생한 것으로 추측한다. 그 이유는 rtt 이후의 데이터는 맞춤법과 띄어쓰기가 어느 정도 지켜진 데이터이기 때문이다.

c. Multi-Task Learning

이번 대회의 STS는 0과 5 사이의 유사도 값을 예측하는 문제를 해결해야 하기에 회귀 문제로만 볼 수 있다. 회귀로만 문제를 해결할 수 있지만, 데이터에서 제공한 binary label을 사용해 Binary Cross Entropy Loss(BCE)를 적용한다면 값 보정 효과로 더 좋은 결과를 도출할 수 있을 것이라는 확신을 다른 연구[1]에서 얻었다. 실제 실험 결과, 회귀 문제에서 가장 많이 사용되는 MSE Loss만 사용하기보다 MSE loss와 BCE를 함께 적용했을 때 dev data에서 pearson 계수가 0.001~0.002 정도 향상되는 것을 확인했다. 또한, MSE Loss와 BCE의 적용 비율이 성능에 미치는 영향도 실험해봤으나 유의미한 결과는 도출하지 못했다.

d. Outlier Checking

인간도 풀지 못하는 문제를 기계보고 해결하라고 명령하면 안 된다고 생각한다. 따라서, 전체 데이터를 확인하여 사람도 유사성에 대한 판단이 어려운 경우 이를 outlier로 정의하고 학습에서 제외시키기로 했다.

e. Oversampling & Undersampling

데이터의 분포를 확인했을 때, 0~0.4 사이의 label이 전체 데이터에서 30% 정도를 차지하고 있는 반면 다른 0.6~5 사이의 label은 고르게 분포하고 있어 데이터 Imbalance가 발생했다는 것을 확인할 수 있었다. 이 문제를 해결하기 위해, oversampling과 undersampling을 실시했다.

- Oversampling

데이터는 많으면 많을수록 유리하다. oversampling에는 AEDA, back-translation 등 다양한 방법이 존재하지만, back-translation만 진행하기로 결정했다. AEDA 등 문장에 특수 문자를 추가하거나 문장 내 단어 순서를 변경하는 방식으로 데이터를 증강하는 경우 문장의 의미를 변경시킬 수 있다. 실제로 실험한 결과, AEDA를 진행한 경우 많은 데이터의 의미가 변질되는 것을 확인할 수 있었다. STS의 경우 문장의 의미적 유사도를 예측하는 것으로 의미가 바뀐다면 이에 상응하는 label도 변동되야 한다. 따라서, AEDA와 같은 문장의 의미를 퇴색시키는 증강 기법은 사용하지 않았다. 최종적으로 back-translation만 남았는데 파파고로 back-translation(ko-en, en-ko)을 진행했다. back-translation을 진행한 이후의 데이터와 기존 데이터를 대조해본 결과 반 이상 데이터의 의미가 살짝 변경된 것을 발견할 수 있었다. 데이터의 의미가 변경됐음에도 불구하고 실험을 진행해봤는데, 제대로 예측하지 못한 데이터의 대부분이 back-translation 데이터여서 back-translation을 적용하지 않기로 했다.

- Undersampling

Oversampling으로 데이터의 분포를 맞추기 어렵다고 판단해 데이터가 과하게 많다고 생각되는 부분을 줄여주기로 했다. 데이터는 문장으로 이뤄져 있기에 분포 기반으로 데이터를 sampling하기 어려웠다. 따라서, 가장 쉬운 방법인 random undersampling을 실시했다. 그 결과, 전체 데이터를 학습시키는 방식보다 성능이 향상됨을 알 수 있었다.

f. K-Fold

K개의 fold를 만들어서 진행하는 교차검증 방식으로 앙상블의 한 부분이라고 볼 수 있다. 다양한 모델에 대해서 k-fold 방식의 학습을 진행했지만, 결과는 생각 외로 좋지 않았다.

g. Learning by Domain

앞서 (b)에서도 언급했듯이, 데이터와 task에 따라 강점을 보이는 모델이 다르다. 데이터를 확인해 본 결과, 데이터를 slack, nsmc와 petition로 분류할 수 있었고, slack과 nsmc 데이터는 구어체에 가까웠고 petition 데이터는 문어체에 가까웠다. 따라서,

구어체로 pre-train한 kcelectra와 kpfbert로 slack과 nsmc 데이터를 학습시키고, 문어체 느낌으로 pre-train한 koelectra와 roberta-large로 petition 데이터를 학습시켰다. 실험 결과, 한 모델로 학습시키는 것보다 좋지 못한 성능을 보였다.

h. Incorrect Result

Fine tuning한 모델로 validation data에 대한 예측 값을 도출해 ground truth와 비교해봤다. 결과가 1 혹은 1.5 이상 차이나는 이상치를 중점으로 확인했고 그 과정에서 오타가 있는 문장을 사람과 달리 잘 예측하지 못하고 문장 순서에 따라서 판단을 제대로 못하는 경우가 있었다. 그러나, 이 데이터에 대해서 추가 학습을 시켜도 유사한 값이 나올 것으로 추측되어 추가 실험은 진행하지 않았다.

i. 앙상블

다양한 방식으로 앙상블을 진행했다. 가장 성능이 좋았던 7개의 모델, 종류가 다른 5개 모델, kfold만 한 모델 등 다양한 방식을 앙상블을 진행했는데, 확실히 단일 모델로만 결과를 도출하는 것보다 좋은 성능을 낼 수 있었다. 앙상블을 통해 각 모델에서 제대로 학습시키지 못한 부분을 보완시켜 최적의 값을 도출한 것으로 추정하고 있다.

3. 한계 및 고찰

a. 제한적인 시간

분업했음에도 불구하고 제한적인 시간으로 인해 시도해보고 싶었던 것들이 존재했지만 모두 실험에 옮기지 못했다. 또한, 부스트캠프 내에서 첫 프로젝트 진행이다보니 시간 관리가 제대로 되지 않았다.

b. 생각보다 너무 적은 epoch 횟수

너무 과도한 epoch을 학습시킨다면 overfitting될 것이라는 생각에 epoch을 최소화했다. 그 결과 대부분의 실험은 5~12 epoch으로 진행됐다.

c. Regression loss function

Loss function에는 MSE Loss 뿐만 아니라 n1 Loss, hober Loss 등이 존재한다. 이번 프로젝트에서는 regression할 때 MSE Loss만 사용했는데, 다른 loss도 사용해봤으면 모델의 성능을 올릴 수 있지 않았을까 싶다.

d. More model & freeze

조사가 부족한 탓일수도 있지만, T5, SMART 등 다른 SOTA 모델을 활용해 실험을 진행해보지는 못했다. 또한, 딥러닝 모델을 학습시킬 때 첫 layer 혹은 마지막 layer를 freeze시키는 방법이 있는데 그 방법을 시도해보지 못해 아쉽다.

e. 전략적인 앙상블

비록 나름 전략적으로 앙상블을 진행했지만, 모델 별로 도출한 pearson 뿐만 아니라 실제 예측 분포를 기반으로 앙상블을 진행했다면 데이터를 더 잘 표현했을 것이다.

f. 데이터 & 모델 분석

데이터 증강하는 부분에 있어서 AEDA와 back-translation에만 집중했다. 그러나 이 방법 말고도 STS task에는 sentence1과 sentence2의 위치를 변경해서 학습시키는 방법도 있다. AEDA와 back-translation에 집중해서 더 많은 방법을 시도해보지 못해서 아쉬움이 남았다. 또, 어떤 모델이 우리 데이터에서 어떤 부분에 강점을 보이는지 분석을 진행했다면 성능을 더 향상시킬 수 있었을 것이다.

g. 현업에서 이 모델을 사용할 수 있을까?

이번 프로젝트에서 우리가 사용한 모델은 대부분 large model이라고 할 수 있다. 이 모델들을 빠르게 서비스에 적용하기에는 리소스 문제가 발생한다. 따라서, 대부분 기업은 모델의 크기를 줄이거나 knowledge distillation 방식으로 모델의 크기를 줄이고자 노력한다. 비슷한 성능을 내면서 모델 크기를 줄이는 연구가 더 많이 진행될 필요가 있다고 생각한다.

h. AI에는 "MUST"가 존재하지 않는다.

Roberta 논문에서 언급한 바에 따르면, large model에서 큰 batch를 사용하면 더 좋은 성능을 도출할 수 있다고 한다. 그러나, 실제로 우리가 실험한 바에 따르면, batch size가 4인 경우 가장 우수한 성적을 보였다. 또한, NLP에서 좋은 데이터를 증강시키기 위해 back-translation을 많이 사용하는데 실제 실험 결과에서는 좋지 못한 성능을 보였다.

4. 참고 문헌

[1] T. Gong et al., "A Comparison of Loss Weighting Strategies for Multi task Learning in Deep Neural Networks," in IEEE Access, vol. 7, pp. 141627-141632, 2019, doi: 10.1109/ACCESS.2019.2943604.

Wrap up Report

작성자 : 최동민_T4218

✍️ 학습목표

개인적인 학습목표로는 성능 개선에 너무 치우치기 보다, 모델과 데이터를 다루어보는 경험과 실제로 특정 Task에 집중하고 고민하는 과정 속에서 배움을 얻는 것이 목표였습니다. 구체적으로는, 모델과 데이터를 개선하기 위해 다양한 시도들을 체계적으로 진행하면서 혹여나 실패하더라도 왜 실패했는지 고민하고 분석하는 것이 목표였습니다.

팀원들과 논의해본 결과 비슷한 목표를 갖고 있었고, 단순 성능보다는 그 과정에 집중해 다양한 시도들을 하면서 결과를 분석하고 이해하는 것에 초점을 맞추는 것으로 합의하였습니다.

✍️ 이번 대회에서 맡은 것

1. KLUE-STS 논문 리뷰

이번 대회는 STS Task에 대한 대회였다. STS에 알맞은 모델과 데이터 구성 방식을 파악하기 위해 KLUE 논문 중에서 특히 STS 부분에 집중하여 읽고 정리하고 팀원들에게 설명하는 역할을 맡았다. KLUE 논문에서는 데이터의 라벨링 방식, 데이터 source가 의미하는 바, 효과적인 Tokenization 방법, 쓸만한 PLM, 그에 따른 fine tuning방법과 hyperparameter 예시에 대한 인사이트를 얻을 수 있었다.

특히, 이번 대회가 KLUE-STS와 유사한 바가 많은 만큼 데이터 source (rtt, sampled)가 의미하는 바를 논문에서 파악할 수 있었고, 그것을 데이터 전처리 혹은 후처리에 활용하는 방법으로까지 이어질 수 있었다.

데이터의 라벨링 방식에서도 여러 시도를 생각할 수 있었다. 사람이 라벨링한 방식을 파악함으로써 그것을 모델의 앙상블에 적용하여, 7개의 모델이 예측해낸 라벨들 중 평균에서부터 먼 순서대로 2개의 결과값을 제외한 후 평균을 내는 방식으로 효과적으로 활용하였다.

2. Multi-classification

한편, 이 문제를 regression으로만 접근할 게 아니라, 0,1,2,3,4,5 점으로 분류하고 그 Loss 함수를 regression의 loss함수와 합쳐 regression과 multi-classification 2개의 task로 접근하는 방법도 있을 수 있겠다는 팀원의 아이디어도 KLUE 논문 리뷰로부터 나왔다. 그러나 이러한 방식의 실험준비가 허술하여 유의미한 성능개선을 이끌어내지는 못하였다.

또한, 먼저 0,1,2,3,4,5 점으로 분류하는 과정을 거치고 이후에 평균을 내는 방식의 아이디어도 나왔다. 이는 사람이 직접 라벨링할 때의 방법을 답습한 것으로, gold-label의 라벨링 방식을 따라하는 것으로 상관계수를 높일 수 있을 것이라 기대하였다. 그러나 대회기간내에 시간이 부족하여 직접 실험을 해보지는 못하였다.

3. prediction check

학습된 모델을 가지고 train data와 validation data를 예측해보고, 정답과 비교하여 제대로 예측하지 못하는 것들로 부터 특징을 찾아내고, 이상치를 잡아내는 작업을 진행하였다. 예측치와 정답이 1.5, 1, 0.8 이상 차이나는 것을 각각 뽑아보고 시각화하였다. 그 과정에서 오타가 있는 문장을 사람과 달리 잘 예측하지 못하는 점, 어투의 차이와 유의어 사용으로 인해 유사도 점수를 낮게 예측하는 경향이 있다고도 파악되었다. 또한, 문장 안에서 단어의 순서로 인해 뜻이 크게 달라지는 경우도 예측하지 못하는 것으로 파악했다.

4. PyTorchLightning code

팀 내에서 PyTorch를 baseline code의 Framework로 사용하였다. 그러나, WandB swap, config를 활용한 실험 관리 등 PyTorchLightning으로 진행하면 편리해보이는 시도들이 있었고, 다른 팀원과 함께 PyTorchLightning으로 앞선 2개의 feature를 포함하여 code를 구현하였다.

✍️ 아쉬운 점

우선 첫 번째로, 명확한 기준이 될 baseline model과 setting이 없었다는 점이 아쉬웠다. 병렬적으로 아이디어와 실험이 진행되는만큼 그것이 유의미한지 무의미한지, 유의미하다면 얼마만큼 유의미한지를 파악할 수 있을 대조군이 필수적이다. 그러나 명확한 대조군이나 기준이 없이 쫓기듯 실험을 진행하다보니 중반부터는 실험의 성능 검증과 정량적인 측정이 제대로 되지 않았다.

두 번째로, 언어모델(PLM) 그 중에서도 유명한 한국어 언어모델만을 사용하였는데, 내부구조나 작동방식을 정확히 이해하지 못한 상태로 사용하였기에 그 안에서 응용할 수 있는 것의 한계와 아쉬움이 분명히 존재하였다.

세 번째로 multi-classification, 문장에서 조사를 떼고 유사도 검사를 하는 방식 등 시간이 부족하여 해보지 못한 실험들이 아쉬웠다.

마지막으로, Github와 같은 협업 tool을 제대로 사용해보지 못한 것이 아쉽다.

✍️ 제언

STS를 multi-classification Task로 상정하고 진행하는 것과, 문장에서 조사를 빼고 유사도 검사를 진행하는 방식 등 해보지 못한 실험들을 시도해보고 넘어갈 수 있겠다.

PLM과 NLP 이론적인 배경에 대해서 더 자세하게 이해하는 것이 그 내부에서 응용할 수 있는 다양한 방법들을 생각해내는데 필수적일 것이다.

다음 프로젝트에서부터는 명확한 기준과 실험마다 정확한 대조군을 정하고 진행할 것이며 팀원들과 다른 사람들이 알아볼 수 있게 최대한 세세하게 작성할 것이다.

또한, 프로젝트의 기본이 되는 template을 먼저 작성할 것이고 그에 따라 실험을 편리하게 할 수 있도록 준비할 것이다.

마지막으로, Github와 같은 협업 tool을 적극적으로 사용하여 그 경험을 쌓고 협업 효율을 극대화 시킬 것이다.

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

우리 팀은 처음부터 학습목표를 정할 때, 대회 순위에 연연해하지 않고 개인이 스스로 공부하고 실력향상이 될 수 있는 발판으로 정하였다. 이는 나의 목표와도 같았다. 단순히 모델을 바꾸고 실험하는데 집중하지 않고, 진행방법을 이해하고 본질적으로 어떤 부분이 가장 큰 효과를 미칠지 다양한 시각으로 고민해보며 실험을 진행하였다. 또한, git을 최대한 활용하여 협업을 하는 것을 목표로 하였다.

○ 나는 어떤 방식으로 모델을 개선했는가?

모델을 개선하여 얻을 수 있는 가장 큰 변화는 '전처리'라고 생각한다. 첫째도 전처리, 둘째로 전처리이다!! 전처리는 단순히 모델의 하이퍼파라미터를 튜닝하고, 모델을 바꿔가며 실험하는 것보다 훨씬 좋은 효과를 낼 수 있을 것이라고 생각하였다. 전처리에 해당하는 데이터 EDA, 데이터 전처리, tokenizer를 분석하며 최대한 모델에게 좋은 input을 주려고 노력하였다.

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

다양한 전처리 실험에 대한 정확한 성능평가를 위해 독립변수와 종속변수를 확실히 설정해두고 실험을 진행하였다. 모델의 input으로 들어갈 데이터를 전처리하는 과정에 많은 시간과 노력을 쏟았다. 최종적으로 문장의 유사도를 측정하는데 있어서 UNK token을 최대한 없애줌으로써 문장의 의미를 살릴 수 있었으며, 띄어쓰기와 특수기호 제거 등의 전처리를 포함하여 모델이 이해할 수 있는 가장 좋은 방법으로 input을 넣어주었다. 멘토님의 말씀 중 '좋은 데이터를 넣어야 좋은 성능이 나온다' 라고 말씀하신 것처럼 모델이 이해하지 못하는 부분을 전처리를 통해 그 의미를 살려주었을 때, 가장 좋은 성능을 얻을 수 있었다.

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

전에는 모델이 좋은 결과를 결정한다고 생각하여 단순히 모델을 변경하고 하이퍼파라미터를 튜닝하는데 대부분의 시간을 보냈다. 하지만 이 방법은 전체적인 구조에서 성능을 향상시킬지를 본질적으로 고려하지 않았으며, 데이터가 중요하다는 점을 놓치고 있었다. 모델 자체의 구조도 제대로 파악하지 않고 실험했기 때문에 대회 후 남는 것이 없었다. 이번 대회에서는 이전의 실수를 반복하지 않기 위해 최종적으로 모델의 성능을 높이는데 어떠한 방법이 가장 큰 영향을 미칠지 고민해보았고, 실제로 멘토님께서 말씀해주신 다양한 방법론들을 고민해보고 적용하여 실험해봄으로써 본질적으로 과정을 이해하고 적용할 수 있었다. 특히 전처리가 중요하다는 것을 깨닫고, 이 부분에 대해 많이 공부하고 실험을 해보았다. 이를 통해 성능 향상에 대한 자세한 이유를 나 스스로 이해하며 실험에 대한 확실한 답을 얻을 수 있었다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

아직 나의 실력적인 부분이 아쉬웠다. 모델의 input부터 output까지 전체적인 코드를 작성해본 경험이 없어서 코드를 이해하고 수정하는데 어려움이 있었다. 또한, 실험할 수 있는 시간이 짧았기에 더 많은 방법들을 실험해보지 못한 점이 매우 아쉽다. 아직은 git에 대한 사용법이 미숙하여 충돌이 자주 일어났지만 많이 배우며 실력향상이 있었다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

대회에서 제공하는 baseline을 내 방식으로 재구성해볼 것이다. 전처리에 관한 더 많은 방법을 생각해보고 tokenizer의 구성 방식을 또한, 해당 task의 sota model을 조사해보고 논문을 읽어보며 더욱 다양한 방법들을 생각해볼 것이다.