Wrap-Up Report

2023. 04. 10 – 2023. 04. 21

NLP-01 연어들
박경택T5079 | **박지은**T5096 | **송인서**T5115 | **윤지환**T5132

github.com/boostcampaitech5/level1_semantictextsimilarity-nlp-01

1. 프로젝트 개요

1) 주제 : Semantic Text Similarity (STS) 태스크 수행 모델 성능 튜닝

2) 주요 목표

- 주어진 베이스라인코드를 바탕으로 STS를 수행하는 모델의 성능을 튜닝
- 지금까지 강의를 통해 배워 온 이론을 바탕으로 다양한 모델과 하이퍼파라미터를 적용해 실험
- EDA(Exploratory Data Analysis)를 바탕으로 데이터의 특성을 파악하고 적절한 전처리 및 증강 적용

3) 프로젝트 환경

H/W	**S/W**	
AIStages(stages.ai) 제공 서버 Tesla V100	**모델 실험**	**EDA**
	(OS) Ubuntu 18.04.5 pytorch 2.0+cu117 Pytorch-lightning 2.0.1 HuggingFace scikit-learn	numpy pandas seaborn plotly matplotlib

4) 프로젝트 팀 구성 및 역할

박경택	**박지은**	**송인서**	**윤지환**
프로젝트 모듈화 데이터 증강 k-fold 교차 검증 GitHub trouble shooting	초기 EDA 데이터셋 버전 관리 데이터 전처리 및 데이터 증강 Notion template 구축	서버 연결 WandB Alert 작성 및 연동 GitHub trouble shooting Wrap-Up Report 종합	WandB sweep 작성 데이터 전처리 GRU 적용 사후분석 및 모델 리서치

5) 협업 툴

소통	Slack / Zoom / Notion
실험 및 버전 관리	Weight & Biases / Git + GitHub / HuggingFace Datasets

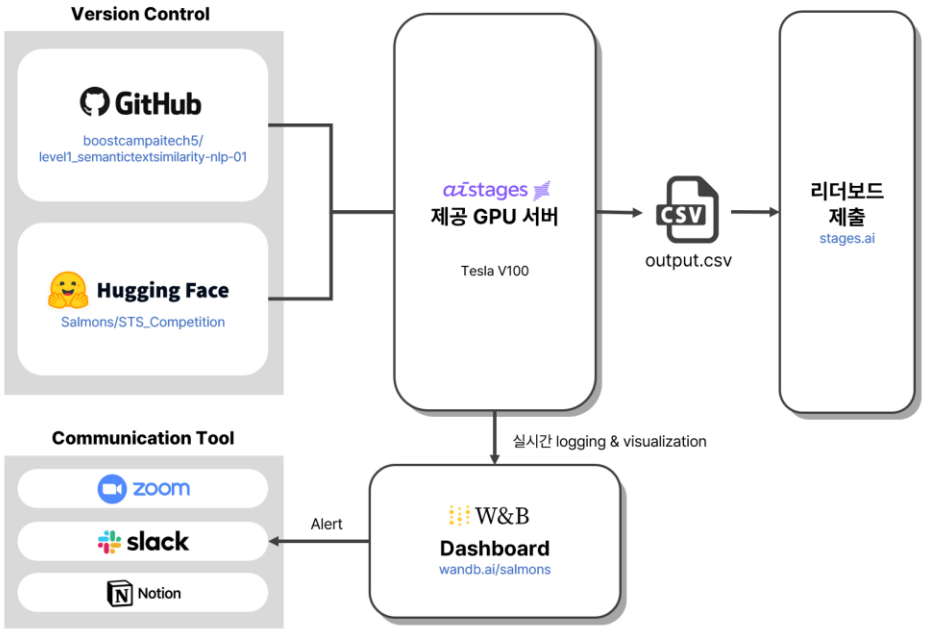
6) 프로젝트 모듈 구조

github.com/boostcampaitech5/level1_semantictextsimilarity-nlp-01

`train.py`		학습을 위한 메인 스크립트
`inference.py`		학습 모델의 평가 및 추론을 위한 스크립트
`config.json`		학습 설정 및 하이퍼파라미터 설정을 위한 JSON
`lr_scheduler.py`		학습률 스케줄러
`sweep.py`		하이퍼파라미터 서치를 위한 스크립트

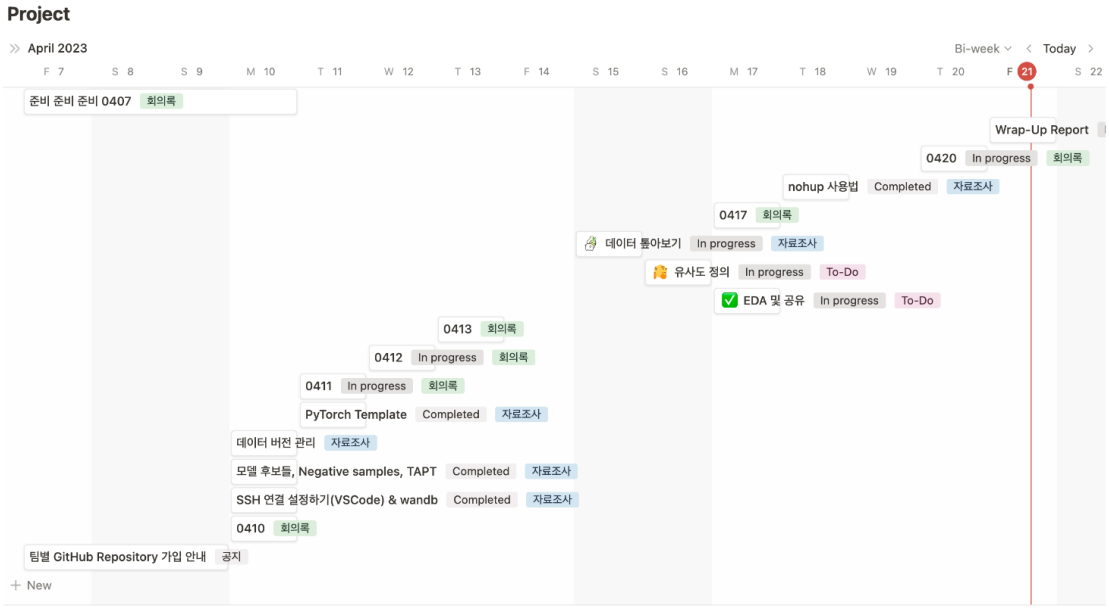
dataloader/	dataset.py dataloader.py	데이터셋 및 데이터로더 설정 모듈
data/		입력 데이터 디렉토리
models/	model.py	모델, 손실함수, 최적화 알고리즘, 평가 지표 정의 모듈
saved/	models/	학습이 완료된 모델 저장 디렉토리
utils/	util.py	유틸리티

7) 프로젝트 전체 구조



2. 프로젝트 수행 절차 및 방법

1) 프로젝트 타임라인



- 피어세션마다 Notion에 회의록 작성 및 주제별(자료조사/할 일 목록(To-do))로 task 관리

2) 프로젝트 과정

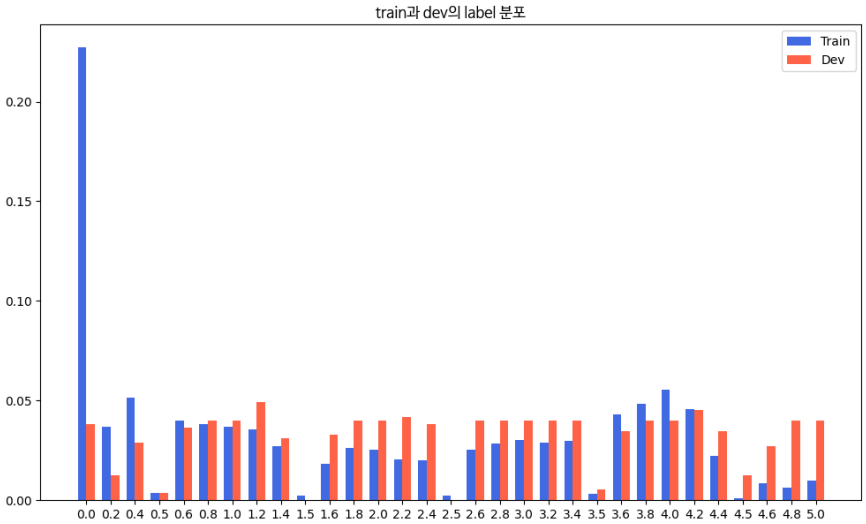
① 사전 조사	a. 문장 유사도 측정 태스크 조사 b. 미팅 계획 수립 c. 프로젝트 정책 수립
② 베이스라인 코드 탐색	프로젝트 관리를 위한 모듈 분리
③ 실험 및 협업 툴 숙지 및 초기 설정	GitHub / HuggingFace / WandB
④ 데이터 EDA	a. 점수(y)에 따른 데이터 분포 b. 출처(source)에 따른 데이터 분포 c. 모델 예측과 정답 간의 오차 분석을 위한 데이터 분포
⑤ 실험 설계	a. 데이터 버전별 실험 b. 모델 버전별 실험 i. klue/roberta-large ii. snunlp/KR-ELECTRA-discriminator iii. Bi-directional GRU layer adding
⑥ 실험 및 결과 분석	

- 의사결정 과정과 실험은 Notion과 WandB를 사용해 기록하였으며 Slack과 Zoom을 활용해 소통했음
- 문제가 발생한 경우 Slack채널 혹은 Slack huddle, Zoom을 통해 팀원과 소통하며 트러블슈팅을 진행했음

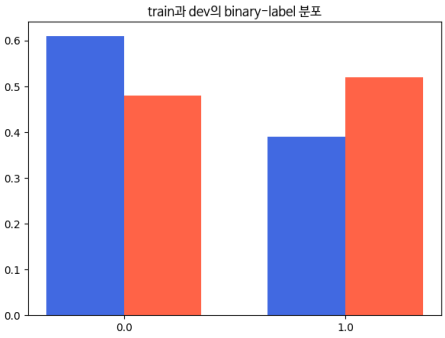
3. 프로젝트 상세

1) EDA

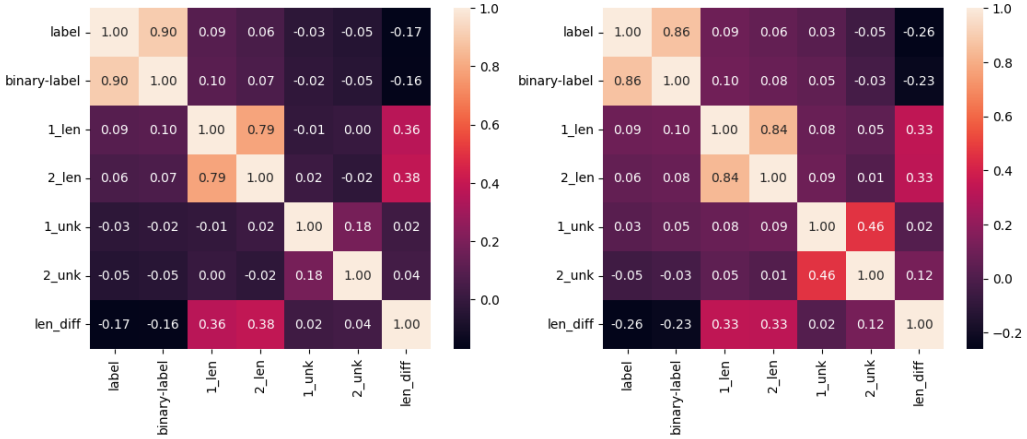
- 데이터는 train/dev/test로 나뉘어 있으며 그 비율은 85/5/10임
- 결측치는 포함되어 있지 않았음
- 데이터는 문장의 출처를 바탕으로 세 개의 카테고리(nsmc, petition, slack)로 나눌 수 있으며 고르게 분포되어 있었음
- 문장을 토크나이징한 후 문장 당 토큰 및 unk 토큰의 개수를 분석한 결과 train 및 dev 데이터의 sentence_1과 sentence_2는 평균적으로 14.7~15.0개의 토큰으로 구성되어 있었으며, 평균적으로 0.02개의 unk토큰을 포함하고 있음
- label은 0부터 5까지의 값을 가지며 소수 첫째자리까지 매겨져 있음
- train 데이터셋에서는 label이 0인 데이터가 전체 데이터의 약 23%를 차지하는 class imbalance가 확인되었으며, dev 데이터셋은 상대적으로 label이 고르게 분포하고 있었음



- 유사도가 2.5점 미만인 경우 0점, 2.5점 이상인 경우 1로 변환하여 이진 레이블로 나타낸 결과, 0과 1의 비율은 train 데이터가 61:39, valid 데이터가 48:52였음



- 문장의 길이와 두 문장 간의 길이 차가 약한 양의 상관관계를 가지고 있음을 확인함



2) 데이터 전처리

- 실험 과정에서 특수문자, 같은 문자의 반복, 영문, 이모티콘 등을 포함한 데이터에 대한 예측 성능이 좋지 않았음을 확인
- 이에 따라 다음과 같은 전처리를 수행
  - 한글과 숫자, 일부 특수문자(?, ;, :, ^)를 제외한 나머지 문자 제거
  - 이모티콘의 역할을 하는 ㅋ, ㅎ, ㅠ 제거
  - 연속된 특수문자는 1개의 문자로, 연속된 문자는 2개로 치환
- 그 결과 target sample에 대한 예측 성능이 개선되었음을 확인함

index	sentence 1	sentence2	label	before	after	영향
4	다음 밥스테이지가 기대됩니다~ ㅎ	다음 후기도 기대됩니다~~	1.4	2.5	2.0	good
173	너무 귀엽네요 ㅋㅋ	스몰 u 너무 귀엽네요ㅋㅋ	2.4	3.3	3.3	-
233	두부라고 하니 두부로 보이는 매직...	두부처럼 생긴 마법...	1.6	2.7	2.2	good
390	편보다 편이 더 재밌는 영화... 편부터는 보지마	본편보다 영화가 더 재밌어... 본편은 보지마	1.6	4.0	3.7	good
412	ㅋㅋ넘 잼썼어요 진짜ㅋㅋ	ㅋㅋ 정말 재밌게 봤어요 ㅋㅋ	2.4	3.6	3.8	bad
433	베란다도 너무 예뻐요!!	우와 가을!!! 다 너무 예뻐요!!!	0.8	1.6	1.3	good
438	나뭇잎을 문 털복숭이라니ㅠㅠㅠ	나뭇잎을 물어뜯는 털입니다 ㅠㅠ	2.0	2.8	2.6	good

3) 데이터 증강

- Reverse Augmentation
  - sentence_1과 sentence_2의 위치를 바꾸는 방식으로 증강함
  - train 데이터의 23%를 구성하고 있는 label=0을 제외한 나머지 데이터 7205개를 대상으로 증강을 진행함

- Style Transfer Augmentation
    - Korean Smile Style Dataset으로 fine-tuning한 BART 모델을 이용해 train 데이터셋의 sentence_1을 격식체(formal)로 변형함
    - KLUE-NLI 데이터셋과 KorSTS 데이터셋으로 fine-tuning한 KR-SBERT를 이용해 원본 문장과 변형된 문장을 인코딩한 후 sentence embedding간의 코사인 유사도를 산출해 값이 0.85 이상인 데이터를 선별, 총 4300개의 데이터를 생성함
- Back Translation(데이터 버전 4.1.0)
    - 네이버 파파고(papago.naver.com)에서 동적 웹 스크래핑(Selenium을 사용)을 통해 주어진 sentence를 영문 번역 후 다시 한국어 문장으로 번역하여 증강함
    - 데이터 분포상 label의 수가 얼마나 부족한지를 우선순위로 역변환을 진행하였으며, 샘플링을 통해 총 2300여개의 데이터를 생성함

데이터셋 버전 상세

버전	상세
1.0.0	원본 데이터셋
1.1.0	googletrans를 사용해 한국어 데이터를 영어 데이터로 번역해 구성한 데이터셋. 번역 실패에 따른 결측치 포함
2.0.0	원본 데이터셋 + reverse augmentation 데이터셋
3.0.0	원본 데이터셋 + style transfer augmentation 데이터셋
4.0.0	원본 데이터셋 + reverse augmentation 데이터셋 + style transfer augmentation 데이터셋
4.1.0	4.0.0 데이터셋에 backtranslation 데이터 추가

*HuggingFace 비공개 팀 저장소에서 데이터 버전 관리를 하였으며, 버전 관리 규칙은 semantic versioning 방식을 따름.

4) 실험 요약

- 모델 선정 및 분석
    - 총 세 가지 버전의 모델(klue/roberta-large, snunlp/KR-ELECTRA-discriminator, Bi-directional GRU layer adding)을 실험했으며 최종적으로 4가지 output을 앙상블해 최종 제출함
- 최종 제출 과정
    ① 각 모델의 성능을 어느정도 보장해주는 수준(>LB 0.915)의 하이퍼파라미터 조합을 탐색(validation data prediction을 사후분석한 결과, valid pearson, valid loss 확인)
    ② 데이터 증강, 전처리, 다양한 모델링 기법에 대한 가설을 세우고 적용해보며 평가지표 및 사후 분석을 통해 적용시킬지를 결정
    ③ WandB sweep을 통해 하이퍼파라미터 튜닝을 진행(lr, epochs, batch size, loss function, optimizer)
    ④ 모델 학습 후 결과를 hard voting

klue/roberta-large	base 모델에 비해 LB 기준 0.02의 성능향상(LB 0.896)을 보였으나 사후분석 결과 일관성 있는 예측을 보여주지 않고 불안정한 모습을 보임 CUDA OOM 문제가 있어 mixed precision training을 활용해 학습함
snunlp/KR-ELECTRA-discriminator	ELECTRA를 32GB의 한국어 데이터로 사전학습한 모델 RoBERTa에 비해 가볍고 빠르며 모든 입력 토큰을 사용하기 때문에 문맥 정보를 더 잘 파악할 것이라는 가정 하에 실험함

	전반적인 성능이 klue/roberta-large에 비해 좋았고 사후분석 결과의 경향성도 안정적이었으며, 예측결과도 일관성 있게 유지되는 장점이 있었음 단일 모델 학습 기준으로 가장 높은 성능(LB 0.9215)를 보였음
Bi-directional GRU adding	ELECTRA가 사전학습에 사용한 데이터의 분포 및 형태가 대회 데이터셋과 다르기 때문에 문맥적 유사도를 세세하게 예측하는 데에 어려움이 있을 것으로 판단, 양방향 layer를 추가해 의미론적 유사도와 문맥정보의 복잡한 연관성을 파악하고자 함 앞의 ELECTRA 모델의 output hidden layer 2개를 Bi-directional GRU 모델의 input으로 연결한 후 tanh activation, dropout, linear layer를 추가해 학습시킴 LB 기준 성능은 0.9188로 크게 향상되진 않았으나 나머지 모델과 다른 경향성을 보여주고 있어 최종 앙상블에 포함하였음

5) K-fold 교차 검증

원본 데이터에서 class imbalance를 확인했고, 크게 2가지 방법으로 K-fold 교차 검증을 적용해 실험했음

1. train.csv와 dev.csv를 합친 후 0.8 비율로 total set을 구성한 후 K-fold 교차 검증을 적용
   → train, dev, test의 분포가 같아지게 하는 가정이 포함되어 있어 기각하였음
2. train.csv에 대해서만 교차 검증을 적용

6) 앙상블

- snunlp/KR-ELECTRA-discriminator 단일 모델, Bi-directional GRU adding, data ensemble, K-fold 적용 모델 4가지의 모델 output의 평균을 구해 최종 결과 제출함. weighted sum역시 시도해보았으나 큰 차이는 없었음
- 성능과 함께 valid data를 통한 예측분포가 다양하게 반영될 수 있도록 loss function, optimizer, 데이터 버전을 조합해 최종결과를 도출함
- 최대한 일반화 성능을 개선하는 것을 목표로 전략을 세움
- 앙상블 결과 최고성능을 달성함

4. 프로젝트 결과

최종 public 점수 0.9245, private 점수 0.9313 달성

5. 자체 평가 의견

잘한 점들

- 실험 설계에 있어 가설을 수립한 후 이유 있는 실험을 진행하고자 노력함
- 실험 시 가설 검증을 위해 직접 결과들을 확인하며 방향성을 정해 나갔음
- GitHub과 HuggingFace Datasets를 이용해 코드 및 데이터 버전 관리를 진행했음
- WandB와 사후분석을 통해 실험 결과에 대한 객관적인 분석을 시도했음

시도했으나 잘 되지 않았던 것들

- 수립한 모든 가설을 검증해보지 못했음
- lr scheduler, gradient accumulating, focal loss와 같은 기술적인 방법들을 시도해보지 못함
- 데이터의 분포를 기반으로 한 학습 전략이 세워지지 않음

아쉬웠던 점들

- flake8, black, GitHub flow를 적용해보지 못했음

- 데이터 분석을 바탕으로 한 유사도 정의가 잘 이루어지지 못했음
- 팀원 간 프로젝트 진행 상황이 유연하게 공유되지 않았음
- 본격적인 프로젝트 시적 전 전체적인 timeline을 설정하지 않아 각 task에 대한 시간 분배가 적절하지 않았음

프로젝트를 통해 배운 점 및 시사점

- 프로젝트 설계 및 구조화가 매우 중요함
- 시간, 컴퓨팅 자원 뿐 아니라 그 밖의 한정된 시간을 어떻게 잘 배분하여 결과를 낼 것인지에 대한 논의가 필요함
- 팀원 간의 진행 상황을 원활하게 공유하고 파악할 수 있는 방법이 필요함

6. 참고자료

- https://huggingface.co/docs/datasets/index
- https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
- https://huggingface.co/heegyu/kobart-text-style-transfer
- https://ws-choi.github.io/blog-kor/bidirectional-rnn-in-pytorch/