문장 간 유사도 측정

Semantic Textual Similarity(STS)

Wrap-up 리포트

2023.04.12 ~ 2023.04.20

NLP - 10조(데굴데굴)

곽민석_T5014

이인균_T5150

임하림_T5178

최휘민_T5221

황윤기_T5229

1. 프로젝트 개요

a. 프로젝트 주제

Semantic Textual Similarity(STS) 프로젝트는 두 개 이상의 문장 사이의 의미 유사성을 측정하는 것을 목적으로 한다. 이를 통해 자연어 처리 모델의 성능을 평가하고, 문장 간의 유사성을 분석하는 등의 다양한 응용이 가능할 것으로 예상된다.

이 프로젝트에서는 문장 두 쌍을 하나의 데이터로 받아 총 약 11,000개의 제한된 데이터를 이용하여 최대한 성능을 끌어올리는 것을 목표로 한다.

b. 프로젝트 구현 내용, 컨셉, 교육 내용과의 관련성 등

- 문장 쌍과 유사도 라벨을 입력 받아 모델을 훈련시키고, 새로운 문장쌍이 들어왔을 때 유사도를 예측하는 모델을 구현하였다.
- EDA를 통해 각 데이터의 분포를 이해하고 효과적인 학습을 위해 데이터 증강 또는 샘플링을 통해 학습 결과를 더욱 향상 시켜줄 수 있음을 이해한다.
- HuggingFace 라이브러리를 통해, 사전 훈련된 모델('RoBERTa', 'KcELECTRA')을 불러와 미세조정 하는 방식으로 모델을 학습하였다.
- 제공된 전체 데이터를 [학습 / 검증 / 테스트]로 나누고, 학습데이터를 통해 모델을 학습하고, 검증데이터를 통해 모델의 성능을 임시로 평가한다. 그 값을 이용해 모델의 최적화를 진행하고, 최종적으로 테스트데이터를 이용해 모델의 성능을 평가한다.
- NLP 도메인에서 가장 좋은 성능을 내는 Transformer기반의 모델을 이해하고, 해당 모델들을 학습하는 방법과 모델의 구조에 대해서 이해한다.
- 해당 모델을 튜닝하면서, pre-trained model에 대해 이해하고 가장 좋은 하이퍼 파라미터를 찾아나가는 과정을 익히고, 효율적으로 관리하는 방법에 대해서 고민해볼 수 있었다.
- 각 모델의 예측 결과를 비교하여 차이점을 평가하고, 각 예측값에 대해 모델마다 가중치를 두어 적절한 조합으로 최종 예측 결과를 도출하였다.

c. 활용 장비 및 재료(개발 환경, 협업 tool 등)

(팀 구성 및 컴퓨팅 환경) 5인 1팀으로 각자에게 지원되는 V100 서버를 활용, 각자 VSCode 혹은 PyCharm 등 편한 IDE를 사용

(협업 환경 및 의사 소통) Notion, Github, Wandb, Slack, Zoom, Gather

d. 데이터셋 및 프로젝트 구조

데이터셋의 구조는 다음과 같다.

id	source	sentence_1	sentence_2	label	binary-label
boostcamp-sts-v1-train-000	nsmc-sampled	스릴도있고 반전도 있고 여느 한국영화 쓰레기들하고는 차원이 다르네요~	반전도 있고,사랑도 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	뿌듯뿌듯 하네요!!	꼬옥 실제로 한번 뵈어요 뿌뿌뿌~!~!	0.0	0.0
boostcamp-sts-v1-train-005	nsmc-rtt	오마이가뜨지저스크라이스트퐛	오 마이 갓 지저스 스크론 이스트 뤤	2.6	1.0
boostcamp-sts-v1-train-006	slack-rtt	전 암만 찍어도 까만 하늘.. ㅠㅠ	암만 찍어도 하늘은 까맣다.. ㅠㅠ	3.6	1.0
boostcamp-sts-v1-train-007	nsmc-sampled	아렇게 귀여운 취들은 처음이네요.***	아렇게 지겨운 공포영화는 처음..	0.6	0.0

id	문장 고유 ID, 데이터의 이름과 버전, train / dev / test 기입.
source	문장의 출처, petition(국민청원), NSMC(네이버 영화), slack(Upstage slack).
sentence_1	문장쌍의 첫 번째 문장.
sentence_2	문장쌍의 두 번째 문장.
label	문장쌍의 유사도, 0에서 5 사이의 소수 첫째 자리까지의 값.
binary-label	label이 2 이하인 경우 0, 아닌 경우 1로 변환된 값.

label의 점수 기준은 다음과 같다.

5점	두 문장의 핵심 내용이 동일하며, 부가적인 내용들도 동일함.
4점	두 문장의 핵심 내용이 동등하며, 부가적인 내용에서는 미미한 차이가 있음.
3점	두 문장의 핵심 내용은 대략적으로 동등하지만, 부가적인 내용에 무시하기 어려운 차이가 있음.
2점	두 문장의 핵심 내용은 동등하지 않지만, 몇 가지 부가적인 내용을 공유함.
1점	두 문장의 핵심 내용은 동등하지 않지만, 비슷한 주제를 다룸.
0점	두 문장의 핵심 내용이 동등하지 않고, 부가적인 내용에서도 공통점이 없음.

- Total Data Numbers : 10,974개, Training Data : 9,324개, Dev Data : 550개, Test Data : 1,100개

2. 프로젝트팀 구성 및 역할

곽민석 : 모델 리서치, 프로젝트 구조 세분화, 파라미터 튜닝 및 구조 개선

이인균 : EDA, 전처리, 모델 실험

임하림 : 모델 리서치, Cross validation(KFold)을 통한 모델 검증, 모델 실험

최휘민 : 모델 리서치, 모델 실험, 모델 평가, 모델 앙상블

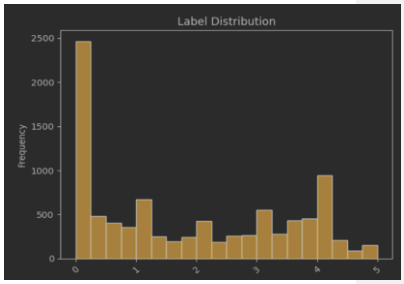
황윤기 : 모델 리서치 및 설계, 스케줄러 적용, 하이퍼 파라미터 튜닝, Wandb 환경 구성

3. 프로젝트 수행 절차 및 방법

	10 (월)	11 (화)	12 (수)	13 (목)	14 (금)	15 (토)
	강의 완강	강의 완강	강의 완강	EDA 브레인 스토밍	EDA 코드 통합	EDA 모델링
16 (일)	17 (월)	18 (화)	19 (수)	20 (목)		
EDA 모델링	EDA 모델링	코드 통합 모델링	파라미터 튜닝	파라미터 튜닝 모델별 앙상블		

1. 프로젝트 개발환경 구축(Github, Notion, Slack, WandB)
2. EDA를 진행하면서 데이터의 구조를 파악
3. EDA를 바탕으로 전처리, 데이터 증강 후 성능 개선 실험
4. 학습 단계에서 Cosine Annealing Warmup Restarts Scheduler, Base Linear Warmup Scheduler 도입.
5. Kfold를 활용해서 모델 일반화 성능 올리기
6. RoBERTa(Nto1), ELECTRA(Nto1), sRoBERTAa(Nto768) 등 다양한 모델을 각자 실험
7. Weights & Biases Sweep을 활용한 하이퍼 파라미터 튜닝 후 실험 반복
8. 모델별 예측치의 평균을 구하는 앙상블을 통해 최종 결과물 제출

4. 프로젝트 수행 결과



● 탐색적 분석 및 전처리 (학습데이터 소개)

- train.csv의 라벨은 0.0과 4.0 근처에서 확연한 불균형을 띄고 있고 이와 달리 dev.csv는 균등한 라벨을 갖고 있었다. 베이스라인 코드를 사용해 예측한 값과 dev.csv의 참값을 비교한 결과 라벨 2.0~3.0에서 다른 구간과 달리 유의미하게 안 좋은 성적을 보였다. 이에 대한 원인이 train.csv의 불균형으로 인한 학습의 부족이라고 판단하여 균형있게 학습할 수 있도록 모델을 개선하기로 했다.
- train.csv의 개수는 9324개로 신경망이 잘 작동하는 최소 개수인 1만 개 근처이긴 하지만 라벨이 연속적인 값임을 고려하면 충분한 데이터라고 볼 수는 없었다. 따라서 다음과 같은 다양한 데이터 증강 방법을 모색하였다. 방법 2와 3은 EDA(Easy Data Augmentation, arXiv:1901.11196) 논문을 참고하였다. 그러나 이들 방법 모두 성적이 유의미한 성적 향상이 없었다. 사전학습된 데이터의 품질보다 증강된 데이터의 품질이 나빠서 이런 현상이 발생하는 것으로 생각하여 데이터 증강은 하지 않는 것으로 결정을 했다.

  (sentence_1과 sentence_2를 서로 뒤바꾸는 것 / 문장 내의 단어의 일부를 지우는 것 / 문장 내의 단어의 일부를 순서를 바꾸는 것 / 문장을 확인해보니 중간 부분이 의미있는 듯하여 첫 단어와 마지막 단어를 잘라내는 것)
- 매우 유사한 문장임에도 띄어쓰기, 문장 부호, 오타를 이유로 베이스라인 코드가 제대로 분별하지 못하는 경우가 있었다.

● 모델 개요

- RoBERTa 모델을 한국어 데이터(KLUE[1])로 사전학습한 모델을 학습 데이터를 이용해 STS-Task에 맞게 미세조정 하였다.
- 비정제된 한국어를 포함한 데이터로 사전학습한 KcELECTRA[2]을 학습 데이터를 이용해 STS-Task에 맞게 미세조정 하였다.

---

[1] https://klue-benchmark.com/

[2] https://github.com/Beomi/KcELECTRA

- RoBERTa 모델을 한국어 데이터[3]로 사전학습 후 768차원으로 출력하는 모델(ko-sroberta-multitask)을 사용하는 것을 시도하였다.

● 모델 선정 및 분석

- RoBERTa 모델은 STS-Task에 좋은 성능을 내는 것이 알려져있고[4], 영어로 사전학습된 모델보다는 한국어로 사전학습된 모델이 좋은 성능을 낼 것으로 예상하여 해당 모델을 사용하였다.

- KcELECTRA 모델은 기존 한국어 사전학습 모델이 잘 정제되어 있는 데이터로 학습이 된 것을 보완하기 위해, 한국 사이트의 댓글과 같은 신조어나 오탈자 등이 포함된 정제되어 있지 않은 데이터를 기반으로 사전학습을 진행한 모델이다. 우리의 학습 데이터 또한 EDA결과 비정제 되어있는 데이터의 분포가 높은 것으로 판단하여, 해당 모델이 좋은 성능을 낼 것으로 예상하였다.

- ko-sroberta-multitask 모델은 768차원으로 인코딩하기 때문에 타 모델과는 다른 특징을 잡아낼 것이라 기대하였다. 그러나 그라디언트가 출력되지 않아서 미세조정이 불가능해서 선정하지 않았다.

● 모델 평가 및 개선

Column1	Column2	Column3	kcelectra1	kcelectra2	roberta1	roberta2	roberta3
source	sentence_1	sentence_2	kcelectra1(87	kcelectra2	roberta1	roberta2(91	roberta3(90
slack-sampled	앜ㅋㅋ두분다 넘나 귀욥..	두분 너무 귀여워요..	4.4	4.2	1.5	1.9	0.9
slack-rtt	소파 위에서 방방 ㅋㅋ	소파 위의 방 ㅋㅋㅋㅋ	4.9	4.8	2.2	2.4	1.7
nsmc-sampled	기여워 죽는줄ㅋㅋㅋ	무서워 죽는줄 알았네.	2.3	1.0	3.5	2.8	3.8
petition-sampled	나경원을 파면시켜주세요	나경원의원 파면시켜주세요	4.1	4.2	1.9	2.9	2.7
petition-sampled	그린밸트해제해주세요	농지 그린벨트 해제하라.	3.1	3.6	1.4	3.2	1.5
nsmc-sampled	완존잼있었는데 끝나버리다니..ㅠㅠ	재미있었고 신선했음....	1.9	2.2	0.4	1.3	0.2
slack-rtt	여기서 하시면 스타되심요. ㅋㅋㅋ	여기서 하면 별이 됩니다. 하하하	2.9	3.6	2.2	3.1	2.2
petition-rtt	소방관 채용인원 추가	소방관 추가	3.9	2.3	3.6	3.2	3.9
slack-rtt	이렇게 지낼 수도 있다고 합니다.	이렇게 살 수도 있습니다.	3.4	2.4	3.7	3.3	2.8
slack-rtt	아침 일찍이라 그런거 아닐까요? ㅋㅋㅋ	이른 아침이라 그럴까요? 하하하	4.2	4.2	2.9	3.4	2.6
slack-sampled	생성 완료했슙니다!	생성 및 초대 완료~!	2.7	3.2	2.0	2.5	1.3
slack-rtt	네 어제 정말 즐거운 시간있습니다.	네 어제 정말 잘 보냈습니다.	2.1	1.9	3.1	2.4	2
nsmc-rtt	이거 존재하였나????	이런게 있었어????	1.5	3.7	2.5	3.6	2.1
petition-sampled	가상화폐거래소폐쇠반대청원합니다	가상화폐 거래소 폐쇠 청원합니다	3.6	1.7	2.9	2.5	3.7
slack-sampled	다음에 기회 되면 또 떠들어요!	다음에 기회되면 또 식사해요~ ㅋ	2.5	1.8	2.9	2.4	1.9
nsmc-sampled	아무리 봐도 절대 안질리는 영화-음악	기준은 아무리 봐도 안질리긔	2.4	2.5	1.4	2.9	1.3

- 학습된 여러개의 RoBERTa 모델과 KcELECTRA 모델의 테스트데이터 결과물을 비교해 봤을 때, 문장에 오타 또는 신조어가 포함된 경우 RoBERTa 모델들이 예측한 결과 값들이 종종 부정확해 보이거나 편차가 큰 것으로 나타났다. 반면에, KcELECTRA 모델은 같은 문장에 관한 결과값들이 상대적으로 균일하고 좀더 정확해 보이는 것으로 나타났다. 위의 이미지처럼 같은 의미인 '귀욥'과 '귀여워요'에 대하여 RoBERTa 모델은 낮은 점수와 점수들의 편차가 큰것을 확인할 수 있다. KcELECTRA 모델이 처음에 학습할 때 정제되지 않은 단어를 포함하여 학습되었기 때문에 상대적으로 오타 또는 신조어에 강한 것이라고 생각된다. 이에 따라, RoBERTa 모델과 KcELECTRA 모델의 적절한 앙상블 전략을 생각해 보았고 RoBERTa

[3] https://github.com/jhgan00/ko-sentence-transformers

[4] https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark

모델로 학습한 여러 모델의 결과 값들의 편차가 큰 결과물에 대해서는 KcELECTRA 모델의 가중치를 높게하여 앙상블을 진행해 보았다. 동일한 조건으로 두 모델을 가중평균 방식으로 앙상블한 결과물의 점수가 0.9229이고, RoBERTa 모델들의 결과 편차가 컸던 상위 10% 만 KcELECTRA와의 가중치를 7:3로 주었던 결과물의 점수가 0.9234으로 성적 향상을 이뤄냈다. 하지만 매우 미세한 차이였기 때문에 최종평가 제출에는 편차에 따른 가중치를 주었던 결과물과 편차에 따른 가중치 없이 가중평균만 했던 두개의 결과물로 제출하였다.

● **일반화 성능 올리기**

- 데이터 자체를 바꾸는 것은 에폭이 돌더라도 동일한 데이터를 학습하기 때문에 좋지 않은 구현이라 생각했고, WeightedRandomSampler를 통해 데이터로더 차원에서 에폭마다 다른 데이터를 균형있게 학습하도록 했다. 이를 통해 0.852->0.865의 성적 향상을 이뤄냈고 test.csv도 dev.csv와 마찬가지로 균등한 라벨을 갖고 있으리라 판단해서 일반화 성능에도 문제가 없다고 판단하였다.

- Sentence1과 Sentence2가 의미적으로는 매우 유사한 문장이지만, 형식이 맞지않아 좋은 성적을 내지 못하는 문제점을 해결하기 위해 Py-Hanspell[5]을 활용해서 맞춤법 교정을 진행하였고 0.8719->0.8755로 성적 향상을 이뤄냈다. 향상 폭은 적었지만 논리적으로 올바른 접근이라 생각해서 코드에 적용하였다.

● **시연 결과**

[5] https://github.com/ssut/py-hanspell

5. 자체 평가 의견

- **잘한 점들**
  - 절반씩 인원을 나눠서 RoBERTa, ELECTRA계열 모델을 따로 맡았고 분업이 잘 이루어졌다.
  - 코드를 이해하는 시간과 아이디어를 내는 시간을 충분히 부여해서 시작은 늦었을지 몰라도 팀의 방향이 바뀌거나 멈추는 일은 없었다.
  - 병렬적으로 여러 시도를 시행하며 모든 인원이 프로젝트의 처음부터 끝까지를 경험하였다.
  - 처음 해보는 협업임에도 불구하고 과정에서 큰 문제가 없었다. 시도해본 것에 대한 결과물 공유가 활발하였다. 또한 좋은 결과를 낸 것에 대해선, 메인스트림으로 병합했다.
- **시도했으나 잘 되지 않았던 것들과 개선사항**
  - 브레인 스토밍을 시도했으나 시간을 정하지 않고 개인에게 맡기다 보니 분업이 효율적으로 이루어지지 않았다.
    → 각자 공부가 끝나면 브레인 스토밍 시간을 진행하고 후에 생각들을 정리하는 시간을 가지기.
  - 모델 재설계 시도해봤으나 큰 성능향상은 없었다.
    → 모델 재설계의 성능향상에 대해서 조사해보고 정리하기
- **아쉬웠던 점들과 개선사항**
  - 깃허브 기능 활용(Issue, PR 등)을 많이 하지 않았고 협업이 낯설다 보니 5명이 모두 다른 코드로 작업을 하게 되었고 작업물 공유가 어려웠다.
    → Issue, PR, Commit 메세지 Template를 설계하여 팀원간의 전달할 내용 통일을 진행.
  - 현재 프로젝트 진척 상황을 표시하는 곳이 없어서 진척도를 파악하기 어려웠다.
    → Trello, Jira 같은 프로젝트 관리 툴을 사용 혹은 Notion과 같은 공유 문서를 이용해 설계
  - 모델의 세부 조작을 통한 성능 개선을 하지 않고, 파인 튜닝과 데이터 수정만으로 성능향상을 시도한 것이 아쉬웠다.
    → 세부 Layer의 조작을 통해, 모델 성능 개선방향에 대해서도 토의.
  - 개발 중 코드 내부에 문서화를 자세히 진행하지 않아서, 서로 다른 코드를 작성하여 다른 팀원들이 어떤 파라미터가 무엇을 의미하는지 이해하는데 시간이 걸렸다.

    → 문서화 기준과 코드 스타일을 함께 정의하여 다음 프로젝트때 사용할 계획.
  - 토의했던 것들을 자세하게 기록하지않은 것.
    → 한 사람이 서기 역할을 맡아 시간을 정하고 그 시간 안에서만 하고 끝나고 아이디어 정리하기.
- **프로젝트를 통해 배운 점 또는 시사점**
  - 모든 회의에서는 생각의 흐름을 적어두는 과정이 필요하다.

○ 어느 방향으로 진행하고 있는지 공유하는 문서를 작성하는 시간과 공간이 필요하다.

○ 기능별로 모듈화를 하는 것이 재사용할 때에 더 편리하다는 것을 깨달았다.

○ 성능 향상에 있어서 많은 가능성을 열어두고 모델 변경 또는 데이터의 조작을 진행해야 한다.

개인 회고

곽민석

1. 성능 향상을 위한 노력

- 이번 프로젝트는 NLP에 관련된 첫 프로젝트로 wandb, Pytorch Lightning에 익숙해지고 이를 활용해보는 것이였습니다.
- 데이터의 특성을 알아보기 위해 streamlit을 이용하여 사용자의 interaction을 통해 데이터 특성을 알아볼수 있게 만들어보았습니다.
- wandb의 sweep 기능을 통해 Learning-rate의 warmup step과 batch 크기 그리고 k-fold의 k 값 등 hyper-params의 적정점을 구해보았습니다.
- klue/roberta-small 외에 다른 모델을 사용하여 성능을 측정하고 사용 해 보았습니다.
    - koGPT, kcElcetra 등을 사용해보았으며, roberta-large가 가장 이번 프로젝트에 적합한 모델임을 알아냈습니다.
- 자신이 학습시킨 모델들 중 결과가 좋은 모델을 사용하여 각 모델의 결과에 대해 가중치를 주어 성능을 올려보았습니다.
- 전체 train dataset에서 80%의 Sentence 쌍의 순서를 바꾸어 데이터를 증강시켜 성능 향상에 도움이 되었습니다.
    - 이를 통해 val_pearson 기준, 0.81에서 0.9까지 성능을 향상시켰습니다.
- "Attention is all you need"를 읽고 learning schedule을 알게되어, LinearLR을 이용해 learning schedule을 적용해봤습니다.
- Inference 부분과 training 부분을 통합하여 팀원들에게 배포하였습니다.
- token 길이를 조절하여 큰 batch 크기에서도 학습이 가능하게 하였습니다.

2. 한계점과 아쉬운 점 그리고 해결 방안

- Hyper-parameters를 sweep을 이용해 찾을 때 너무 마구잡이 식으로 찾아본것 같습니다.
    - 이에 대해 동일한 task에 대해 선행된 연구에서 사용한 방식을 먼저 찾아보고 접근하는 방법으로 진행하는것이 좋을 것 같습니다.
- 모델의 hyper-parameter의 성능 확인을 하나의 고정된 random seed 값에서 진행하였습니다.
    - 다음 실험에서는 seed 값도 sweep에 포함하여 해당 parameter가 얼마나 강건한지 측정해보려 합니다.

- 모델을 만들어 나가면서 기능 추가가 어떤것이 되었는지 검수를 모두 하지 못한채로 진행해 어디서부터 성능향상이 된것인지 일정 부분 이후로 추적하지 못했습니다.
    - git을 조금 더 적극적으로 사용하여 추적이 가능하게 하고 sweep에서 각 기능의 On/Off가 가능하도록 구성하려합니다.
- 각자의 결과물을 취합할 때 git을 잘 활용하지 못한것 같아 조금 아쉽습니다.
    - 다음 프로젝트에는 conflict가 발생하지 않도록 모듈화를 하여 작업을 진행하려 합니다.

3. 다음 프로젝트에서...

아직 구체적인 계획은 없지만, 팀원들 모두 이번에 사용한 sweep 부분 등 다음 프로젝트에서 사용가능한 모듈에 대해 따로 분리하여 관리해보자는 의견이 나왔습니다. 이번 프로젝트를 기획하며 만든 것을 재활용함으로 인한 시간절감이 이유였는데, 나 또한 이점에 대해 강하게 동의합니다.

또한 위의 한계점과 해결방안 이외에도 선행된 task에 대한 기본 조사를 통해 관련 지식을 먼저 습득하고 프로젝트에 참여해보려 합니다.

이인균

- 나는 내 학습 목표를 달성하기 위해 무엇을 어떻게 했는가?
    - 우리 팀과 저의 초기 목표는 좋은 성적을 받도록 노력하되, 그 과정에서 인터넷에서 찾게 된 코드를 그대로 사용하지 않고 원리와 구현을 최대한 이해하는 것이었습니다.
    - 저는 EDA, EDA에 따른 전처리, 데이터 증강 부문에 집중했습니다. EDA 결과 label 피처에 대해서는 불균형이 있다는 점을 알게 되었습니다. WeightedRandomSampler를 통해 데이터를 균형있게 학습하도록 했고 0.852->0.865의 성적 향상을 이뤄냈습니다.
    - 다양한 데이터의 오류를 고치기 위해 띄어쓰기, 맞춤법 수정, 별도의 단어 토큰 추가 등의 전처리를 했습니다. 다양한 데이터 증강 방법을 시도했습니다.
    - kcelectra 모델에 LinearLR 스케줄러를 적용시켰고 하이퍼 파라미터 튜닝을 했습니다. 기존 CosineAnnealingLR을 쓴 모델에 비해 0.92->0.93으로 성능이 향상되었고 최종 제출에 활용되었습니다.
- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?
    - 총 4가지의 데이터 증강법을 시도하였으나 유의미한 성능 향상은 없었습니다. 그러나 자연어에 대한 데이터 증강 방법을 익혔습니다.
    - Easy Data Augmentation 논문 속 유의어로 교체하는 것 / 유의어를 삽입하는 것의 구현 방법을 이해하지 못해서 사용을 하지 않았던 점이 아쉬움이 듭니다.
    - 단어 '자브라'와 'zabra'의 차이로 인해 예측이 크게 엇나갔던 문장이 있었고 이것이 전처리의 한계라고 생각했지만 hangulize 라이브러리의 존재를 프로젝트 피드백 시간에 알게 되었습니다. 더 정보를 찾아보지 않았던 것에 아쉬움이 듭니다.
    - 모델의 개선(특히 문장1과 2의 교환)이 오차 범위 이내인지 파악하는 데에 어려움이 있었습니다. 피드백 시간에 10번 가량 실행하여 오차범위를 파악하는 것이 정석이라는 것을 알게 되었습니다.
    - 768차원으로 출력하는 ko-sroberta-multitask 모델을 쓴 뒤 코사인 유사도를 구하는 예측을 시도하였습니다. 그러나 그라디언트가 출력되지 않아서 파인튜닝이 불가능하였습니다. 시간이 없어서 더 이상 시도하지 못했던 점이 아쉬움이 듭니다.
- 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?
    - Easy Data Augmentation 방법론이 아닌 다른 데이터 증강을 시도해보고 싶습니다.
    - 모델링 이전 단계에 집중했는데 다음에는 모델링 단계에 힘을 써보고 싶습니다.

임하림

메모 포함[1]: @limchickennn@gmail.com
이마리미님에게 할당되었습니다.

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

따라가기 위해 모르는 것은 다 받아 적었다.
팀원과 회의를 할 때 알아듣지 못 하는 용어나 기법들이 굉장히 많았다. 중요한 개념이면 바로 물어보았지만 흐름을 깨지 않기 위해 모르는 다른 단어들은 적어놓고 하루 일과가 끝나면 모든 단어들을 알아보고 기록하는 시간을 가졌다. 점점 모르는 단어가 적어지고 프로젝트를 따라가는 것에 있어서 훨씬 수월했다. 추가적으로 팀원과 나의 차이가 있어서 따라갈 수밖에 없는 포지션일때, 나만의 진도를 뒤에서 가져가는 것보다 진도는 맞춰가고 따로 시간을 내서 뒤에서부터 채워 나가야 한다는 것을 알았다.

Pytorch Lightning을 능숙하게 사용하기
Pytorch도 능숙하지 않아서 작동 방식을 정확히 이해하기 위해 기존의 pytorch코드와 비교해가며 주석을 달아가며 이해했습니다.

WandB와 sweep을 통한 hyper-parameter tuning
boostcamp 내에 있는 예시와 같이 WandB tutorial에 있는 코드도 참고해서 작성하는 방법과 결과를 확인하는 방법을 이해했습니다

2. 나는 어떤 방식으로 모델을 개선했는가?

많은 모델들의 성능을 확인하고 최적의 성능을 내는 hyper-parameter를 확인했습니다.
같은 계열의 모델이라도 적절한 batchsize, learning rate 등이 각각 다른 것을 확인하고, hugging face에서 프로젝트에 적합한 모델을 찾기 위해 적절한 hyper-parameter를 sweep으로 돌려 점진적으로 성능을 올렸습니다.

KFold 방식의 cross validation을 활용하여 모델을 검증했습니다.
data의 수가 작다고 판단해서 train data와 test data를 합쳐서 KFold를 5로 설정하여 generalization 성능을 올렸습니다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

모든 프로젝트는 문제를 최대한 자세히 정의하고 이해한 다음 코드를 시간을 가지고 찬찬히 뜯어보고 이해하는 것이 더 중요하다는 것을 깨달았다.
팀에서 유일하게 Kaggle을 안 해봐서 익숙한 팀원들을 따라가기 위해 코드를 자세하게 뜯어보지 않아, 참고해서 코드를 가져오는 것까지는 할 수 있었지만 그 코드를 수정하고 발전시키는 것에는 확실하게 한계가 있었다. 그것이 오히려 내가 팀의 성적에 도움을 줄 수 없는 요인이 되었고 시간을 추가적으로 내서 확실하게 이해하고 난 뒤에 팀 성적에 기여할 수 있었다. 단순히 붙여넣기만 하는 작업이 아니라면, 전체적인 구조를 이해했다면 훑어보지 말고 이해 안 되는 거 없이 찬찬히 뜯어봐야 한다는 것, 그리고 정석으로 하는 것이 가장 빠른 길임을 알았다.

문제가 발생했을 때, 문제 자체를 어디까지 이해했는가를 먼저 확인하고 큰 문제에서 작은 문제로 찾아가야 한다.
KFold를 적용할 때, val data에 대해서는 성능이 굉장히 좋게 나왔지만 정작 inference하면 점수가 작게 나오는 문제가 있었다. 처음에는 train data와 val data가 작아서 그렇구나 생각하고 넘어갔지만 대회 막바지에 코드를 자세히 보니 KfoldDataloader에서 train data와 val data를 합쳤는데 test pearson을 뽑는 코드가 test data가 아니라 val data를 기준으로 산출하고 있었다. 즉, KFold 함수 작동방식와 데이터를 나누는 이유를 확실히 이해하지 못 해서 data를 제대로 분리시키지 않았고, 코드도 확인을 정확히 못 해서 inference에 비해 test pearson이 비정상적으로 점수가 잘 나왔던 건데 확실히 하고 넘어가지 않아 많은

시간을 소비했다. 문제점이 있으면 반드시 밑단의 밑단을 면밀히 해결하고 그 이유를 확실히 확인해야하는 것을 알았다.

4. 마주한 한계와 아쉬운점, 그리고 그것을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

Problem: 생각과 논리의 흐름을 작성하지 않았다.
하나의 결과를 도출하기 위해서는 많은 생각과 논리의 흐름이 필요하다. 그 결과가 작동하지 않는다면 그 흐름을 보고 각각의 흐름이 타당한지 확인해야 하는데, 그 과정을 기록하지 않아서 확인할 수가 없었다.
Try: 모든 나의 생각과 논리의 흐름을 '왜'라는 근거에 집중해서 작성하고, 결과를 확인한 후에도 성공과 실패를 작성하는데 이것도 '왜'라는 근거로 작성하자

Problem: 코드를 공유할 때 왜 라는 질문이 너무 많이 생성되었다.
상대방이 작성한 코드를 보고 다른 팀원들이 나의 코드를 볼 때, 그 기능과 작동 방식은 검색을 통해 알 수 있지만 '왜'라는 질문에 대해 알 수 없는 코드가 있는 것 같다. 변수로 지정할 수 없는 단순히 더해주는 숫자이거나, 혹은 변수 name은 잘 설정했지만 그 변수 값이 의미하는 바가 중요한 경우는 더더욱 그런 것 같다. 서로에게 질문이 들기전에 반드시 왜 라는 질문이 나오지 않게 고민을 하고 팀원에게 공유하는 것이 좋겠다는 생각을 했다.
Try: 변수의 이름이나 모델명으로 알 수 없는 단순한 상수는 반드시 팀 공유 페이지에 작성하거나 코드에 주석을 작성하자.

최휘민

메모 포함[2]: @hmc123123@gmail.com
최휘민님에게 할당되었습니다.

○ 우리 팀과 나의 학습목표는 무엇이었나?

- 저와 우리 팀은 좋은 학습성적을 달성하기 위해, 프로젝트를 어떻게 효율적으로 진행해야 할지 고민했습니다. 하지만 우리 팀은 NLP 대회 관련 경험이 없었기 때문에 역할 지정없이 생각나는 아이디어를 공유하고 각자 하고 있는 것을 공유해 겹치지 않게 하는 것으로 병렬적으로 진행해보려고 하였습니다.

- 저는 이전에 Kaggle을 가볍게 경험하면서 전처리와 단일모델 설계를 해보았기 때문에 이번에는 복수의 모델 비교와 적용 그리고 모델 평가로 진행하였습니다.

○ 나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 주어진 학습 데이터와 테스트 데이터를 이해하고, 한글 데이터의 문장 유사도 측정에 어울리는 모델의 목록을 살펴보았습니다. 주어진 데이터는 신조어나 오타가 많이 포함되어 있었고 KcELECTRA 모델이 기존의 한국어 transformer 모델들과는 달리, 네이버 댓글과 같은 정제되지 않은 단어를 포함하여 사전 학습을 진행하였기 때문에 우리의 한글 데이터의 문장 유사도 측정에 적합한 모델이라고 판단하고 팀원들에게 주어진 데이터의 특징과 한국어 transformer 모델들의 특징과 성능을 비교하는 정보를 공유했습니다.

- KcELECTRA를 포함하여 KoBERT, KOELECTRA 등 여러 모델들을 적용해보고 WandB의 Sweep을 통해 Hyper-Parameter 튜닝을 진행하고 결과물을 확인해 보았습니다. 기존 RoBERTa 모델과 차이가 없거나 성능이 떨어지는 모델을 버리고 진행하였습니다.

- 예전 경험을 살려 학습과 테스트 과정에서 drop해서 버리던 'source' 데이터를 살려서 클래스 데이터처럼 추가해 주자는 의견을 전처리 단계로 제시해 봤고 결과는 검증 데이터의 점수는 올랐지만 제출 점수는 크게 차이가 없거나 떨어진 결과였습니다. 학습은 잘됐지만 테스트 데이터는 'source'의 분포를 따라가지 않았다고 결론을 내리고 사용하지는 않았습니다.

- KcELECTRA와 RoBERTa 모델의 결과 값들을 비교해 어떤 입력 데이터 형태가 모델에게 어려운지, 또는 쉬운지, 어떤 입력 데이터가 모델에게 잘못된 예측을 유발하는지 파악해 모델과 데이터의 특성을 살피는 모델 평가를 진행하였고 이를 바탕으로 앙상블 전략과 데이터 전처리를 생각했습니다.

○ 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 팀원간 역할을 제대로 정하고 진행한 것이 아니었고 같은 기능을 구현하는 것만 피했기 때문에 제가 본격적으로 모델 평가를 하기 시작했을 때부터는 다른 팀원들이 하지 않았던 전처리방식 도입이나 모델 개선 방법을 대회 마감까지 여유가 없어서 아이디어는 있어도 시도해보지 못한 것이 너무 아쉬웠습니다.

○ 한계/교훈을 바탕으로 다음 프로젝트에서 시도해보고 싶은 점은 무엇인가?

- 팀원간 역할을 확실히 나누어 시간과 업무를 효율적으로 분류해 보고 최대한 많은 아이디어를 시도해 보고 또한 유용한 협업툴을 도입해 사용해보고 싶다고 생각합니다.

- 팀원과 협업을 위한 파이썬 프로젝트 템플릿을 도입하고 주석을 잘 달고 코드를 정리해보고 싶습니다.

황윤기

○ **나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?**

- Kaggle Competition의 상위권 모델들에 대해서 많이 탐색하고, 우리 모델에 적용하도록 노력하였다. 상위권 모델들은 어떤 방법을 사용하여 모델을 개선하고, 성능 향상을 기대했으며, 우리 모델에 적용하여도 효과가 있을지에 대해서 고민해보는 시간을 많이 가졌다.
- 베이스라인 코드에 사용된 모델말고도 우리 Task에 효과적인 사전학습 모델은 무엇이 있을까에 대해서 찾아보고, 실제로 팀원들이 이용할 수 있도록, 최적화된 하이퍼파라미터를 찾기 위해 Wandb를 이용하여 여러 실험을 통해 개선하였다.
- 효과적인 데이터 전처리에 대해서 고민하였으며, 도입해보았다. EDA를 통해 데이터가 비정형적으로 되어있는 것을 확인한 뒤에, 띄어쓰기 사용 또는 맞춤법 교정을 통해 전처리를 진행하여 팀원들과 데이터셋을 공유하였다.

○ **나는 어떤 방식으로 모델을 개선했는가?**

- Kaggle Competition 상위권 모델과 여러 방법론을 찾아본 결과 새로운 Learning Rate Scheduler의 도입을 통해 모델을 개선하였다. 많은 효과적인 모델들이 Cosine Annealing Warmup Restarts Scheduler를 사용하여 모델의 성능 향상을 기대함에 따라 우리 모델에도 적용하려고 했지만, PyTorch Scheduler에서 제공하는 해당 Scheduler는 Step이 진행함에 따라서, Warmup시에 최대로 올라가는 Learning Rate를 주기마다 반복해서 사용한다는 문제점이 있다는 것을 알았다. 그래서 여러 글들을 참고하여, 학습의 후기에는 Learning Rate를 더 세세하게 조절해줘야할 것으로 생각해, 매 주기마다 최대로 올라가는 Learning Rate를 감소시켜주는 비율(gamma)값을 도입한 Custom Scheduler를 작성하고, 우리 모델에 적용하도록 모듈을 공유하였다.
- 우리가 가지고 있는 데이터셋은 비정형의 한글로 작성된 글이 많기에, 데이터의 특성(비정형 + 한글)에 맞춰서 사전학습을 진행한 모델을 찾도록 노력하였다. 또한 앙상블을 위해, 베이스라인 코드에서 제공된 RoBERTa계열 모델이 아닌, 다른 계열의 모델을 기준으로 두고 탐색하였다. 그로 인해 찾은 새롭게 찾은 모델이 KcELECTRA였다. 해당 모델은 Korean Comment로 ELECTRA를 사전학습을 진행한 모델이었기에, 내가 찾고있는 기준과 부합하였다. 하지만 해당 모델을 미세조정하며 학습할 때 Loss값이 생각과 다르게 수렴하지 않는 문제가 생겨서, 팀원들에게 가장 적합한 기준점이 되는 하이퍼파라미터를 찾기 위해서 여러번의 실험을 통해서 Loss값이 수렴하는 하이퍼파라미터를 찾았다.

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

- 여러 모델의 실험을 통해, 수렴하는 Loss값을 갖는 모델의 예측결과를 표시해보고, 같은 Training Data로 학습하고 비슷한 성능 지표를 가져도 결과가 다르게 나온다는 것을 알았다. 아마 이건 모델마다 중점적으로 보는 부분이 달라서 일어날 수 있는 문제점이라고 생각할 수 있을 것 같다. 그렇기에 모델 앙상블을 통해서 성능 지표의 높은 향상을 기대할 수 있을 것이라는 생각을 하였다.

○ **요약**

이번 대회를 진행하면서 나는 데이터의 조작과 다양한 모델들의 실험과 Scheduler 개선의 중점을 두고 성능향상을 진행하였다. 하지만 스페셜 피어세션에서 만난 다른 조들의 얘기를 듣고, 다음 대회에서는 모델의 세부적인 Layer 조작을 통해 성능향상을 기대해보아도 좋을 것 같다라는 생각이 들었다. 또한 Data증강 및 조작은 무조건적인 성능향상을 가져올 것이라 생각했지만, 어느 한계점을 넘어가면 들인 시간에 비해 기대할만한 성능향상을 가져오지는 않는다라는 것을 깨달았다. 그렇기에 Data 증강에 너무 큰 시간을 쏟을 필요도 없다는 것을 깨달았다. 다음 대회부터는 이번 대회를 바탕으로 모델을 개선하기 위한 진행절차에 대해서 감을 잡았기에, 더 좋은 결과를 만들어낼 수 있을 것을 기대한다.