STS Project Report

태그 report

DEMO

GitHub : https://github.com/snaiws/NLP_project

Deploy URL : Model API

Team & Role

- 송민혜
- 엄정호
- 이규호
- 이용욱

과제 목표

- 한국어 문장의 유사도 분석 모델 훈련 및 BEST 모델 API 구현
- 두 개의 한국어 분장을 입력 받아 두 문장의 의미적 유사도를 출력하는 사용자 interface 구현

Contents

1. 데이터전처리
2. 모델 선정
3. 훈련 및 평가
4. 실험
5. Model API
6. 한계 및 개선 방안
7. Reference

1. 데이터 전처리

- Data Set
  - KLUE-STS (11668 x 6)
    - AIRBNB(리뷰)
    - policy(뉴스)
    - parakQC(스마트홈 쿼리)
- train data에서 중복 제거 (11668 → 11661)
- regex를 사용하여 한글, 숫자만 남기고 문장 전처리
- train, valid data를 9:1로 나누어 실험 진행

DataSet	Size
Train	10494
Valid	1167
Test	519

2. 모델 선정

STS benchmark 에서 좋은 성능을 보였던 pre-trained model들을 비교, 실험하여 선정했습니다.

2-1. KLUE-BERT-base

reference

- 한국어로 사전 학습된 BERT 모델입니다.
- **BERT: Bidirectional Encoder Representations from Transformers**
  - Transformer 모델의 인코딩 layer에 Masked sentence 두 개를 붙여 입력하고 mask토큰과 다음 문장을 예측하는 사전학습을 한 모델입니다.

Model	Layers	Embedding Size	Hidden Size	# heads
KLUE-BERT-base	12	768	768	12

2-2. KLUE-RoBERTa

reference

- 해당 모델은 RoBERTa를 KLUE dataset을 통해 사전 학습 시킨 모델이다. 사전 학습된 모델의 크기에 따라 small, base, large로 나누어 집니다.
- **RoBERTa: A Robustly Optimized BERT Pretraining Approach**
  - BERT 모델이 underfit 되어 있다고 판단하여 여러가지 tuning을 진행한 모델입니다.
    - BERT 보다 더 많은 데이터를 긴 시간 동안 큰 batch size와 더 긴 sequence를 이용하여 학습 시킨 모델
    - Dynamic Masking으로 다양성 확보
    - NSP loss제거

Model	Layers	Embedding Size	Hidden Size	# heads
KLUE-RoBERTa-base	12	768	768	12
KLUE-RoBERTa-large	24	1024	1024	16

2-3. KoELECTRA-base-v3-discriminator

reference

- v3의 KoELECTRA는 34G의 한국어 Corpus를 이용하여 사전 학습 시킨 모델입니다.
- **fine_tuning 된 모델의 코드를 GitHub에서 발췌 하여 사용하였습니다.**
- **ELECTRA : Efficiently Learning an Encoder that Classifies Token Replacements Accurately**
  - GAN과 비슷한 형태로 generator와 discriminator가 존재합니다.
    - generator는 MLM(Masked Language Model) 로 대표적으로 BERT 등이 있습니다.
  - Replaced Token Detection(RTD) 방식으로 사전 학습을 진행하여 모든 input token에 대해 학습합니다.
    - RTD : 각 token이 generator에 의해 생성된 token인지, 원래 input token인지 판단합니다.(이진 분류)

Model	Layers	Embedding Size	Hidden Size	# heads
KoELECTRA-base-v3-discriminator	12	768	256	4

2-4. sentence RoBERTa

reference

- 코사인 유사도를 사용하여 비교할 수 있는 의미 있는 문장 임베딩을 도출하기 위해
  BERT를 개선하여 Siamese and triplet network 구조를 사용하는 모델 입니다.
- **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks**
  - sBERT를 학습하는 대표적 방법인 NLI와 STS문제를 푸는 것 중 STS문제를 풀기 위해 사용하였습니다.
    - 문장을 각각 BERT의 입력으로 넣고 mean pooling을 통해 문장 임베딩 벡터를 얻습니다.
    - 두 벡터의 코사인 유사도를 구한 후 레이블 유사도와의 평균제곱오차(MSE)를 최소화하는 방식으로 학습합니다.

2-5. 최종 모델

- 최종 모델 : KoELECTRA-base-v3-discriminator (fine-tune ver.)
- 최종 모델 채택 이유 : f1 score가 높습니다.
- 실험 결과, 모델 별 가장 높은 성능은 다음 표와 같습니다.

Mode	Pearsons' r	F1_score	Loss	Epoch	BatchSize	Optimizer	Learning Rate
KLUE-BERT-base	0.8264	0.8193	_	25	56	AdamW	3e-5
KLUE-RoBERTa-base	0.9251	0.8512	0.3351	5	58	AdamW	2e-5
KLUE-RoBERTa-large	0.7982	0.8440	_	5/25 (early stopping)	20(고정)	AdamW	2e-5
KoELECTRA-base-v3-discriminator	0.9228	0.8646	0.4084	4/20 (early stopping)	32	AdamW	5e-5
sentence-RoBERTa	0.8853					AdamW	

3. 훈련 및 평가

3-1. Train

- Regression Task
  - STS Dataset의 "real-label" 값을 이용하여 0≤ target ≤ 5 사이의 값을 예측했습니다.
  - 예측된 target값은 의미적 유사도를 의미하며, 0에 가까울 수록 유사하지 않음을 의미합니다.
  - 해당 Task에서 **평가 지표**는 **Pearsons' r coefficient**를 사용했으며,
    이는 real-label 과 target 간의 선형 상관 관계를 측정합니다.

- Binary Classification Task
  - Regression Task로 예측된 target값을 중앙값인 3.0을 기준으로 0과 1로 분류했습니다.
  - 분류된 target값은 0은 두 문장이 유사하지 않음, 1은 두 문장이 유사함을 의미합니다.
  - 해당 Task에서 **평가 지표**는 **F1 score**를 사용했으며,
    binary로 분류된 label값 과 target간의 precision과 recall에 관한 조화 평균 값을 측정합니다.

3-2. Hyper-Parameter Tuning

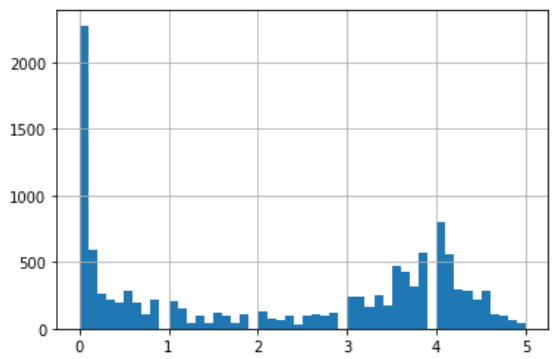
Optuna를 이용하여 최적의 batch_size와 learning_rate를 찾았습니다. 시간 관계상 loss함수와 optimizer를 고려하진 못했습니다.

- TPE sampler와 hyperband pruner를 사용했습니다.
- TPE sampler
  - 베이지안 최적화 기법에 기반한 Optuna의 디폴트 샘플러입니다.
  - 처음엔 랜덤 샘플러처럼 시작하지만 결과를 기록하고 이에 따라 다음 시도할 파라미터를 정합니다.
- hyperband pruner :
  - pruner는 epoch마다 결과를 기록합니다.
  - pruner는 알고리즘에 따라 epoch를 더 진행할 필요가 없다고 판단되면 trial의 이른 epoch 단계에서 break 하고 다음 trial로 넘어가는 기능입니다.
  - 여러 pruner 가운데 공식 문서의 추천에 따라 TPE sampler와 제일 어울린다고 하는 hyperband pruner를 사용했습니다.

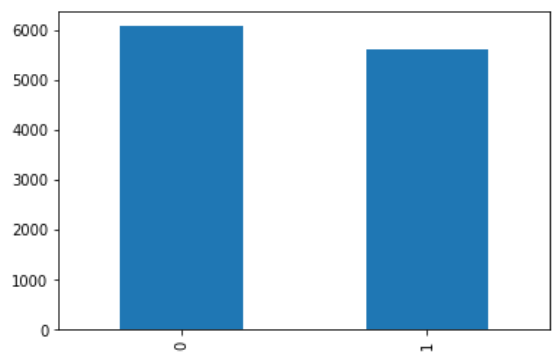
4. 실험

4-1. Data Augmentation

Data EDA 결과, Data의 분포가 0 ≤ label≤ 5의 경우, 일정하지 않았습니다.



Train Data의 real-label 분포



Train Data의 binary-label 분포

이를 보완하고 모델의 성능 향상과 과적합을 방지하고자, 데이터 증강에 대한 실험을 진행했습니다.

- EDA(Easy Data Augmentation)기법을 사용하였고, 한국어로 쓸 수 있도록 WordNet부분만 교체한 KorEDA 를 사용했습니다.
  - SR(Synonym Replacement) : 불용어가 아닌 n개의 단어를 랜덤하게 뽑고 임의로 선택한 동의어와 바꾼다.
  - RI(Random Insertion) : 문장 안에 불용어가 아닌 임의의 단어를 선택해 해당 단어의 임의의 동의어를 랜덤한 위치에 삽입한다.(n번 반복)
  - RS(Random Swap) : 문장에서 임의의 두 단어를 선택하고 자리를 바꾼다.(n번 반복)
  - RD(Random Delection) : 문장에서 임의의 단어를 p의 확률로 삭제
- 한계 : WordNet만을 단순히 바꿔서 결과를 내기 때문에 의미가 변형되는 경우가 생김 → 안전하게 데이터를 증강 하기 위해 RD, RS만 사용
  - WordNet으로 KAIST에서 만든 Korean WordNet(KWN)을 사용하였는데 국립국어원(모두의 말뭉치)에서 제공하는 NIKLex를 사용했다면

    KWN의 동의어개수(9714개)보다 훨씬 많은 60000개의 동의어를 가지고있어 더 다양한 증강법을 시도 할 수 있었을 것 같습니다.

	원본 데이터	증강 데이터
Train	11668	62995
Test	519	6353

4-2. Data Collect

프로젝트에 사용된 데이터 셋은 약 1만 개의 문장 쌍 입니다.

다른 STS Task가 30k 이상의 데이터 셋을 이용하기 때문에 이에 비해 확연히 적은 양입니다.

이를 보완하고자, 문장 쌍의 데이터를 추가하여 모델의 성능을 향상 시키고자 실험을 진행했습니다.

- KorSTS

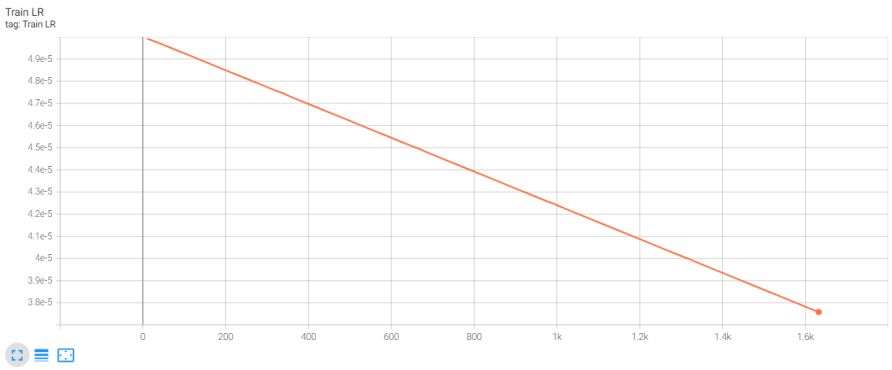
Dataset Overview

KorSTS	Total	Train	Dev.	Test
Source	-	STS-B	STS-B	STS-B
Translated by	-	Machine	Human	Human
# Examples	8,628	5,749	1,500	1,379
Avg. # words	7.7	7.5	8.7	7.6

4-3. Visualization

모델의 학습을 Tracking 하고자 Visualization을 시도했습니다.

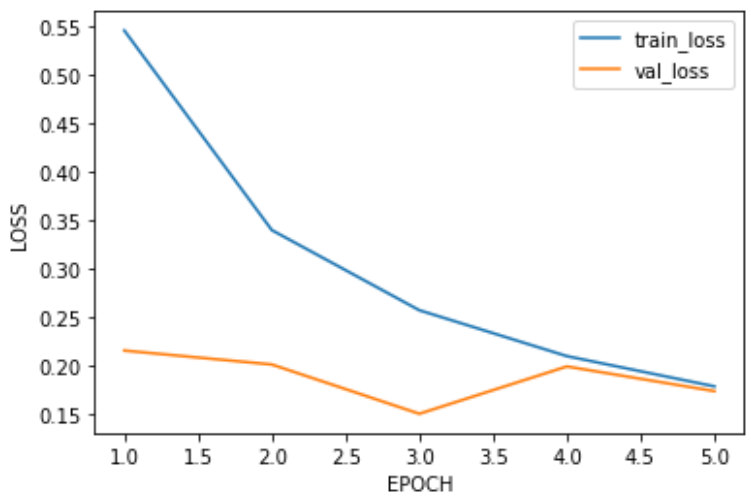
- Tensorboard를 이용하여 모델 학습 중에 log를 남기고, 동적 시각화를 구현했습니다.



KoELECTRA 적용 예시 Learning Rate

KoELECTRA 적용 예시 Train Loss

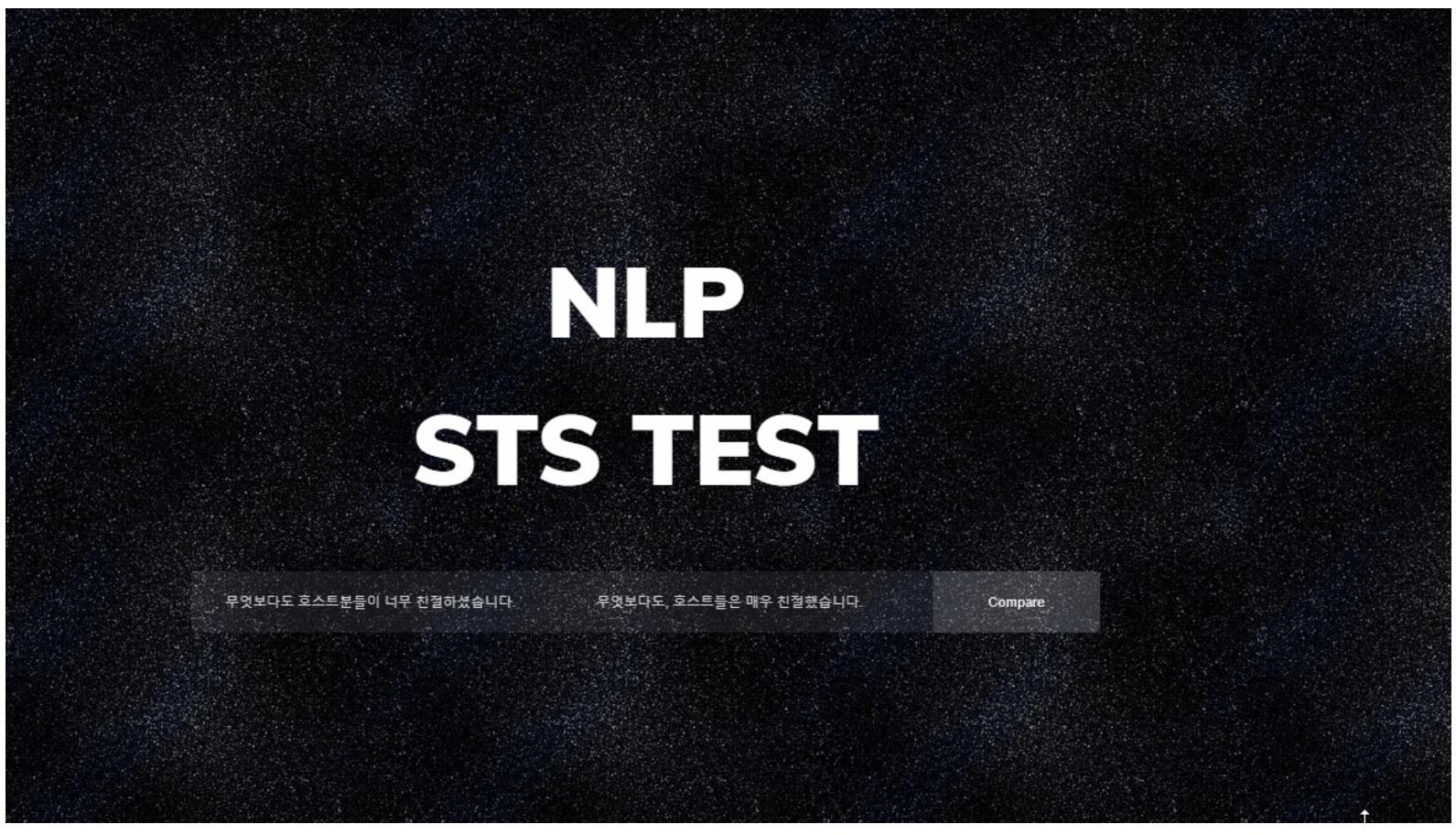
- 학습을 완료한 후, Validation Loss와 Train Loss를 비교하여 좀 더 직관적으로 확인할 수 있도록 시각화 했습니다.



- 커스텀 모델 아키텍쳐를 시각화하기 위해 torchviz와 hidden layer와 bertviz 라이브러리를 사용했습니다.

5. Model API

Flask를 사용하여 구현했습니다.



6. 한계 및 개선 방안

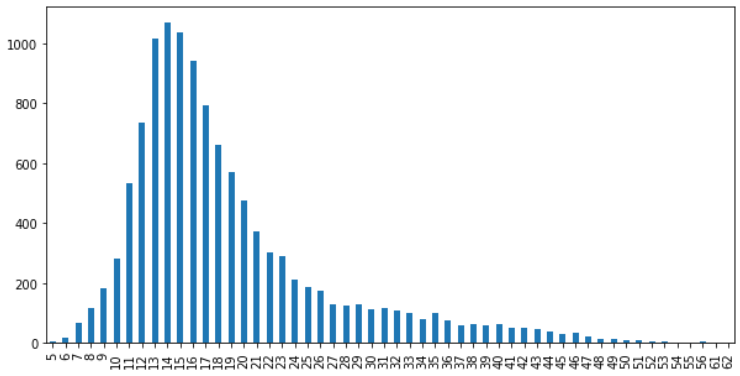
6-1. 한계

- GPU 용량의 한계로 인해 대용량 모델을 돌리지 못했던 점
- loss 함수와 optimizer를 비교해보지 못한 점

- output layer를 여러 방식으로 시도해보지 못한 점
- 데이터 증강을 적용해보지 못한 점
- 추가 데이터를 활용하지 못한 점
- 모델 아키텍쳐를 코드로 시각화 하지 못한 점
- 추가적인 layer를 시도해보지 못한 점
- 실패한 시도를 포함하여 모든 기록을 세세하기 기록하지 않은 점
- 진행 사항이 옳은지 판단이 안되고 서로의 코드를 검증할 여력이 없음

6-2. 개선 방안

- 알려진 모델의 하이퍼 파라미터 튜닝 시, 논문의 parameter를 참고하여 약간의 후보군에 대해 grid search하는 방안을 제시합니다.
- 데이터에서 pronoun과 noun을 뽑아 category를 만든 후 같은 category에 해당하는 단어들로 치환하여 증강하는 방안을 제시합니다.
- 아래 그래프는 BERT tokenizer를 거친 sentence1 데이터의 토큰 길이 분포입니다.



  토큰 길이 기준으로 elbow 지점을 잘라내어 짧은 토큰들로만 학습 후 긴 토큰들을 추가학습하는 방법으로 더 빠른 결과를 얻을 수 있을 것으로 기대됩니다.
- Pytorch nn.Module 클래스의 get_parameters 메소드의 결과를 사용한다면, 직접 시각화를 시도해볼 수 있을 것으로 기대됩니다.
- time 라이브러리를 사용하여 훈련마다 기록을 저장한다면, 더 객관적이고 의미 있는 보고서를 작성할 수 있을 것으로 기대됩니다.
- 모델을 쪼개고 GPU를 직렬로 연결하여 파이프 라인 방식으로 연속으로 처리하는 방법을 사용한다면, 대용량 모델을 처리할 수 있을 것으로 기대됩니다.
- 기획 단계에서 목표와 제한을 뚜렷하게 정하고 Test Driven Development를 통해 프로젝트를 진행하는 방법으로 확신을 가지고 진행할 수 있을 것으로 기대됩니다.
- 국립국어원에서 제공하는 NIKLex를 통해 WordNet을 구축하고 데이터 증강을 시도 했다면 데이터에 적용하고 더 좋은 성능을 보여줄 수 있을 것으로 기대됩니다.

7. Reference

optuna tutorial

optuna 공식문서