부스트캠프

Wrap-Up 리포트

Stage 2 - KLUE

이름 : 김동우

캠퍼ID : T1017

◆ 본인의 점수 및 순위

→ 리더보드 accuracy : 78.7%, 81등

◆ 분석의 마음가짐

- EDA 결과 데이터 수가 적었고 불균형이 심했다.
- 따라서, competition score에 연연하기보다는 다양한 시도를 해보기로 다짐하였다. (하이퍼 파라미터 튜닝보다는 여러 방법론을 적용)

◆ 분석 과정

1. Stage2 첫 주에는 처음 접해보는 nlp의 분석 과정을 baseline code를 통해 살펴보았다.

- entity와 sentence가 어떻게 데이터 셋으로 구성되는지
- tokenizer가 어떻게 tokenize시키는지
- tokenize의 결과인 input_ids, token_type_ids, attention_mask가 어떻게 구성되는지
- model에 input이 들어가면 output의 형태가 어떤지
- huggingface의 tokenier, trainer, config 등 여러 함수는 무슨 파라미터를 입력으로 받아 어떻게 작동되는지

…

2. baseline code의 흐름과 코드들의 기능을 어느 정도 파악한 뒤에는 기존 baseline code를 토대로 나만의 baseline code(Mybaseline)를 만들었다.

3. 기존의 baseline code에서 아무 변경 없이 학습한 결과

→ LB accuracy = 72.5

4. Mybaseline 사용

- config01

model name	bert-base-multilingual-cased
tokenize max length	100
seed	7
learning rate	5e-5
batch size	16
warm up steps	500
weight decay	0.01
label smoothing factor	0
val size	0.3

- result

check point	val accuracy	LB accuracy
1500	68.9	67.3
2000	67.6	
2500	71.4	71.3
3000	**71.9**	**71.5**
3500	71.5	71.4

- explanation

stage1 때 profile 별로 validation을 나누지 않아 학습 과정에서 validation set도 overfitting 되는 문제가 일어났다. 따라서 validation score로 성능을 파악할 수 없었고 이번에도 이를 확인하기 위해 거의 모든 checkpoint 모델을 제출하여 validation score가 성능 지표가 될 수 있을지를 확인하였다.

⇓

[변경 사항]

val size : 0.3 → 0 (모든 데이터를 학습에 사용)

⇓

- **config02**

model name	bert-base-multilingual-cased
tokenize max length	100
seed	7
learning rate	5e-5
batch size	16
warm up steps	500
weight decay	0.01
label smoothing factor	0
val size	**0**

- result

check point	val accuracy	LB accuracy
2500		72.0
3000		**72.5**
3500		71.5

- explanation

epoch이 아닌 step으로 model을 저장하여 모든 데이터를 사용해도 같은 step에 가장 좋은 성능이 나타나는 것을 확인하였다.
따라서, 이후에는 validation set split으로 최선의 step을 찾아 모든 데이터를 사용하여 해당 step의 모델로 inference 하였다.

⇓

[변경 사항]

토론 게시판에 캠퍼 분들이 제공한 추가 데이터 사용

데이터가 20만개가 넘어 batch size : 16 → 64

⇓

- **config03**

model name	bert-base-multilingual-cased
tokenize max length	100
seed	7
learning rate	5e-5
batch size	**64**
warm up steps	500
weight decay	0.01
label smoothing factor	0
val size	0.2

- result

check point	val accuracy	LB accuracy
3000	81.59	71.3
7000	**83.43**	**71.5**
9000	83.43	71.4

- explanation

개수가 적은 라벨의 데이터가 추가되어 전체적인 일반화에는 도움이 되었을지 몰라도 training set과 라벨 분포가 비슷한 test set에서는 성능이 좋지 않게 나왔다.
그래도 개수가 적은 라벨의 분류에는 강점이 있을 것 같아 차후에 앙상블에 사용하기로 하였다.

⇓

[변경 사항]

위 모델들을 앙상블

⇓

- **config04**

  (soft vote ensemble) config01 ckpt-3000 + config03 ckpt-9000

- result

check point	val accuracy	LB accuracy
3000 + 9000		75.0

- explanation

기존 데이터로 학습된 모델과 추가 데이터로 학습된 모델을 앙상블하니 기존 모델보다 3% 이상의 성능이 개선된 결과가 나왔다.
대부분을 차지하는 “관계_없음”은 기존 데이터로 학습된 모델이 분류해주고 수가 적은 라벨에 분류에는 추가 데이터로 학습된 모델이 영향을 끼쳐 성능이 올라간 것 같다.

⇓

[변경 사항]

additional special token [ENT], [/ENT] 추가

⇓

- **config05**

model name	bert-base-multilingual-cased
tokenize max length	100
seed	7
learning rate	5e-5
batch size	16
warm up steps	500
weight decay	0.01
label smoothing factor	0
val size	0.3

- result

check point	val accuracy	LB accuracy
2000	71.19	
2500	**72.44**	**71.0**
3000	71.63	

- explanation

처음에는 entity 앞, 뒤로 [ENT], [/ENT]를 추가하고 entity의 위치는 1로 표시한 entity embedding을 token_type_ids에 더하여 학습해보았지만, 성능이 너무 좋지 않았다.

아마 기존 pre-trained는 token_type_ids가 0과 1인 경우에만 학습되었는데, entity embedding을 더하게 되면 2인 경우도 있기 때문에 학습이 잘 안된 것 같다.

따라서 [ENT], [/ENT]만 추가하여 학습해보았는데, 이것 또한 결과가 그리 좋지 않았다.

이 방법론에 대해 모델이 충분한 데이터로 학습하여야 되는데, 데이터 양은 너무 적고 기존 pre-trained 방법과 달라 성능이 좋지 않은 것 같다.

⇓

[변경 사항]

val size : 0.3 → 0 (모든 데이터를 학습에 사용)

⇓

- config06

model name	bert-base-multilingual-cased
tokenize max length	100
seed	7
learning rate	5e-5
batch size	16
warm up steps	500
weight decay	0.01
label smoothing factor	0
val size	**0**

- result

check point	val accuracy	LB accuracy
2500		**70.0**

- explanation

기존 모델의 성능이 좋지 않으니 모든 데이터를 학습에 사용해도 성능이 좋지 않았다.

⇓

[변경 사항]

model : bert → xlm-roberta

learning rate : 5e-5 → 1e-5

batch size : 16 → 32

warm up steps : 500 → 300

label smoothing factor : 0 → 0.5

⇓

- **config07**

model name	**xlm-roberta-large**
tokenize max length	100
seed	7
learning rate	**1e-5**
batch size	**32**
warm up steps	**300**
weight decay	0.01
label smoothing factor	**0.5**
val size	0.3

- result

check point	val accuracy	LB accuracy
1500	75.3	74.3
1800	**75.41**	**74.9**
1900	75.3	74.6

- explanation

xlm 모델은 하이퍼파라미터를 미세하게 조정하지 않으면 학습이 잘 안되거나 validation set의 evaluation이 잘 되지 않았다. 많은 시도에도 학습이 잘 되지 않아 토론 게시판에서 성능이 좋다고 했던 하이퍼파라미터를 빌려 학습하였다.
전과 다른 특이점은 label smoothing factor가 추가되었다는 것인데, 분류된 확률이 적은 라벨에 대해서도 학습을 원활하게 시키게 하여 잘 되는 것일까 생각하였다.

⇓

[변경 사항]

val size : 0.3 → 0 (모든 데이터를 학습에 사용)

⇓

- **config08**

model name	xlm-roberta-large
tokenize max length	100
seed	7
learning rate	1e-5
batch size	32
warm up steps	300
weight decay	0.01
label smoothing factor	0.5
val size	**0**

- result

check point	val accuracy	LB accuracy
1800		**78.7**
1900		78.0

- explanation

전체 데이터로 학습이니 성능이 더욱 올랐다. 테스크에 맞는 모델을 찾는 것이 매우 중요하다는 것을 느꼈다.

⇓

[변경 사항]

위 모델들을 앙상블

⇓

- **config09**

  1. (soft vote ensemble) config07 ckpt-1800 + config03 ckpt-9000
  2. (hard vote ensemble) config07 ckpt-1800 + config03 ckpt-9000 + config04

- result

check point	val accuracy	LB accuracy
1. 1800 + 9000		77.8
2. 1800 + 9000 + ens		77.8

- explanation

config04에서는 기존 데이터로 학습한 모델과 추가 데이터로 학습한 모델을 앙상블 하니 성능이 좋아졌지만, xlm을 사용한 모델과 앙상블하니 성능이 떨어졌다.
xlm의 성능이 기존보다 너무 좋다보니 다른 모델과의 앙상블은 오히려 성능을 저하시킨 것 같다.

5. QuestionAnswering task

- 관계 추출 task를 QA task로 변형하여 분류를 시도하였다.
- 라벨 별로 질문을 구성하였다.
  '인물:배우자': 'entity1의 배우자는 누구인가요?',
  '인물:직업/직함': 'entity1의 직업은 무엇인가요?', ...
- 데이터 셋은 sentence와 위에서 생성된 question을 붙여 tokenize한 것과 질문에 답에 해당하는 entity2의 start index와 end index로 구성하였다.
- 위에서 만든 데이터 셋을 BertForQuestionAnswering으로 학습하였다.

- inference 방법은 일단 기존 모델로 “관계_없음” 만을 분류한다.
- 그 다음, 하나의 test data에 대해 모든 질문에 대한 question을 만들고 데이터 셋과 같은 구성의 test 데이터 셋을 구성한다.
  (한 test data 당 41개의 데이터가 생성)
- 모든 질문에 대한 답 중 entity2와 가장 가까운 답을 가지는 질문을 선별하고 해당 질문에 대한 라벨로 데이터를 분류한다.

- 하지만 inferene 결과가 질문 별로 다 비슷하였고 이 결과도 매우 유의미하지 않아서 실패하였다.

◆ 시도했던 것들과 잘된 혹은 잘되지 않은 이유

• 잘된 것

1. 앙상블
   → 서로 완전히 다른 관점에서 학습된 두 모델의 앙상블의 효과는 대단하였다.
2. task에 맞는 모델 선정 (xlm)
   → 하이퍼파라미터 튜닝이나 다른 기능을 추가하는 것보다 성능을 더욱 크게 향상시킬 수 있었다.
3. label smoothing
   → 모든 라벨에 대해 골고루(?) 학습시켜주는 것은 생각보다 모델 학습에 안정성을 가져올 수 있었다.

• 잘되지 않은 것

1. 앙상블
   → 다른 관점에 비슷한 성능을 앙상블하면 큰 향상을 불러올 수 있지만, 성능이 작은 모델과의 앙상블은 성능을 하락시킬 수도 있었다.
2. [ENT], [/ENT] 토큰 추가, entity embedding 추가
   → 아이디어 자체는 너무 좋았지만, fine-tuning 하기엔 데이터 수도 적었고 기존 pre-trained의 데이터 셋과 너무 달랐다.
   → 게다가, 'RuntimeError: CUDA error: CUBLAS_STATUS_ALLOC_FAILED when calling 'cublasCreate(handle)' 에러로 하루종일 학습이 안되었는데, [ENT], [/ENT] 토큰을 tokenizer에 추가시켜주고 model의 embedding size을 resize할 뿐만 아니라 model의 token_type_embedding에 직접 접근하여 차원을 바꿔줘야 했었다.
3. QA task
   → 기존 pipeline을 사용하여 시험 삼아 training 데이터 셋을 분류해본 결과도 매우 안좋았고 실제 학습시켜서 몇몇 데이터의 output만 봐도 성능이 매우 안좋았다.
   → 데이터 셋이 적어 fine-tuning이 거의 안된 탓도 있을 것이다.

◆ 시도하지 않았던, 못했던 아이디어와 그 이유

1. NER을 추가하여 분류

→ 성능이 향상되었다는 이야기를 많이 들었지만, 정확이 어떻게 개체명을 추가하여 학습하는 것인지 알지 못하여 시도하지 않았다. 이후에 탐색해보고 구현해 봐야겠다.

2. 여러 data augmentation

→ back translation과 같은 여러 방법들이 토론 게시판에 나왔지만, 의미의 변경이나 entity의 변경 등 다양한 문제를 동반할 것이라 생각했다. 따라서 augmentation보단 다른 방법에 집중하였다.

3. Pseudo labeling 방법

→ 괜찮은 아이디어라 생각하였지만, 시간 관계상 직접 시도해보진 못하였다.

4. “관계_없음”과 다른 것 이진 분류 후 다른 것들만 다시 분류

→ 가장 먼저 해보고 싶은 아이디어였으나, 피어 분이 해본 결과 성능이 좋지 않았고 이는 오히려 학습의 난이도를 떨어트려 성능이 향상되지 않을 것 같아 시도하지 않았다.

◆ 아쉬웠던 점과 보완할 점

1. [ENT], [/ENT], entity embedding

→ 위에서도 말했듯이 에러를 해결하는 과정에서 너무 많은 시간을 사용하였다. 하지만 이로 인해 모델의 구조나 학습 과정을 자세하게 알게 되었다.

2. QA task

→ 이 방법론을 제대로 숙지하지 못하였다. 데이터 셋 구성, 학습까지는 원활했으나 성능이 좋지 않았고 inference 과정이 확실하지 않았다. 차후에는 라벨은 다르나 질문이 비슷한 문제에 차별점을 두고 inference 과정도 충분히 숙지하여 구현해 봐야겠다.

3. 다양한 아이디어 탐색

→ 시간이 촉박하여 독창적인 아이디어를 생각하지 못하고 직접 아이디어를 찾지도 못하였다. 토론 게시판이나 피어세션을 통해 전해 들은 아이디어만을 사용해본 것이 아쉽다. 다음엔 직접 아이디어를 찾아보며 정보를 찾는 능력도 향상시켜야 할 것 같다.

◆ 느낀점

NLP에 대해서는 막연하게 어려운 분야라는 생각이 있었습니다. 컴퓨터 비전과 같이 확실한 이미지를 보며 학습을 하고 결과를 낼 수 없을 것이라고 생각했습니다. 이런 생각 때문에 NLP라는 분야에 흥미는 크지 않았습니다.

하지만, 이번 Stage를 통하여 NLP의 학습 흐름을 알게 되었고 생각보다 많은 task와 다양한 방법들이 존재했음을 배웠습니다. 그 중 챗봇과 같은 실용적인 모델들이 어쩌면 컴퓨터 비전보다도 더 매력있을 수도 있겠구나 생각했습니다.

아직은 AI 분야에서 막 걸음마를 뗀 수준이라 어느 분야가 잘 맞고 흥미로운지는 알지 못하나 NLP에 대한 저의 생각을 깬 것만으로도 충분히 의미 있는 Stage 였습니다.

이번 Stage를 계기로 더 넓은 시야로 AI 분야를 보며 더 다양한 정보를 배울 수 있을 것입니다. 감사합니다.