DACON

Pre-trained 모델 설명자료

AI 야, 진짜 뉴스를 찾아줘! AI 경진대회

K-러닝

2020-1-2

모델 설명 : SKTBrain의 KoBERT이용 custom_bert_clf

(출처: GitHub - SKTBrain/KoBERT: Korean BERT pre-trained cased (KoBERT))

- KoBERT 모델에 3개의 linear classifier를 추가하여 custom_bert_clf라는 모델을 생성하여 학습을 하였습니다. 모델을 구성하는 라이브러리 및 구조는 아래와 같습니다.

1. 사용된 라이브러리들 및 버전

mxnet 1.7.0

tqdm 4.41.1

sentencepiece 0.1.94

gluonnlp 0.10.0

transformers 3.0.0

datasets 1.1.3

pytorch 1.7.0+cu101

kobert 0.1.2

numpy 1.19.4

sklearn 0.22.2.post1

2. KoBERT를 선택한 이유

구글에서 공개한 기존의 BERT에서도 다국어 모델을 지원하고 있지만, 한국어 corpus를 영어와 동일한 방법으로 학습시켰기 때문에 발생하는 한계점이 존재합니다. 따라서 대량의 한국어 corpus를 이용해 학습시킨 모델인 SKTBrain의 KoBERT를 사용하였습니다.

3. 모델의 Architecture.

1) SKTBrain의 KoBERT config

```
predefined_args = {
        'attention_cell': 'multi_head',
        'num_layers': 12,
        'units': 768,
        'hidden_size': 3072,
        'max_length': 512,
        'num_heads': 12,
        'scaled': True,
        'dropout': 0.1,
        'use_residual': True,
        'embed_size': 768,
        'embed_dropout': 0.1,
        'token_type_vocab_size': 2,
        'word_embed': None,
    }
```

2) custom_bert_clf 구조

```
custom_bert_clf(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(8002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0): BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
```

```
          (value): Linear(in_features=768, out_features=768, bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (output): BertSelfOutput(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
      )
      (output): BertOutput(
        (dense): Linear(in_features=3072, out_features=768, bias=True)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
```

……이하 11개 BertLayer, 총 12개……

```
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
)
(dropout): Dropout(p=0.1, inplace=False)
(classifier): Linear(in_features=768, out_features=256, bias=True)
(classifier2): Linear(in_features=256, out_features=64, bias=True)
(classifier3): Linear(in_features=64, out_features=2, bias=True)
(gelu): GELU()
    )
```

3) custom_bert_clf의 config


```
{
  "architectures": [
    "custom_bert_clf"
  ],
  "attention_probs_dropout_prob": 0.1,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 1,
  "type_vocab_size": 2,
  "vocab_size": 8002
}
```


사용한 Tokenizer : distilKoBERT의 KoBertTonkenizer

(출처: GitHub - monologg/DistilKoBERT: Distillation of KoBERT from SKTBrain (Lightweight KoBERT))

1. KoBertTokenizer

- Sentencepiece 기반의 토크나이저로 KoBERT모델에서 제공하는 pre trained 토크나이저 입니다. DistilKoBERT에는 KoBERT에서 사용된 동일한 토크나이저가 transformers 라이브러리에 호환되도록 제작되어 있습니다.
- custom_bert_clf는 대부분 transformes 라이브러리를 이용하여 학습 및 예측이 이뤄졌기 때문에 monologg/DistilKoBERT 에서 제공하는 KoBertTokenizer를 사용했습니다.

2. KoBertTokenizer Train corpus

데이터	문장	단어
한국어 위키	5M	54M
한국어 뉴스	20M	270M

3. KoBertTokenizer Vocabulary

A. 크기: 8,002

B. 한글 위키 + 뉴스 텍스트 기반으로 학습한 Tokenizer(SentencePiece)

C. Less number of parameters(92M < 110M)

custom_bert_clf 학습에 사용한 파라미터

1. 사용한 파라미터

```
from transformers import Trainer, TrainingArguments

training_args = TrainingArguments(output_dir= '/content/drive/MyDrive/dacon/results_1229',
                                  num_train_epochs= 5, #훈련 횟수를 증가시킴
                                  per_device_train_batch_size= 64,
                                  per_device_eval_batch_size = 64,
                                  warmup_steps = int(1670 * 0.001), #warm up하는 즉 lr을 0으로 시작하는 케이스를 줄임
                                  weight_decay = 0.001, #예전보다 규제의 강도가 줄어들음
                                  save_steps = 1000,
                                  overwrite_output_dir=True,
                                  evaluate_during_training=True)
```

2. 설명

1) output_dir : 모델 예측과 체크포인트가 기록되는 output directory
2) num_train_epochs : 수행할 총 training epoch의 수
3) per_device_train_batach_size : 학습에 사용할 batch 사이즈
4) per_device_eval_batch_size : 평가에 사용할 batch 사이즈
5) warmup_steps : ‘warmup_steps’ 학습 단계에 걸쳐 learning rate schedule을 0에서 1로 선형적으로 증가시키는 역할을 합니다.
6) weight_decay : 기존의 가중치를 일정 비율로 감소시킨 다음 에러 값의 미분한 값을 차감하여 가중치가 급격하게 증가하는 것을 막아준다. 이를 통해 overfitting을 방지할 수 있습니다.
7) save_steps : 두 개의 체크포인트가 저장되기 전, 업데이트 단계의 수
8) overwrite_output_dir : True인 경우, output directory의 내용을 덮어 씁니다.
9) evaluate_during_training : True인 경우, 각 단계에서 학습 중에 평가를 실행합니다.