팀_데이터베이_코드설명

- 해당 코드는 GPU가 없는 노트북을 사용하여 시간이 오래 걸리는 것 때문에 구글 코랩이 아닌 Vscode를 이용하여 실행시켰습니다.
- Vscode의 업데이트 상황이나 여러가지 변수로 동작하지 않을 수 있습니다.
- 가급적이면 구글 코랩을 사용해주세요.
- 또한 KoBert의 기본적인 사용 예제를 가져와서 이용했습니다.

코드 진행 순서

1) 최대 길이 확인

1. 데이터 불러오기
2. 토큰화 및 정수화
3. 민원중 최대길이 찾기

2) 모델 학습

1. BERT모델, Vocabulary 불러오기
2. 학습용 데이터 불러오기
3. 데이터 전처리
4. 파라미터 및 옵티마이저 설정
5. 모델 학습
6. 모델 검증
7. 최종 데이터 생성 및 저장

최대 길이 확인

데이터 불러오기

```
train_df = pd.read_json("trn.json")
val_df = pd.read_json("valid.json")

x_train = []
x_test = []
for d in train_df['Q'].values:
    x_train.append(d)

for d in val_df['Q'].values:
    x_test.append(d)
```

-> 리스트형식을 만들기 위해 'Q' 항목만 리스트로 뽑아냈습니다.

토큰화 및 정수화

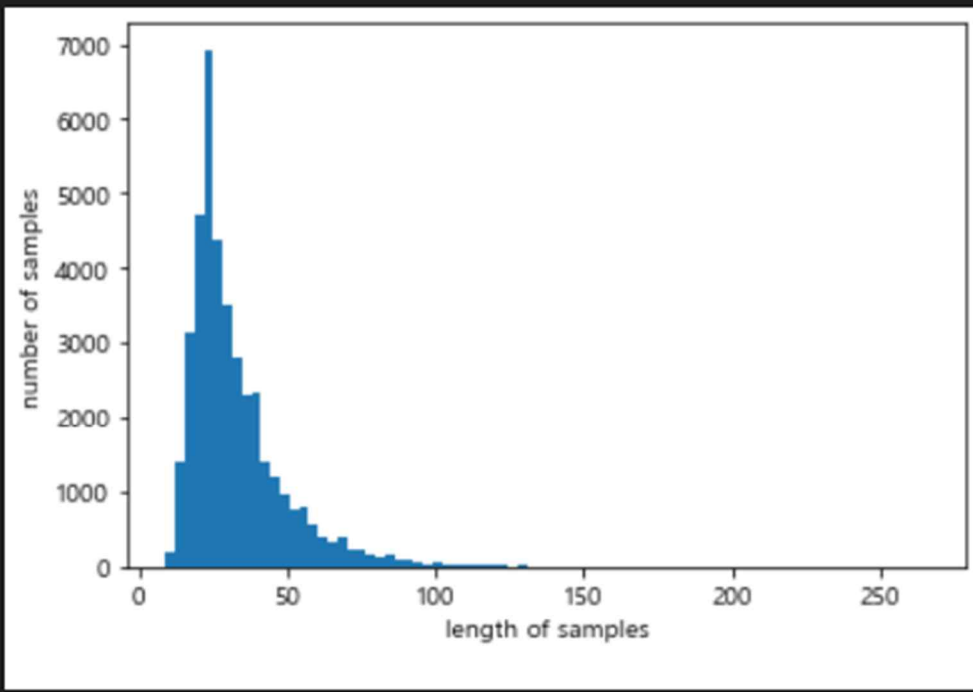
```
import urllib.request
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# 토큰화 및 정수화
tokenizer = Tokenizer()
tokenizer.fit_on_texts(x_train)
X_train_encoded = tokenizer.texts_to_sequences(x_train)
print(X_train_encoded[:5])
```

민원 중 길이가 가장 긴 길이를 찾음

```
print('문의 최대 길이 : %d' % max(len(l) for l in X_train_encoded))
print('문의 평균 길이 : %f' % (sum(map(len, X_train_encoded))/len(X_train_encoded)))
plt.hist([len(s) for s in x_train], bins=80)
plt.xlabel('length of samples')
plt.ylabel('number of samples')
plt.show()
```

```
문의 최대 길이 : 68
문의 평균 길이 : 7.299400
```



Kobert 학습 모델

예제에 포함된 코드로 install부터 시작

```
!pip install mxnet
!pip install gluonnlp pandas tqdm
!pip install sentencepiece
!pip install transformers==3
# 최신 버전으로 설치하면 "Input: must be Tensor, not str" 라는 에러 발생
!pip install torch
```

```
# Git허브에서 KoBERT 파일 로드
!pip install git+https://git@github.com/SKTBrain/KoBERT.git
```

```
import torch
from torch import nn
import torch.nn.functional as F
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import gluonnlp as nlp
import numpy as np
from tqdm import tqdm, tqdm_notebook
```

```
from kobert.utils import get_tokenizer
from kobert.pytorch_kobert import get_pytorch_kobert_model

from transformers import AdamW
from transformers.optimization import get_cosine_schedule_with_warmup
```

BERT모델, Vocabulary 불러오기

```
# BERT모델, Vocabulary 불러오기
bertmodel, vocab = get_pytorch_kobert_model()
```

```
using cached model
using cached model
```

GPU 사용 시

```
device = torch.device("cuda:0")
```

Json으로 된 파일 읽어오기.

```
import pandas as pd

# Json으로 된 파일 읽어오기.
train_df = pd.read_json("trn.json") # 코드파일과 json파일의 위치가 일치하기 때문에 파일명만 입력.
val_df = pd.read_json("valid.json")

x_train = []
x_test = []
for d in train_df['Q'].values:
    x_train.append(d)

for d in val_df['Q'].values:
    x_test.append(d)
```

train data의 id 속성 제거

```
train_df.drop(['id'], axis = 1, inplace= True)
train_df.head(5)
```

-> 'Q'와 'dep' 속성만 남기기 위해

train data의 dep 정수로 치환

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder.fit(train_df['dep'])Z
train_df['dep'] = encoder.transform(train_df['dep'])
train_df.head()
```

	Q	dep
0	진해구 제황산동 일반쓰레기와 재활용쓰레기의 배출일자를 알고자 합니다.	72
1	위택스에서 자동차세 연납신청을 하려합니다. 기간은 언제부터인지 부서에 확인 부탁드립니다.	50
2	자꾸 #@전번#으로 전화가 계속 오네요. 창원시청으로 확인되는데, 어느 부서에서 연...	65
3	기한시험 합격 후 구비서류 문의드립니다.	64
4	온누리상품권 가맹점 등록 문의드립니다.	10

dep 정수가 무슨 부서인지 확인 및 총 몇개로 분류를 진행하면 되는지 확인

```
mapping = dict(zip(range(len(encoder.classes_)), encoder.classes_))
mapping
```

```
Output exceeds the size limit. Open the full output data in a text editor
{0: '가정복지과',
 1: '개발사업과',
 2: '건강관리과',
 3: '건강증진과',
 4: '건설과',
 5: '건설도로과',
 6: '건축경관과',
 7: '건축허가과',
 8: '경남도청',
 9: '경제교통과',
 10: '경제기업사랑과',
 11: '경제살리기과',
 12: '공원개발과',
 13: '관광과',
 14: '교통물류과'
```

-> 99개의 부서가 나옴

Val data 전처리

```
# val data 전처리1
val_df.head()
```

	id	Q	dep
0	C0-0	10월 22일 기초생활 수급자로 확정되었는데, 의료비 혜택이 궁금해서 문의합니다.	사회복지과
1	C1-0	전봇대에 불법 현수막이 게시되어 있습니다. 단속 요청합니다.	건축허가과
2	C2-0	화물차 정기분 등록면허세를 내라는 고지서를 받아보았습니다. 근데 전년도보다 면허세가...	세무과
3	C3-0	작년에 자동차세 연납신청을 하고 납부완료했습니다. 올해도 적용이 되는지요.	세무과
4	C4-0	의창구 대원동에서 임대사업을 하고자 합니다. 임대사업자 등록을 하려는 어떻게 해야 ...	건축허가과

```
# val data 전처리2
val_df.drop(['id'], axis = 1, inplace= True)
val_df.head(5)
```

	Q	dep
0	10월 22일 기초생활 수급자로 확정되었는데, 의료비 혜택이 궁금해서 문의합니다.	사회복지과
1	전봇대에 불법 현수막이 게시되어 있습니다. 단속 요청합니다.	건축허가과
2	화물차 정기분 등록면허세를 내라는 고지서를 받아보았습니다. 근데 전년도보다 면허세가...	세무과
3	작년에 자동차세 연납신청을 하고 납부완료했습니다. 올해도 적용이 되는지요.	세무과
4	의창구 대원동에서 임대사업을 하고자 합니다. 임대사업자 등록을 하려는 어떻게 해야 ...	건축허가과

 # val_data 전처리3 앞서 train data를 정수로 변환 했기 때문에 맞춰서 변환하기위해 mapping을 뒤집었습니다.

```
re_mapping = dict(map(reversed, mapping.items()))
re_mapping
```

Output exceeds the size limit. Open the full output data in a text editor

```
{'가정복지과': 0,
 '개발사업과': 1,
 '건강관리과': 2,
 '건강증진과': 3,
 '건설과': 4,
 '건설도로과': 5,
 '건축경관과': 6,
 '건축허가과': 7,
 '경남도청': 8,
 '경제교통과': 9,
 '경제기업사랑과': 10,
```

```
idx = []
for d in val_df['dep'].values :
  idx.append(re_mapping.get(d))
# print(idx)
val_df['dep'] = idx
val_df.head()
```

	Q	dep
0	10월 22일 기초생활 수급자로 확정되었는데, 의료비 혜택이 궁금해서 문의합니다.	40
1	전봇대에 불법 현수막이 게시되어 있습니다. 단속 요청합니다.	7
2	화물차 정기분 등록면허세를 내라는 고지서를 받아보았습니다. 근데 전년도보다 면허세가...	49
3	작년에 자동차세 연납신청을 하고 납부완료했습니다. 올해도 적용이 되는지요.	49
4	의창구 대원동에서 임대사업을 하고자 합니다. 임대사업자 등록을 하려는 어떻게 해야 ...	7

기본 Bert tokenizer 사용 토큰화와 정수화 , 패딩 함수

```
# 기본 Bert tokenizer 사용 토큰화와 정수화 , 패딩 함수
tokenizer = get_tokenizer()
tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)
print(tok)

class BERTDataset(Dataset):
    def __init__(self, dataset, sent_idx, label_idx, bert_tokenizer, max_len,
                 pad, pair):
        transform = nlp.data.BERTSentenceTransform(
            bert_tokenizer, max_seq_length=max_len, pad=pad, pair=pair)

        self.sentences = [transform([i[sent_idx]]) for i in dataset]
        self.labels = [np.int32(i[label_idx]) for i in dataset]

    def __getitem__(self, i):
        return (self.sentences[i] + (self.labels[i], ))

    def __len__(self):
        return (len(self.labels))
```

파라미터 설정 - max_len 과 batch_size, num_epochs, log_interval 을 제외하고는 예제와 똑같이 설정했습니다.

```
max_len = 68 # 제일 처음 최대길이를 구해서 68이라는 것을 알았기 때문에 최대 길이를 68로 설정하였습니다.
batch_size = 50
warmup_ratio = 0.1
num_epochs = 7 # 학습 횟수, 5번은 82.02% 가 나왔고 10번은 과적합으로 오히려 더 적게 나왔음.
max_grad_norm = 1
log_interval = 20 # 학습 중 진행도를 나타낼 간격
learning_rate = 5e-5
```

```
data_train = BERTDataset(train_df.values, 0, 1, tok, max_len, True, False)
data_test = BERTDataset(val_df.values, 0, 1, tok, max_len, True, False)
```

pytorch용 DataLoader 사용

```
# pytorch용 DataLoader 사용
train_dataloader = torch.utils.data.DataLoader(data_train, batch_size=batch_size, num_workers=0, shuffle=True)
test_dataloader = torch.utils.data.DataLoader(data_test, batch_size=batch_size, num_workers=0, shuffle=True)
# shuffle = True, 를 해주면 데이터가 섞이면서 학습하게 된다.
```

```
# 토큰화와 패딩이 잘 이루어졌는지 확인.
data_train[0]
```

```
(array([   2, 4360, 7848, 5495, 4128, 7951, 6516, 5872, 3807, 6779, 6050,
        5561, 6983, 3988, 7003, 6779, 6050, 5561, 7095, 2295, 7126, 7158,
        3168, 7147, 4986,  517,   54,    3,    1,    1,    1,    1,    1,
           1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
           1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
           1,    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
           1,    1]),
 array(28),
 array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0]),
 72)
```

-> 68개의 길이로 패딩된 것을 확인 할 수 있습니다.

BERT 모델 불러오기

```
class BERTClassifier(nn.Module):
    def __init__(self,
                 bert,
                 hidden_size = 768,
                 num_classes = 99, # train data의 dep을 정수화 시킬 때, 총 분류 가짓수가 99개인 것을 확인 하였음.
                 dr_rate=None,
                 params=None):
        super(BERTClassifier, self).__init__()
        self.bert = bert
        self.dr_rate = dr_rate

        self.classifier = nn.Linear(hidden_size , num_classes)
        if dr_rate:
            self.dropout = nn.Dropout(p=dr_rate)
```

```
    def gen_attention_mask(self, token_ids, valid_length):
        attention_mask = torch.zeros_like(token_ids)
        for i, v in enumerate(valid_length):
            attention_mask[i][:v] = 1
        return attention_mask.float()

    def forward(self, token_ids, valid_length, segment_ids):
        attention_mask = self.gen_attention_mask(token_ids, valid_length)

        _, pooler = self.bert(input_ids = token_ids, token_type_ids = segment_ids.long(), attention_mask = attention_mask.float().to(token_ids.device))
        if self.dr_rate:
            out = self.dropout(pooler)
        return self.classifier(out)
```

```
# model = BERTClassifier(bertmodel, dr_rate=0.5).to(device)  # GPU사용시 해당 코드로 실행
model = BERTClassifier(bertmodel, dr_rate=0.5)  # GPU가 없어서 이것을 사용
```

optimizer 와 schedule 설정 사용 예제와 동일하게 가져옴

```
no_decay = ['bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
    {'params': [p for n, p in model.named_parameters() if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
    {'params': [p for n, p in model.named_parameters() if any(nd in n for nd in no_decay)], 'weight_decay': 0.0}
]

# 옵티마이저 선언
optimizer = AdamW(optimizer_grouped_parameters, lr=learning_rate)
loss_fn = nn.CrossEntropyLoss() # softmax용 Loss Function 정하기 <- binary classification도 해당 loss function 사용 가능

t_total = len(train_dataloader) * num_epochs
warmup_step = int(t_total * warmup_ratio)

scheduler = get_cosine_schedule_with_warmup(optimizer, num_warmup_steps=warmup_step, num_training_steps=t_total)
```

학습 평가 지표인 accuracy 계산

```
# 학습 평가 지표인 accuracy 계산
def calc_accuracy(X,Y):
    max_vals, max_indices = torch.max(X, 1)
    train_acc = (max_indices == Y).sum().data.cpu().numpy()/max_indices.size()[0]
    return train_acc
```

모델 학습 시작

 # .to(device) 가 있는 부분들은 모두 GPU를 사용시

GPU없이 실행하였음.

```
# 모델 학습 시작
# .to(device) 가 있는 부분들은 모두 GPU를 사용시
# GPU없이 실행하였음.
for e in range(num_epochs):
    train_acc = 0.0
    test_acc = 0.0
    model.train()
    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm_notebook(train_dataloader)):
        optimizer.zero_grad()
        # token_ids = token_ids.long().to(device)
        # segment_ids = segment_ids.long().to(device)
        token_ids = token_ids.long()
        segment_ids = segment_ids.long()
        valid_length= valid_length
        # label = label.long().to(device)
        label = label.long()
        out = model(token_ids, valid_length, segment_ids)
        loss = loss_fn(out, label)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_grad_norm) # gradient clipping
        optimizer.step()
        scheduler.step()  # Update learning rate schedule
        train_acc += calc_accuracy(out, label)
        if batch_id % log_interval == 0:
            print("epoch {} batch id {} loss {} train acc {}".format(e+1, batch_id+1, loss.data.cpu().numpy(), train_acc / (batch_id+1)))
    print("epoch {} train acc {}".format(e+1, train_acc / (batch_id+1)))
```

학습 저장

```
# 학습 저장
torch.save(model, 'my_model_torch_7epochs_shuffle_True_1207.h5')
```

Val data를 이용하여 평가진행

```
# 테스트 문장 예측
test_acc = 0.0
test_sentence = '세금 관련 질문 있습니다.'
test_label = 49 # 실제 부서
model.eval() # 평가 모드로 변경
unseen_test = pd.DataFrame([[test_sentence, test_label]], columns = [['Q', 'dep']])
unseen_values = unseen_test.values
test_set = BERTDataset(unseen_values, 0, 1, tok, max_len, True, False)
test_input = torch.utils.data.DataLoader(test_set, batch_size=1, num_workers=0)

for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm(test_input)):
#   token_ids = token_ids.long().to(device)
#   segment_ids = segment_ids.long().to(device)
  token_ids = token_ids.long()
  segment_ids = segment_ids.long()
  valid_length= valid_length
  out = model(token_ids, valid_length, segment_ids)
  max_vals, max_indices= torch.max(out, 1)
  print("max_vals : ", max_vals)
  print("max_indices : ", max_indices) # 학습된 모델이 예측한 부서의 정수값이 출력된다.
```

```
model.eval() # 평가 모드로 변경

for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm(test_dataloader)):
  #   token_ids = token_ids.long().to(device)
  #   segment_ids = segment_ids.long().to(device)
    token_ids = token_ids.long()
    segment_ids = segment_ids.long()
    valid_length= valid_length
  #   label = label.long().to(device)
    label = label.long()
    out = model(token_ids, valid_length, segment_ids)
    test_acc += calc_accuracy(out, label)
    print("epoch {} batch id {} test acc {}".format(e+1, batch_id+1, test_acc / (batch_id+1)))
print("epoch {} test acc {}".format(e+1, test_acc / (batch_id+1)))
```

결과 제출용 test 데이터 불러오기 및 전처리, 예측

```
# 결과 제출용 test 데이터 불러오기 및 전처리
test_df = pd.read_json("test.json")
id_list = []
for i in test_df['id'].values:
    id_list.append(i)
q_list = []
for i in test_df['Q'].values:
    q_list.append(i)
test_df.drop(['id'], axis = 1, inplace= True)

dep = [0 for i in range(0, len(test_df))]
test_df['dep'] = dep
test_df.head()
```

```
unseen_values = test_df.values
test_set = BERTDataset(unseen_values, 0, 1, tok, max_len, True, False)
test_input = torch.utils.data.DataLoader(test_set, batch_size=1, num_workers=0)
```

```
model.eval() # 평가 모드로 변경

result_dep = []
for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(tqdm(test_input)):
    #   token_ids = token_ids.long().to(device)
    #   segment_ids = segment_ids.long().to(device)
    token_ids = token_ids.long()
    segment_ids = segment_ids.long()
    valid_length= valid_length
    out = model(token_ids, valid_length, segment_ids)
    max_vals, max_indices= torch.max(out, 1)
    r=max_indices.tolist()
    print("max_indices : ",r) # 모델이 예측하는 민원의 부서 출력
    result_dep.append(r[0])
```

결과 정리

```
# 결과 정리
dep_str = []
for i in result_dep :
    dep_str.append(mapping.get(i))


result = pd.DataFrame()
result['id'] = id_list
result['Q'] = q_list
result['dep'] = dep_str

result.head()
```

```
# 결과 저장
result.to_json("test_accuracy_1207_trainacc.json", orient='records', indent=3, force_ascii=False)
```

➔ 제출 데이터 형식에 맞추기위해 orient = 'records'