Machine Learning (Autumn 2023)
Homework 8th week-2 (November 4)

Student ID 2022712151

Name 정지원

Instruction: This report is a LightGBM-based Legal Document Classification Model for Claims/Facts/Judgments/Conclusions

(1) 목표

1. AIHub에서 텍스트 데이터 수집을 진행한다.
2. 데이터에 대한 탐색적 분석을 수행하고 활용 목적에 알맞는 전처리를 수행한다.
3. TF-IDF를 활용해 텍스트 데이터를 임베딩한다.
4. 임베딩된 데이터로 LightGBM모델을 학습시켜 판결문의 주장/사실/판단/결론에 대한 분류를 수행한다.

(2) 데이터 소개 및 수집

이 데이터셋은 Aihub 의 법률/규정 (판결서, 약관등) 텍스트 분석 데이터로 1 만 건 이상의 판결문을 대상으로 기초사실, 주장 등을 가공한 데이터와 판례 내용을 기반으로 판결문 분석 데이터 구축, 1 만건 이상의 약관의 유/불리 조항 판단. 위법성과 유리 판단 이유 태깅 및 레이블링을 통해 소비자 입장에서의 유/불리 확인을 위한 법률 텍스트 분석 데이터셋이다. AiHub 의 (https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=580) 사이트를 통해 다운로드를 받았다.[그림 1]

[그림 1] 데이터 다운로드

이번 과제의 목적은 위 데이터 중 판결문 데이터(TL_1. 판결문)를 가지고 주장, 사실, 판단, 결론이라는 4 가지의 레이블을 분류하는 과정을 거친다.

(3) EDA 및데이터 전처리

(3.1) 라이브러리 import 및 초기설정

- 시각화에 사용할 ml-things을 설치한다.
- 사용하고자 하는 분석에 필요한 라이브러리들을 불러온다.

Code:

```
import pandas as pd
import numpy as np
import re
import io
import os
from tqdm.notebook import tqdm
tqdm.pandas()
from torch.utils.data import Dataset, DataLoader
from ml_things import plot_dict, plot_confusion_matrix, fix_text
from sklearn.metrics import classification_report, accuracy_score

# GPU 메모리 및 최대 시퀀스 길이에 따른 배치 수 설정이 필요
# 시퀀스 길이가 짧다면 32 이상의 배치 사이즈를 사용하는 것이 가능
batch_size = 32

data_path ='./라벨링데이터/TL_1.판결문'

# 분류 라벨
labels_ids = {'01_주장':0, '02_사실':1, '03_판단':2, '04_결론':3}

# How many labels are we using in training.
# This is used to decide size of classification head.
n_labels = len(labels_ids)
```

(3.2) EDA 수행 및 데이터 전처리

- 사용하고자 하는 데이터 샘플을 불러와 살펴본다.
- Json 확장자 파일에 9개의 key(info, concerned, org, disposal, mentionedItems, assrs, facts, dcss, close)와 각 key에 해당하는 내용들이 포함되어 있다.

Code:

- 읽어온 JSON 객체에 특정 Key의 존재 여부를 확인하기 위한 is_json_key_present 함수를 정의한다.
- 설정한 경로에서 데이터를 불러와 딕셔너리로 반환하는 JsonDataet 클래스를 정의한다.

Code:

```
import json
def is_json_key_present(json, key):
    try:
        buf = json[key]
    except KeyError:
        return False
    return True

class JsonDataset(Dataset):


  def __init__(self, path):

    # Check if path exists.
    if not os.path.isdir(path):
      # Raise error if path is invalid.
      raise ValueError('Invalid `path` variable! Needs to be a directory')
    self.texts = []
    self.labels = []


    for label in ['01.민사', '02.형사', '03.행정']:
      sentiment_path = os.path.join(path, label)

      # Get all files from path.
      for (root, directories, files) in os.walk(sentiment_path):
      # Go through each file and read its content.
        for file_name in tqdm(files, desc=f'{root} files'):
          file_path = os.path.join(root, file_name)
          with open(file_path, "r") as json_file:
            json_data  =  json.load(json_file)
            # print('json_data',json_data)
            if(is_json_key_present(json_data,"assrs") and
is_json_key_present(json_data['assrs'],"acusrAssrs")) :
              for data in json_data['assrs']['acusrAssrs'] :
                self.texts.append(fix_text(data))
                self.labels.append('01_주장')

            if(is_json_key_present(json_data,"assrs") and
is_json_key_present(json_data['assrs'],"dedatAssrs")) :
              for data in json_data['assrs']['dedatAssrs'] :
                self.texts.append(fix_text(data))
                self.labels.append('01_주장')

            if(is_json_key_present(json_data,"facts") and
is_json_key_present(json_data['facts'],"bsisFacts")) :
              for data in json_data['facts']['bsisFacts'] :
                self.texts.append(fix_text(data))
```

```
            self.labels.append('02_사실')


          if(is_json_key_present(json_data,"dcss") and
is_json_key_present(json_data['dcss'],"courtDcss")) :
            for data in json_data['dcss']['courtDcss'] :
              self.texts.append(fix_text(data))
              self.labels.append('03_판단')

          if(is_json_key_present(json_data,"close") and
is_json_key_present(json_data['close'],"cnclsns")) :
            for data in json_data['close']['cnclsns'] :

              self.texts.append(fix_text(data))
              self.labels.append('04_결론')
    # Number of exmaples.
    self.n_examples = len(self.labels)
    return

  def __len__(self):
    return self.n_examples

  def __getitem__(self, item):
    return {'text':self.texts[item],
            'label':self.labels[item]}
```

- Train, test 데이터 분리를 진행해준다.
  - 사전 정의한 Jsondataset 클래스로 데이터를 불러온다.
  - 불러온 데이터에 pd.DataFrame.from_records를 적용해 데이터프레임으로 만든다.
  - Train_test_split()을 통해 데이터프레임을 9:1의 비율로 나눈다.
  - Label 별로 나누어져 있는 데이터프레임을 합쳐 훈련 데이터와 테스트 데이터를 만든다.

Code :

```
import pandas as pd
from sklearn.model_selection import train_test_split


# JSON Data
dataset = JsonDataset(path=data_path)
df = pd.DataFrame.from_records(dataset)
df.groupby('label').size()

df = df.sample(frac=0.05, random_state=123)

df1 = df[df['label'].str.startswith('01')]
```

```
df2 = df[df['label'].str.startswith('02')]
df3 = df[df['label'].str.startswith('03')]
df4 = df[df['label'].str.startswith('04')]

# 전체 데이터에서 train, test 분리 (9 : 1)
df1_train, df1_test = train_test_split(df1, test_size=0.1,
random_state=123)
df2_train, df2_test = train_test_split(df2, test_size=0.1,
random_state=123)
df3_train, df3_test = train_test_split(df3, test_size=0.1,
random_state=123)
df4_train, df4_test = train_test_split(df4, test_size=0.1,
random_state=123)

df_train = pd.concat([df1_train,df2_train,df3_train,df4_train])
df_test = pd.concat([df1_test,df2_test,df3_test,df4_test])
```

- StopWords 처리
  - Konlpy 라이브러리에서 한국어 형태소 분석기인 Okt를 import한다.
  - 한국어에서 중요한 의미를 가지지 않는 단어들을 불용어 처리하기 위한 set을 만든다.
  - Okt와 stop_words를 활용하여 텍스트 데이터를 전처리하는 함수를 정의한다.
  - 데이터에 전처리를 수행하고 새로운 column prepro_data로 추가한다.
  - 전처리 과정에서 모든 문장이 삭제되어 공백이 되는 데이터가 존재할 수 있으므로 해당 데이터를 제거한다.
  - 데이터를 정렬하고 데이터를 shuffle한다.

Code :

```
# 한국어 형태소 분석기 Okt 를 사용하기 위해 import
from konlpy.tag import Okt
# Okt 의 객체 생성
okt = Okt()

# 한국어에서 의미가 거의 없는 단어들을 불용어 처리하기 위해 집합을 만듦
stop_words = set(['은', '는', '이', '가', '아', '하', '들', '것', '의', '있',
'되', '수', '보', '주', '등', '한', '에'])
# 전처리 함수
def preprocessing(okt, review):
  # 한글이 아닌 단어는 지우고 " "으로 대체
  review_text = re.sub("[^가-힣ㄱ-ㅎㅏ-ㅣ]", " ", review)
  # 형태소 분석을 하며 stemming 기법을 사용
  token_li = okt.morphs(review_text, stem=True)
  # 이전 stop_words 에 포함되지 않는 단어만 clean_review 에 들어갈 수 있게 함
  clean_review = [token for token in token_li if not token in stop_words]
  # 단어들을 ' '기준으로 이어 붙혀서 하나의 string 타입의 변수를 생성함
```

```
  clean_review = ' '.join(clean_review)

  # 전처리 완료된 문장을 return
  return clean_review.strip()
# preprocessing 함수를 적용
df_train["prepro_data"] = df_train["text"].progress_apply(lambda x:
preprocessing(okt, x))
df_test["prepro_data"] = df_test["text"].progress_apply(lambda x:
preprocessing(okt, x))
# 전처리 과정에서 모든 문장이 삭제될 가능성이 있으므로, 해당 데이터 제거
```

(4) 모델링 및 결과 확인

1. TFidfVectorizer를 통해 훈련, 테스트 데이터 embedding

Code :

```
# tfidf embedding
from sklearn.feature_extraction.text import TfidfVectorizer

# TfidfVectorizer를 초기화하여 사용할 수 있다.
tfidf_vectorizer = TfidfVectorizer(max_features=5000)

# 훈련 데이터에 대해 fit_transform을 수행하여 TF-IDF 행렬을 생성한다.
x_train_tfidf = tfidf_vectorizer.fit_transform(x_train)

# 테스트 데이터에 대해 transform을 수행하여 TF-IDF 행렬을 생성한다.
x_test_tfidf = tfidf_vectorizer.transform(x_test)
```

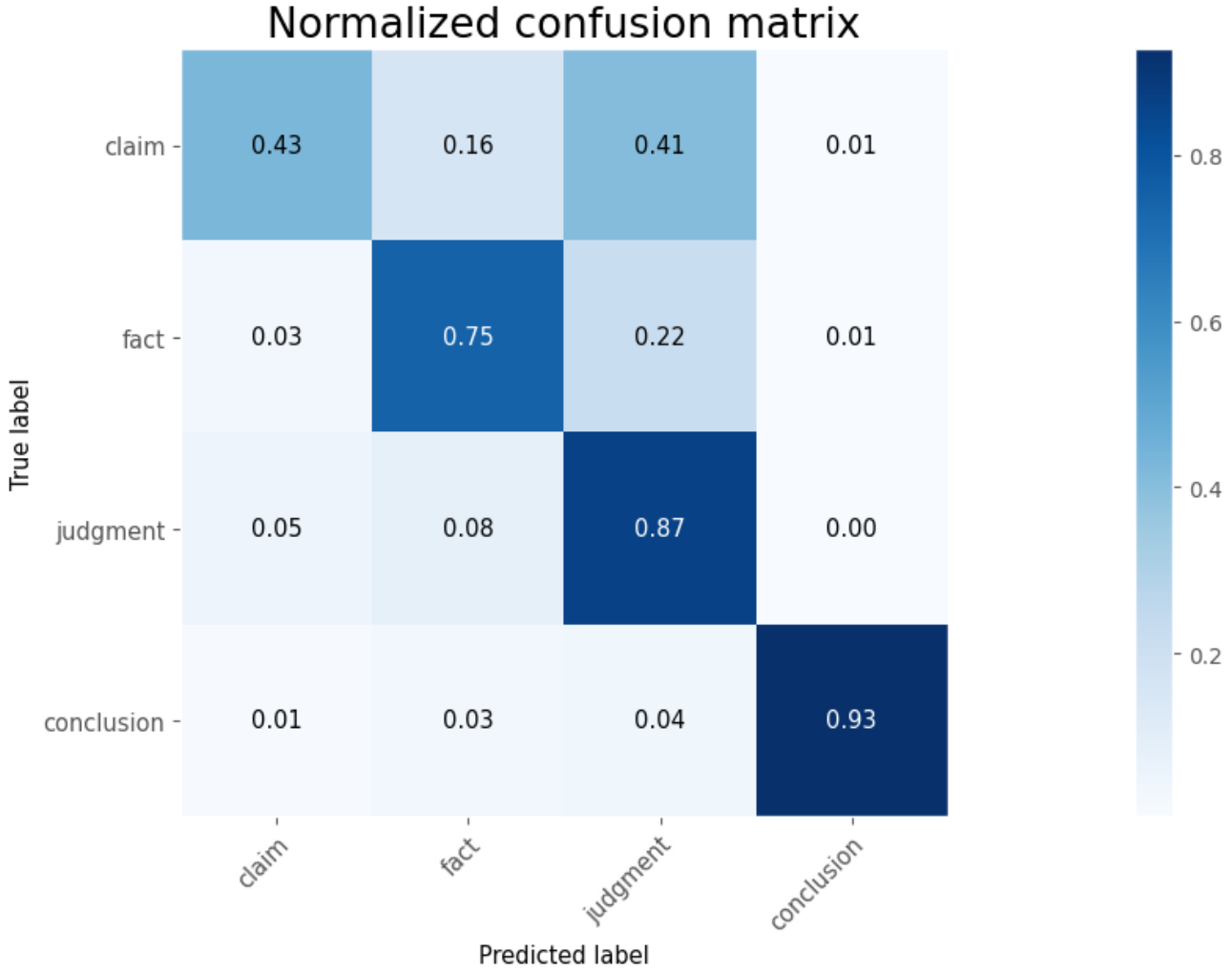
2. LightGBM 모델을 사용하며 GridSearch를 통해 5-fold cross validation 방식으로 모델을 학습시킨다.

Code :

```
# modeling
from lightgbm import LGBMClassifier

# LightGBM 분류기를 초기화한다.
lgb_classifier = LGBMClassifier()

# 사용하고자 하는 파라미터 조합을 설정한다.
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.05, 0.1],
    'num_leaves': [31, 63, 127]
}

# GridSearchCV를 사용하여 5-fold cross validation을 수행하며 모델을 학습시킨다.
from sklearn.model_selection import GridSearchCV
```

```
grid_search = GridSearchCV(lgb_classifier, param_grid, cv=5,
scoring='accuracy',n_jobs=-1,verbose=2)
grid_search.fit(x_train_tfidf, y_train)

# 최적의 파라미터를 출력한다.
print("Best Parameters:", grid_search.best_params_)
Best Parameters: {'learning_rate': 0.05, 'n_estimators': 200, 'num_leaves':
31}
```

3. 훈련된 모델을 적용하여 4 진 분류(0:주장, 1:사실, 2:판단, 3:결론)를 수행하고 모델의 성능을 확인한다.



Results :

```
Classification Report:
              precision    recall  f1-score   support

      01_주장       0.66      0.43      0.52       177
      02_사실       0.76      0.75      0.75       327
      03_판단       0.78      0.87      0.82       591
      04_결론       0.95      0.93      0.94       111

    accuracy                           0.78      1206
   macro avg       0.79      0.74      0.76      1206
weighted avg       0.77      0.78      0.77      1206
```