|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 2021-12-01 |
| 교육장소 | YGL 학과장 및 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. TF-IDF, skikit-learn을 활용한 TF-IDF구현   from sklearn.feature\_extraction.text import CountVectorizer  corpus = [  'you know I want your love',  'I like you',  'what should I do'  ]  vector = CountVectorizer()  print(vector.fit\_transform(corpus).toarray())  # do know like love should want what you your (tfidf.vocab에서 나온 0~7순서)  print(vector.vocabulary\_)  from sklearn.feature\_extraction.text import TfidfVectorizer  corpus = [  'you know I want your love',  'I like you',  'what should I do'  ]  tfidfv = TfidfVectorizer().fit(corpus)  print(tfidfv.transform(corpus).toarray())  print(tfidfv.vocabulary\_)   1. 로이터 데이터로 TF-IDF학습하기   from tensorflow.keras.datasets import reuters  import matplotlib.pyplot as plt  import seaborn as sns  import numpy as np  import pandas as pd  (x\_train, y\_train),(x\_test, y\_test) = reuters.load\_data(num\_words=10000, test\_split=0.2)  print('훈련 샘플의 수 : {}'.format(len(x\_train)))  print('테스트 샘플의 수 : {}'.format(len(x\_test)))  print(x\_train[0])  print(x\_test[0])  print(y\_train[0]) # 훈련 기사의 레이블  num\_classes = max(y\_train)+1  print('클래스의 수 : {}'.format(num\_classes))  print('훈련용 뉴스의 최대 길이 : {}'.format(max(len(I)for I in x\_train)))  print('훈련용 뉴스의 평균 길이 : {}'.format(sum(map(len, x\_train))/len(x\_train)))  plt.hist([len(s) for s in x\_train], bins= 50)  plt.xlabel('length of samples')  plt.ylabel('number of samples')  plt.legend()  plt.show()  fig, axe = plt.subplots(ncols =1)  fig.set\_size\_inches(12, 5)  sns.countplot(y\_train)  unique\_elements, counts\_elements = np.unique(y\_train, return\_counts=True)  print('각 클래스 빈도수 : ')  print(np.asarray((unique\_elements, counts\_elements)))  word\_to\_index = reuters.get\_word\_index()  print(word\_to\_index)  index\_to\_word = {}  for key, value in word\_to\_index.items():  # value에 3을 더하는 이유 : pad, eos, OOV 같은건 앞에 몰아두기때문에 비우고 시작하는다는 의미  index\_to\_word[value+3] = key  word\_to\_index['the']  print(index\_to\_word[4]) # 빈도수 상위 1번째 단어  print(index\_to\_word[16]) # 빈도수 상위 13번째 단어  # 0 <pad>  # 1 <sos>  # 2 <unk>  for index, token in enumerate(("<pad>","<sos>","<unk>")):  index\_to\_word[index] = token  print(' '.join([index\_to\_word[index] for index in x\_train[0]]))  # 전체 데이터에 대해서 decoded  decoded = []  for i in range(len(x\_train)):  t = ' '.join([index\_to\_word[index] for index in x\_train[i]])  decoded.append(t)  x\_train = decoded  # 전체 데이터에 대해서 decoded  decoded = []  for i in range(len(x\_test)):  t = ' '.join([index\_to\_word[index] for index in x\_test[i]])  decoded.append(t)  x\_test = decoded   1. TF-IDF로 로이터 데이터 학습하기   from sklearn.model\_selection import train\_test\_split  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.feature\_extraction.text import TfidfTransformer  from sklearn.naive\_bayes import MultinomialNB  from sklearn import metrics  # 단어의 수를 카운트하는 사이킷런의 카운트벡터라이저.  count\_vect = CountVectorizer()  # fit\_transform : 학습 할 때와 동일한 기반 설정으로 동일하게 테스트 데이터를 변환해야 하는 것  x\_train\_counts = count\_vect.fit\_transform(x\_train)  # 카운트벡터라이저의 결과로부터 TF-IDF 결과를 얻습니다  tfidf\_transformer = TfidfTransformer()  x\_train\_tfidf = tfidf\_transformer.fit\_transform(x\_train\_counts)  # 나이브 베이즈 분류기를 수행  # x\_train은 TF-IDF의 벡터, y\_train 레이블  clf = MultinomialNB().fit(x\_train\_tfidf, y\_train)  def tfidf\_vectorizer(data):  data\_counts = count\_vect.transform(data)  data\_tfidf = tfidf\_transformer.transform(data\_counts)  return data\_tfidf  y\_pred = clf.predict(tfidf\_vectorizer(x\_test))  print(metrics.classification\_report(y\_test, y\_pred))   1. LstM모델 구현   import tensorflow as tf  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, LSTM, Embedding  from tensorflow.keras.preprocessing.sequence import pad\_sequences  from tensorflow.keras.utils import to\_categorical  from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint  from tensorflow.keras.models import load\_model  # 로이터 로드  (x\_train, y\_train), (x\_test, y\_test) = reuters.load\_data(num\_words=1000, test\_split=0.2)  # max\_len= 100 --> pad\_sequences  max\_len = 100  x\_train = pad\_sequences(x\_train, maxlen= max\_len)  x\_test = pad\_sequences(x\_test, maxlen=max\_len)  # y\_train --> one-hot encoding  y\_train = to\_categorical(y\_train)  y\_test = to\_categorical(y\_test)  # 모델  vocab\_size = 1000  embedding\_dim = 128  hidden\_units = 128  num\_classes = 46  # 모델 옵션지정  model = Sequential()  model.add(Embedding(vocab\_size, embedding\_dim))  model.add(LSTM(hidden\_units))  model.add(Dense(num\_classes, activation='softmax'))  es = EarlyStopping(monitor='val\_loss', mode='min', verbose=1, patience= 4)  mc = ModelCheckpoint('best\_model.h5', monitor='val\_acc', mode='max', verbose = 1, save\_best\_only=True)  model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['acc'])  # history = 모델 학습  history = model.fit(x\_train, y\_train, batch\_size = 128, epochs= 30, callbacks=[es, mc], validation\_data=(x\_test, y\_test))  # 베스트모델을 h5변수에다가 저장  loaded\_model = load\_model('best\_model.h5')  # 테스트 정확도  print("\n 테스트 정확도 : %.4f" % (loaded\_model.evaluate(x\_test, y\_test)[1]))  # train과 test loss function  epochs = range(1, len(history.history['acc'])+1)  plt.plot(epochs, history.history['loss'])  plt.plot(epochs, history.history['val\_loss'])  plt.title('model loss')  plt.ylabel('loss')  plt.xlabel('epoch')  plt.legend(['train', 'test'], loc='upper left')  plt.show()   1. 실제뉴스기사 크롤링 및 분류   !pip install beautifulsoup4  !pip install newspaper3k  !pip install konlpy  !git clone <https://github.com/SOMJANG/Mecab-ko-for-Google-Colab.git>  cd Mecab-ko-for-Google-Colab/  !bash install\_mecab-ko\_on\_colab190912.sh  from konlpy.tag import Mecab  mecab = Mecab()  mecab.morphs('하하하하하이')   1. BeautifulSoup의 사용법   from bs4 import BeautifulSoup  html = '''  <html>  <head>  </head>  <body>  <h1>장바구니  <p id='clothes' class= 'name' title = '라운드티'> 라운드티  <span class = 'number'> 25 </span>  <span class = 'price'> 29000 </span>  <span class = 'menu'> 의류 </span>  <a href = "http://www.naver.com"> 바로가기 </a>  </p>  <p id='watch' class= 'name' title = '시계'> 시계  <span class = 'number'> 28 </span>  <span class = 'price'> 32000 </span>  <span class = 'menu'> 악세사리 </span>  <a href = "http://www.facebook.com"> 바로가기 </a>  </p>  </h1>  </body>  </html>  '''  soup = BeautifulSoup(html, 'html.parser')  print(soup.select('body'))  print(soup.select('h1 .name .menu'))   1. Newspaper3k 패키지   from newspaper import Article  url = 'https://www.etnews.com/20200503000018'  article = Article(url, language='ko')  article.download()  article.parse()  print('기사 제목 : ')  print(article.title)   1. BeautifulSoup와 newspaper3k를 통해 크롤러 만들기   import requests  import pandas as pd  from bs4 import BeautifulSoup  def make\_urllist(page\_num, code, date):  urllist= []  for i in range(1, page\_num + 1):  url = 'https://news.naver.com/main/list.nhn?mode=LSD&mid=sec&sid1='+str(code)+'&date='+str(date)+'&page='+str(i)  headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.90 Safari/537.36'}  news = requests.get(url, headers=headers)  soup = BeautifulSoup(news.content, 'html.parser')  # CASE 1  news\_list = soup.select('.newsflash\_body .type06\_headline li dl')  # CASE 2  news\_list.extend(soup.select('.newsflash\_body .type06 li dl'))    # 각 뉴스로부터 a 태그인 <a href ='주소'> 에서 '주소'만을 가져옵니다.  for line in news\_list:  urllist.append(line.a.get('href'))  return urllist  url\_list = make\_urllist(2, 101, 20200506)  print('뉴스 기사의 갯수 :',len(url\_list))  idx2word = {'101': '경제', '102':'사회','103':'생활/문화','105':'과학'}  from newspaper import Article  # 데이터프레임을 생성하는 함수  def make\_date(urllist, code):  text\_list = []  for url in urllist:  article = Article(url, language='ko')  article.download()  article.parse()  text\_list.append(article.text)    # 데이터프레임의 'news'티 아래 파싱한 텍스트를 밸류로 붙여준다.  df = pd.DataFrame({'news':text\_list})  # 데이터프레임의 'code'키 아래 한글 카테고리명을 붙여준다.  df['code'] = idx2word[str(code)]  return df  data = make\_date(url\_list, 101)  data[:10]   1. 데이터 수집 및 전처리   code\_list = [102, 103, 105]  def make\_total\_data(page\_num, code, date):  df = None    for code in code\_list:  url\_list = make\_urllist(page\_num, code, date)  df\_temp = make\_date(url\_list, code)  print(str(code) + '번 코드에 대한 데이터를 만들었습니다. ')  if df is not None:  df = pd.concat([df, df\_temp])  else:  df = df\_temp  return df  df = make\_total\_data(1, code\_list, 20200506)  print('뉴스 기사의 갯수 :', len(df))  df = make\_total\_data(100, code\_list, 20200506)  import os  csv\_path = './news\_data.csv'  df.to\_csv(csv\_path, index=False)  if os.path.exists(csv\_path):  print('{} File Saved!'.format(csv\_path))  df = pd.read\_table(csv\_path, sep=',')  df.head()  df['news'] = df['news'].str.replace("[^ㄱ-ㅎㅏ-ㅣ 가-힣]","")  df['news']  print(df.isnull().sum())  df.drop\_duplicates(subset=['news'],inplace=True)  print('뉴스 기사의 갯수 :', len(df))   1. 데이터 탐색   df['code'].value\_counts().plot(kind='bar')  print(df.groupby('code').size().reset\_index(name='count'))  from konlpy.tag import Mecab  tokenizer = Mecab()  kor\_text = '밤에 귀가하던 여성에게 범죄를 시도한 대 남성이 구속됐다서울 제주경찰서는 \  상해 혐의로 씨를 구속해 수사하고 있다고 일 밝혔다씨는 지난달 일 피해 여성을 \  인근 지하철 역에서부터 따라가 폭행을 시도하려다가 도망간 혐의를 받는다피해 \  여성이 저항하자 놀란 씨는 도망갔으며 신고를 받고 주변을 수색하던 경찰에 \  체포됐다피해 여성은 이 과정에서 경미한 부상을 입은 것으로 전해졌다'  print(tokenizer.morphs(kor\_text))   1. 불용어 제거   stopwords = ['에','는','은','을','했','에게','있','이','의','하','한','다','과','때문','할','수','무단','따른','및','금지','전재','경향신문','기자','는데','가','등','들','파이낸셜','저작','등','뉴스']  def preprocessing(data):  text\_data = []  for sentence in data:  temp\_data = []  temp\_data = tokenizer.morphs(sentence)  temp\_data = [word for word in temp\_data if not word in stopwords]  text\_data.append(temp\_data)  text\_data = list(map(' '.join, text\_data))  return text\_data  text\_data = preprocessing(df['news'])  from sklearn.model\_selection import train\_test\_split  from sklearn.feature\_extraction.text import CountVectorizer  from sklearn.feature\_extraction.text import TfidfTransformer  from sklearn.naive\_bayes import MultinomialNB  from sklearn import metrics  x\_train, x\_test, y\_train, y\_test = train\_test\_split(text\_data, df['code'], random\_state=0)  print('훈련용 뉴스 기사의 갯수 : ', len(x\_train))  print('테스트용 뉴스 기사의 갯수 : ', len(x\_test))  print('훈련용 레이블의 갯수 : ', len(y\_train))  print('테스트용 레이블의 갯수 : ', len(y\_test))  # DTM  count\_vect = CountVectorizer()  x\_train\_counts = count\_vect.fit\_transform(x\_train)  # TF-IDF 행렬  tfidf\_transformer = TfidfTransformer()  x\_train\_tfidf = tfidf\_transformer.fit\_transform(x\_train\_counts)  # 모델학습 (나이브베이즈모델)  clf = MultinomialNB().fit(x\_train\_tfidf, y\_train)   1. 평가 및 테스트   def tfidf\_vectorizer(data):  data\_counts = count\_vect.transform(data)  data\_tfidf = tfidf\_transformer.transform(data\_counts)  return data\_tfidf  new\_sent = preprocessing(["민주당 일각에서 법사위의 체계·자구 심사 기능을 없애야 한다는 \  주장이 나오는데 대해 “체계·자구 심사가 법안 지연의 수단으로 \  쓰이는 것은 바람직하지 않다”면서도 “국회를 통과하는 법안 중 위헌\  법률이 1년에 10건 넘게 나온다. 그런데 체계·자구 심사까지 없애면 매우 위험하다”고 반박했다."    ])  print(clf.predict(tfidf\_vectorizer(new\_sent)))  y\_pred = clf.predict(tfidf\_vectorizer(x\_test))  print(metrics.classification\_report(y\_test, y\_pred)) | |