에브리타임 강의평 분석

Attention

김주한 오재호 박지영

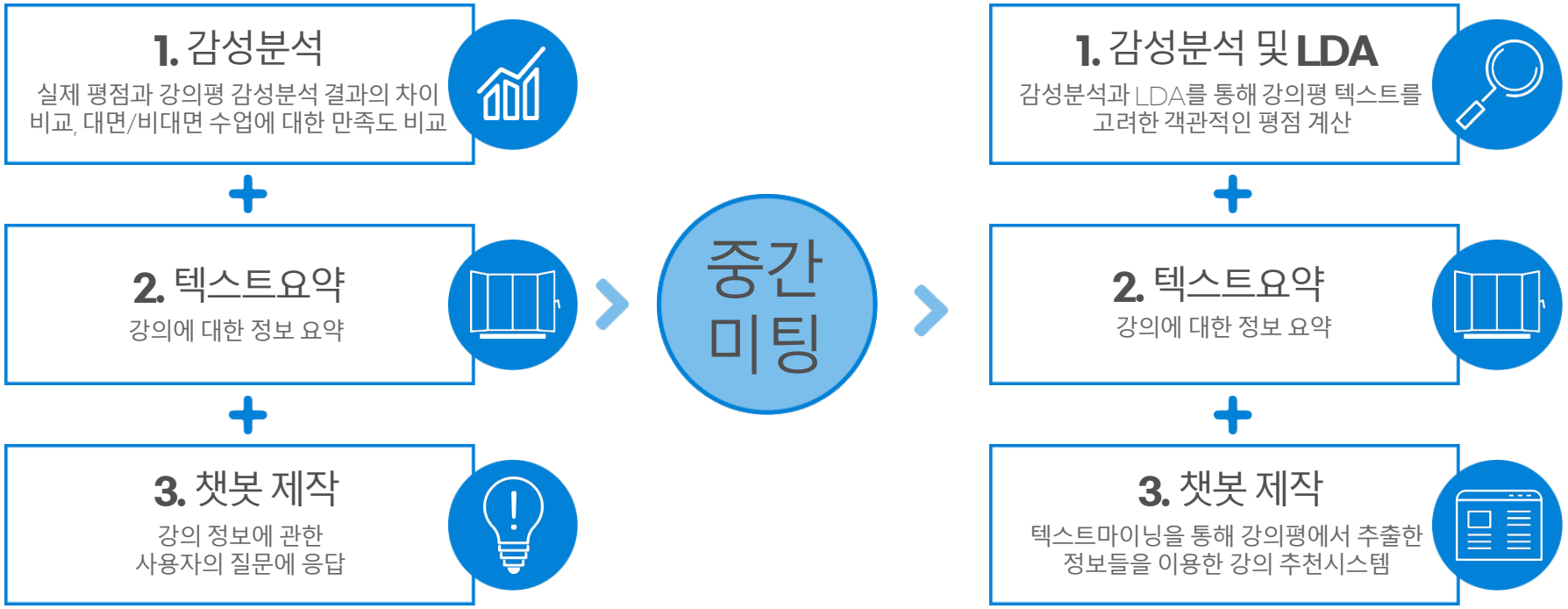
텍스트마이닝 팀프로젝트 최종발표

프로젝트 주제

에브리타임 강의평 분석

강의평 요약 서비스 및 챗봇 제공

프로젝트 소개


1. 감성분석
실제 평점과 강의평 감성분석 결과의 차이 비교, 대면/비대면 수업에 대한 만족도 비교
2. 텍스트요약
강의에 대한 정보 요약
3. 챗봇 제작
강의 정보에 관한 사용자의 질문에 응답
중간 미팅
1. 감성분석 및 LDA
감성분석과 LDA를 통해 강의평 텍스트를 고려한 객관적인 평점 계산
2. 텍스트요약
강의에 대한 정보 요약
3. 챗봇 제작
텍스트마이닝을 통해 강의평에서 추출한 정보들을 이용한 강의 추천시스템

데이터 수집

셀레니움을 사용한 에브리타임의 강의평 텍스트를 크롤링

Selenium

03

```
from selenium import webdriver
import time
import pandas as pd
import sys

data=pd.DataFrame()
r= 5
driver=webdriver.Chrome('./chromedriver.exe')#path설정
driver.maximize_window()
time.sleep(r)

year=2022 #년도
hacgi=1 #학기
#로그인
driver.get('https://everytime.kr/timetable/{year}/{hacgi}'.format(year=year,hacgi=hacgi))
driver.find_element_by_name('userid').send_keys('ma                    ')
driver.find_element_by_name('password').send_keys(                    'kjh')
driver.find_element_by_xpath('//*[@class="submit"]/input').click()
time.sleep(r)



driver.find_element_by_xpath('//*[@id="container"]/ul/li[1]').click() #시간표
time.sleep(r)
while True:
    before_e = driver.find_elements_by_css_selector('table > tbody > tr')[-1]
    driver.execute_script("arguments[0].scrollIntoView(true);", before_e)
    time.sleep(4)
    after_e =driver.find_elements_by_css_selector('table > tbody > tr')[-1]
    if before_e==after_e:
        break
```

데이터 수집

셀레니움을 사용한 에브리타임의 강의평 텍스트를 크롤링

04

```
num=len(driver.find_elements_by_css_selector('table > tbody > tr')) #강의수(반복수)
time.sleep(r)
for n in range(1,num+1):
    lecture_code=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[2]'.format(n=n)).text #과목코드
    classification=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[1]'.format(n=n)).text #구분(교필)
    lecture_name=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[3]'.format(n=n)).text #과목이름
    professor_name=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[4]'.format(n=n)).text #교수이름
    credit=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[5]'.format(n=n)).text #학점
    avg_score=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[7]/a'.format(n=n)).get_attribute('title') #평점
    print(lecture_code,classification,lecture_name,professor_name,credit,avg_score)

    url=driver.find_element_by_xpath('//*[@id="subjects"]/div[2]/table/tbody/tr[{n}]/td[7]/a'.format(n=n)).get_attribute('href') #tr[i]
    driver.get(str(url))
    time.sleep(r)


    #driver.switch_to.window(driver.window_handles[1])
    #time.sleep(8)

    lecture_review=driver.find_elements_by_css_selector('div.articles > article > p.text') #강의평
    semester=driver.find_elements_by_css_selector('div.articles > article > p.info > span.semester') #강의평 작성 학기
    score=driver.find_elements_by_css_selector('p.rate > span.star > span.on') #별점

    #추천수 속도 너무 느림
    # recommands=[]
    # for r in range(1,len(lecture_review)+1):
    #     try:
    #         recommand=driver.find_element_by_xpath('//*[@id="container"]/div[4]/div[2]/article[{n}]/p[2]/span[2]'.format(n=r)).text
    #         recommands.append(int(recommand))
    #     except:
    #         recommands.append(0)

    print(len(lecture_review),len(semester),len(score))
    time.sleep(r)
    lecture_reviews=[]
    semesters=[]
    scores=[]
    for i in lecture_review:
        lecture_reviews.append(i.text)
    for j in semester:
        semesters.append(j.text.replace(' 수강자',''))
    for k in score:
        x=float(k.get_attribute('style').replace('width: ','').replace('%;',''))/20
        scores.append(x)

    #print(lecture_reviews)
    #print(semesters)

    dic={}
    dic['lecture_code']=[lecture_code]*len(lecture_review)
    dic['classification']=[classification]*len(lecture_review)
    dic['lecture_name']=[lecture_name]*len(lecture_review)
    dic['professor_name']=[professor_name]*len(lecture_review)
    dic['credit']=[int(credit)]*len(lecture_review)
    dic['score']=scores
    dic['avg_score']=[float(avg_score)]*len(lecture_review)
    dic['semester']=semesters
    dic['lecture_review']=lecture_reviews
    data=data.append(pd.DataFrame(data=dic), ignore_index=True)

    time.sleep(r)
    #driver.close()
    #driver.switch_to.window(driver.window_handles[0])
    driver.back()
    time.sleep(r)
```

데이터 수집 컬럼 설명

lecture_code	강의코드	semester	년도 + 학기
classification	전필 / 전선 / 교필 / 교선 등	group_meeting	팀플 유무 (없음/보통/많음)
lecture_name	강의 이름	assignment	과제 유무 (없음/보통/많음)
professor_name	교수 이름	grade	학점 (보통/깐깐함/너그러움)
credit	학점	attendance	출석방식 (전자출결/직접호명 등)
score	평점	test_n	시험 횟수
avg_score	강의평 평점	lecture_review	강의평 텍스트

데이터 전처리

노이즈 캔슬링

re정규식, ~~hanspell~~

토크나이징

kobert 토크나이저, konlpy okt

불용어 처리

깃허브에 있는 500개 가량의 데이터로 처리
+ 200개 추가

데이터 전처리

```
import pandas as pd
from hanspell import spell_checker
df=pd.read_csv('movie_data.csv',encoding='utf-8-sig')
df
df['review']=df['review'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
#df['lecture_review']=spell_checker.check(df['lecture_review'].str).checked
a=df['review'].to_list()
b=[]
for i in a:
    try:
        b.append(spell_checker.check(i).checked)
    except:
        b.append('')
df['revised_review']=b
df.to_csv('movie_data_노이즈캔슬링.csv',encoding='utf-8-sig',index=False)
```

노이즈 캔슬링

re정규식, ~~hanspell~~

토크나이징

kobert 토크나이저, konlpy okt

불용어 처리

깃허브에 있는 500개 가량의 데이터로 처리
+ 200개 추가

데이터 전처리

```
okt = Okt()

data['word'] = 0

for i in data.index:
    data['word'][i] = okt.pos(data['revised_review'][i])

data['token'] = 0

for i in data.index :
    words = []
    for word in data['word'][i] :
        if word[1] in ['Verb', 'Noun', 'Adjective'] :
            words.append(word)
    data['token'][i] = words

data
```

노이즈 캔슬링

re정규식, ~~hanspell~~

토크나이징

kobert 토크나이저, konlpy okt

불용어 처리

깃허브에 있는 500개 가량의 데이터로 처리
+ 200개 추가

데이터 전처리

```
nouns = []

for i in data.index :
    for word in data['token'][i] :
        if word[1] in ['Noun'] :
            nouns.append(word[0])

from collections import Counter

count_nouns = Counter(nouns)
stopwords = pd.read_csv('불용어.csv')
stopwords = stopwords.drop(stopwords.columns[0], axis=1)
stopwords = stopwords['stopwords'].values.tolist()
top_400_nouns = count_nouns.most_common(n=400)
stopword_movie = []
for item in top_400_nouns :
    stopword_movie.append(item[0])

stopword_movie
```

	stopwords
0	영화
1	이
2	정말
3	것
4	거
...	...
395	할리우드
396	역
397	점점
398	어른
399	전부

400 rows × 1 columns

노이즈 캔슬링

re정규식, ~~hanspell~~

토크나이징

kobert 토크나이저, konlpy okt

불용어 처리

깃허브에 있는 500개 가량의 데이터로 처리
+ 200개 추가

감성분석

긍정/부정/중립
라벨링 실시

↓

Rescore를
위한 감성분석
(Kobert 이용)

Train/ test 데이터셋 나눠준 뒤 버트 토크나이저 적용

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data_list, test_size=0.2, random_state=42)
print("train shape is:", len(train))
print("test shape is:", len(test))
```

```
# 기본 Bert tokenizer 사용
tokenizer = get_tokenizer()
tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)

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))
```

감정분석을 위한 모델 학습

```
model = BERTClassifier(bertmodel, dr_rate=0.5).to(device)
# Prepare optimizer and schedule (linear warmup and decay)
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 -> 얼마나 타겟값을 많이 맞추었는가
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
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)
        valid_length= valid_length
        label = label.long().to(device)
        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)))

    model.eval() # 평가 모드로 변경

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

감성분석

긍정/부정/중립
라벨링 실시

↓

rescore를
위한 감성분석
(Kobert 이용)

강의평을 TEST 데이터로 넣어서 강의평을 긍정/부정/중립으로 예측

```
PATH = '/content/drive/My Drive/2022-1/텍마KoBERT_에타.pt'
model = BERTClassifier(bertmodel,  dr_rate=0.5).to(device)
model.load_state_dict(torch.load('/content/drive/My Drive/2022-1/텍마model_state_dict.pt'))
model.eval()

def softmax(a) :
    exp_a = np.exp(a)
    sum_exp_a = np.sum(exp_a)
    y=exp_a / sum_exp_a
    return y


#dataset_test
true_label=[]
test_eval=[]
tokenizer = get_tokenizer()
tok = nlp.data.BERTSPTokenizer(tokenizer, vocab, lower=False)
model.eval()


def softmax(a) :
    exp_a = np.exp(a)
    sum_exp_a = np.sum(exp_a)
    y=exp_a / sum_exp_a
    return y

model
########예측
df= pd.read_csv('/content/drive/My Drive/2022-1/텍마/everytime_data.csv')
df= df.iloc[10001:,:]
test_eval=[]
pred=[]
for i in df['lecture_review']:
    data = [i, '0']
    dataset_another = [data]
    another_test = BERTDataset(dataset_another, 0, 1, tok, max_len, True, False)
    test_dataloader = torch.utils.data.DataLoader(another_test, batch_size=batch_size, num_workers=5)

    for batch_id, (token_ids, valid_length, segment_ids, label) in enumerate(test_dataloader):
        token_ids = token_ids.long().to(device)
        segment_ids = segment_ids.long().to(device)

        valid_length= valid_length
        label = label.long().to(device)

        out = model(token_ids, valid_length, segment_ids)
        soft=out.detach().cpu().numpy()
        a=np.array(list(soft[0]))
        a=list(softmax(a))
        pred.append(max(a))
        test_eval.append(a.index(max(a)))


df['model_pre']=test_eval
df['pred']=pred
df.to_excel('/content/drive/My Drive/2022-1/텍마/결과1.xlsx',encoding='ANSI')
```

rescoring을 위한 모델의 예측 lable과 예측 확률을 칼럼으로 추가

test_n	ture_revi	ised_revi	word	token	model_pre	pred
0.5	교수님 ?	교수님 ?	[('교수',	[('교수',	1	0.8883815408
0.5	교수님이	교수님이	[('교수',	[('교수',	1	0.8883815408
0.5	교수님도	교수님도	[('교수',	[('교수',	1	0.8883815408
0.5	학점 짜르	학점짜리	[('학점',	[('학점',	1	0.8883815408
0.5	출석 안뻐	출석 안ㅃ	[('출석',	[('출석',	1	0.8883815408
0.25	제주도 ?	제주도 ?	[('제주도	[('제주도	1	0.8883815408
0.25	중간기말	중간 기말	[('중간',	[('중간',	1	0.8883815408
0.25	진짜 너두	진짜 너두	[('진짜',	[('진짜',	1	0.8883815408
0.25	열심히 ㄸ	열심히 ㄸ	[('열심히	[('열심히	1	0.8883815408
0.25	말해 뭐할	말해 뭐할	[('말', 'No	[('말', 'No	1	0.8883815408
0.25	낚시에 공	낚시에 공	[('낚시',	[('낚시',	1	0.8883815408
0.25	교수님이	교수님이	[('교수',	[('교수',	1	0.8883815408
0.25	말이 필요	말이 필요	[('말', 'No	[('말', 'No	1	0.8883815408
0.25	최고의 수	최고의 수	[('최고',	[('최고',	1	0.8883815408
0.25	개꿀 그저	개꿀 그저	[('개꿀',	[('개꿀',	1	0.8883815408
0.25	승마를 처	승마를 처	[('승마',	[('승마',	1	0.8883815408
0.25	첫주 두번	첫 주 두	[('첫', 'No	[('첫', 'No	0	0.5018251538
0.25	첫 주동안	첫 주 동안	[('첫', 'No	[('첫', 'No	1	0.6230173707
0.25	출석만 본	출석만 본	[('출석',	[('출석',	1	0.4485501945
0.25	코로나 때	코로나 때	[('코로나	[('코로나	1	0.88774997
0.25	말 타러 오	말 타러 오	[('말', 'No	[('말', 'No	1	0.8919169307
0.25	개이득개	개이득개	[('개이득	[('개이득	0	0.5000865459
0.25	개쓰레기	개쓰레기	[('개', 'No	[('개', 'No	0	0.573099196
0.25	진짜 개꿀	진짜 개꿀	[('진짜',	[('진짜',	1	0.7480988503
0.25	코로나 싫	코로나 싫	[('코로나	[('코로나	0	0.4682018161
0.25	코로나 ㅌ	코로나 ㅌ	[('코로나	[('코로나	1	0.4532813132
0.25	교수님 ㄴ	교수님 ㄴ	[('교수',	[('교수',	1	0.8930651546
0.25	그저 꿀꿀	그저 꿀 ㄱ	[('그저',	[('꿀', 'No	1	0.6407115459
0.25	코로나 ㄸ	코로나 ㄸ	[('코로나	[('코로나	2	0.466904521
0.25	코로나때	코로나 ㄸ	[('코로나	[('코로나	2	0.4463536441
0.25	진짜 힐링	진짜 힐링	[('진짜',	[('진짜',	1	0.8882315159
0.25	시험기간	시험기간	[('시험',	[('시험',	2	0.4783061743
0.25	그냥 힐링	그냥 힐링	[('그냥',	[('그냥',	1	0.8036942482
0.25	개꿀 과목	개꿀 과목	[('개꿀',	[('개꿀',	2	0.4483948052
0.25	코로나의	코로나의	[('코로나	[('코로나	0	0.4548357725
0.25	수업번가	수업 번가	[('수업',	[('수업',	1	0.8611807227
0.25	코로나땜	코로나 뜸	[('코로나	[('코로나	1	0.7407232523
0.25	너무 아수	너무 아수	[('너무',	[('아쉬웠	1	0.884794116
0.25	이번학기	이번 학기	[('이번',	[('이번',	0	0.5097192526
0.25	코로나 꿀	코로나 꿀	[('코로나	[('코로나	2	0.46676296
0.25	지각만 안	지각만 안	[('지각',	[('지각',	1	0.8675150871
0.25	말너무 ㅅ	말 너무 ㅅ	[('말', 'No	[('말', 'No	1	0.7841132879
0.25	시간공지	시간 공지	[('시간',	[('시간',	0	0.5248690248

new score=
(기존평점*1/2)+(스케일링 된 label*예측확률)

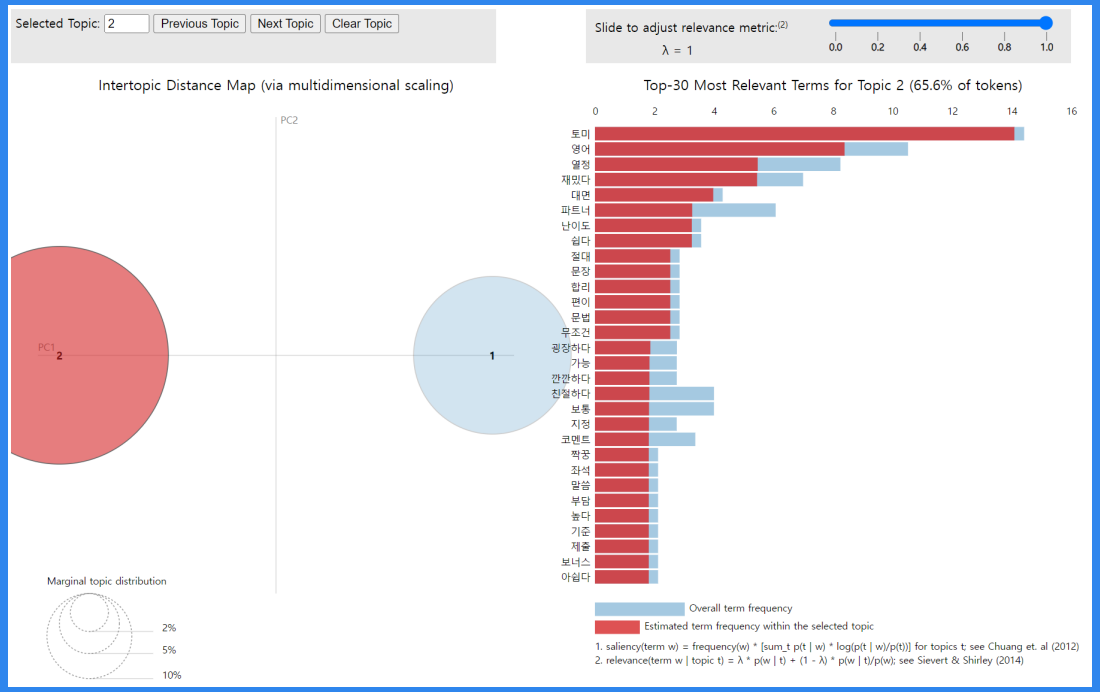
```
num_topics = 3
chunksize = 2000
passes = 20
iterations = 400
eval_every = None

temp = dictionary[0]
id2word = dictionary.id2token

model = LdaModel(
    corpus=corpus,
    id2word=id2word,
    chunksize=chunksize,
    alpha='auto',
    eta='auto',
    iterations=iterations,
    num_topics=num_topics,
    passes=passes,
    eval_every=eval_every
)
print(str(model.print_topics()))
top_topics = model.top_topics(corpus) #, num_words=20)

# Average topic coherence is the sum of topic coherences of all topics, divided by the number of topics.
avg_topic_coherence = sum([t[1] for t in top_topics]) / num_topics
print('Average topic coherence: %.4f.' % avg_topic_coherence)

from pprint import pprint
pprint(top_topics)
#시각화
#lda_visualization = gensimvis.prepare(model, corpus, dictionary, sort_topics=False)
#pyLDAvis.save_html(lda_visualization, 'LDA_RESULT.html')
pyLDAvis.enable_notebook()
vis = pyLDAvis.gensim_models.prepare(model, corpus, dictionary)
pyLDAvis.display(vis)
```

LDA

명사, 동사, 형용사만 사용

토픽 모델링을 통해 강의력, 교수 인성 등
의 키워드가 있을 시
긍정/부정/중립 라벨링

강의력, 교수인성 feature 추가
(기존 feature 팀플횟수 과제횟수 학점부
여 시험횟수)




```
insung_bad = ['권위', '최악', '파탄','꼰대']
insung_good = ["친절하다","친절", "유쾌하", "유쾌하다","열정","착하다","나이스"]
lecture_bad= ["지루하다",'졸리다','따분하다']
lecture_good= ['재밌다',"재미있다",'유익하다', '꿀장하다', '피드백', '배려', '유익']
```

LDA

명사, 동사, 형용사만 사용

토픽 모델링을 통해 강의력, 교수 인성 등
의 키워드가 있을 시
긍정/부정/중립 라벨링

lecture_faculty	insung
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	0.5
1	0.5
1	0.5
1	0.5
1	0.5
1	0.5
1	0.5
1	0.5
1	0.5
1	0.5

교수인성 feature 추가
re 팀플횟수 과제횟수 학점부
여 시험횟수)

```python
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from bs4 import BeautifulSoup
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from gensim.summarization.summarizer import summarize
np.random.seed(seed=0)
```

```python
data['extractive'] = ''
for i in data.index :
  try:
    data['extractive'][i] = summarize(data['lecture_reviews'][i], word_count=10)
  except ValueError:
    data['extractive'][i] = data['lecture_reviews'][i]

  if data['extractive'][i] == '':
      data['extractive'][i] = data['lecture_reviews'][i]
```

lecture_reviews
발표 많이 하면 에이쁠 받습니다. 본인이 발표 하는걸 좋아하지 않는다면 점수 받기 ...
학기 중에 과제 꽤 많았는데 그거를 바탕으로 꾸준히 복습해 두시면 시험을 잘 볼 수...
농구좋아하시는분이면 무조건 추천필기와 실기로 시험봅니다. 훈련이 조금 힘들 수 있지...
시험도 크게 어렵진 않고 인생 살아가는데 큰 도움이 됩니다. 꼭꼭꼭 들으세요. 우리...
수강신청 성공하고 가서 볼링만 치면 나옴강 개꿀강의 무엇보다 교수님이 너무 좋으심....
...
수업 매우 좋아요. 정보융합학부면 졸업전에는 꼭 들어야 하는 수업. 수업 퀄리티는 ...
수업이 매우 좋아요. 정보융합학부라면 필수로 수강해야 합니다. 가져가는게 많습니다....
중간고사는 레포트로 대체 기말고사도 레포트로 대체레포트 가이드라인은 교수님이 제시해...
내용도 유익하고 편하게 들을 수 있는 강의라 너무 좋았습니다. 우선 격주 비대면 수...
학기 데사개 수업 열심히 들은 학생이라면 잘 따라올 수 있어요. 연장선상의 느낌입니...

extractive
그리고 가면 갈수록 내용이 어려워져서 초반에 발표를 많이 하면 좋아요.
학기 중에 과제 꽤 많았는데 그거를 바탕으로 꾸준히 복습해 두시면 시험을 잘 볼 수...
시험은 중간 필기 기말 실기 한번씩 봤는데 쉽고 학점 잘 주십니다
시험은 단답형 문제 위주로 나오고 난이도도 쉬워서 시험 하루 전날 빡세게 공부해서 ...
교수님이 자세도 잘 알려주심시험 문제도 다 알려줌.
...
수업 매우 좋아요. 정보융합학부면 졸업전에는 꼭 들어야 하는 수업. 수업 퀄리티는 ...
수업이 매우 좋아요. 정보융합학부라면 필수로 수강해야 합니다. 가져가는게 많습니다....
중간고사는 레포트로 대체 기말고사도 레포트로 대체레포트 가이드라인은 교수님이 제시해...
우선 격주 비대면 수업이었고 수강생이 적어 비교적 편안한 분위기였습니다.
교수님 매우 열정적이여서 노력만한다면 많은것을 배워갈 수 있는 수업이라고 생각합니다...

텍스트 요약

Gensim 라이브러리 활용

↓

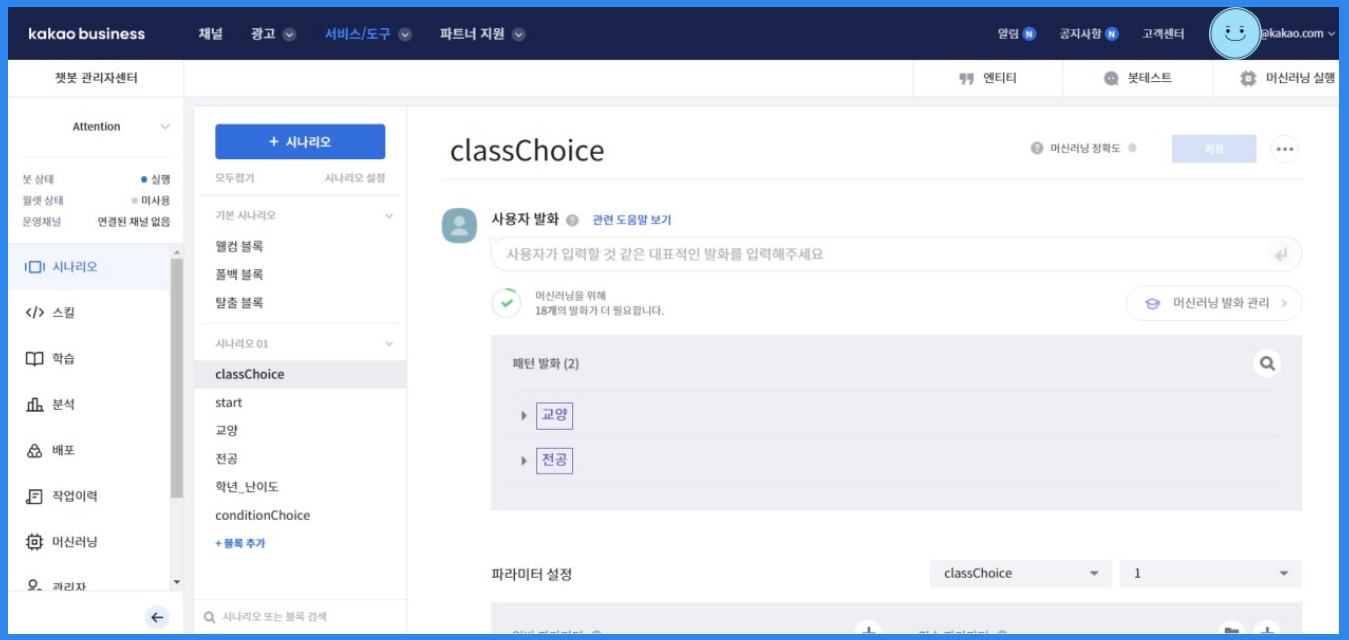
추출요약을 통해
강의평들을 20단어 이내로 요약시킴

↓

챗봇에 강의 추천과 함께 제공

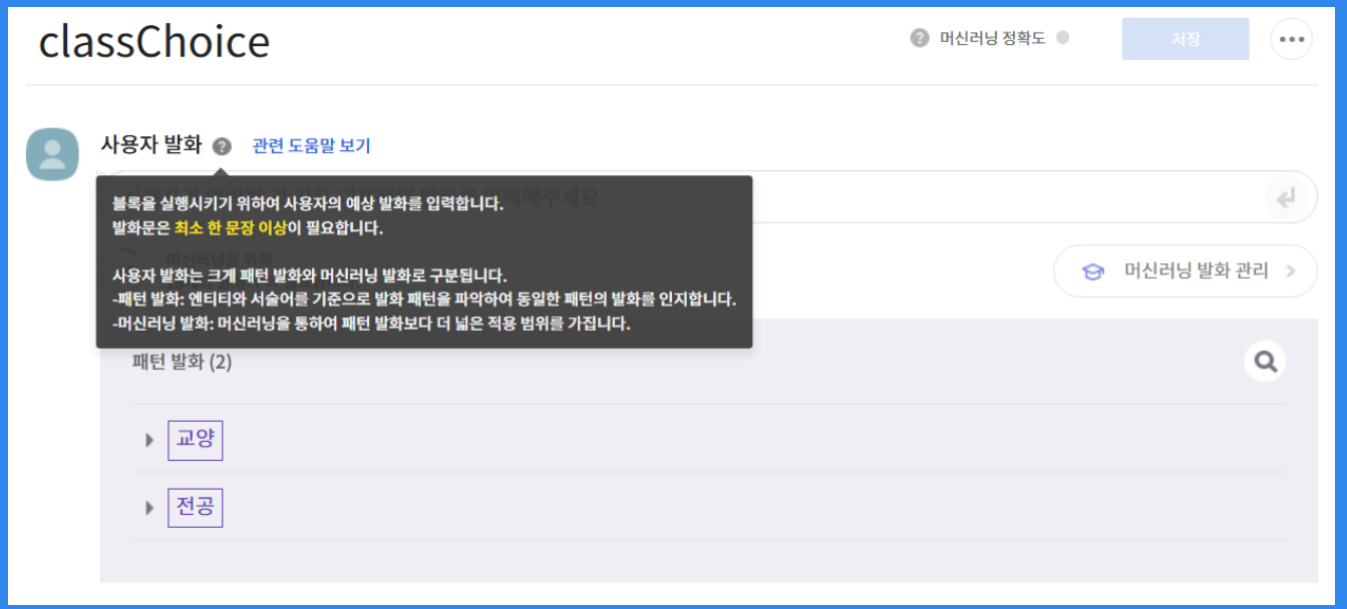
챗봇

api설명


kakao business
채널
광고
서비스/도구
파트너 지원
챗봇 관리자센터
엔티티
봇테스트
머신러닝 실행
Attention
+ 시나리오
classChoice
사용자 발화
사용자가 입력할 것 같은 대표적인 발화를 입력해주세요
패턴 발화 (2)
교양
전공
파라미터 설정
classChoice
1
시나리오
스킬
학습
분석
배포
작업이력
머신러닝
start
conditionChoice

챗봇

api설명



사용자 발화

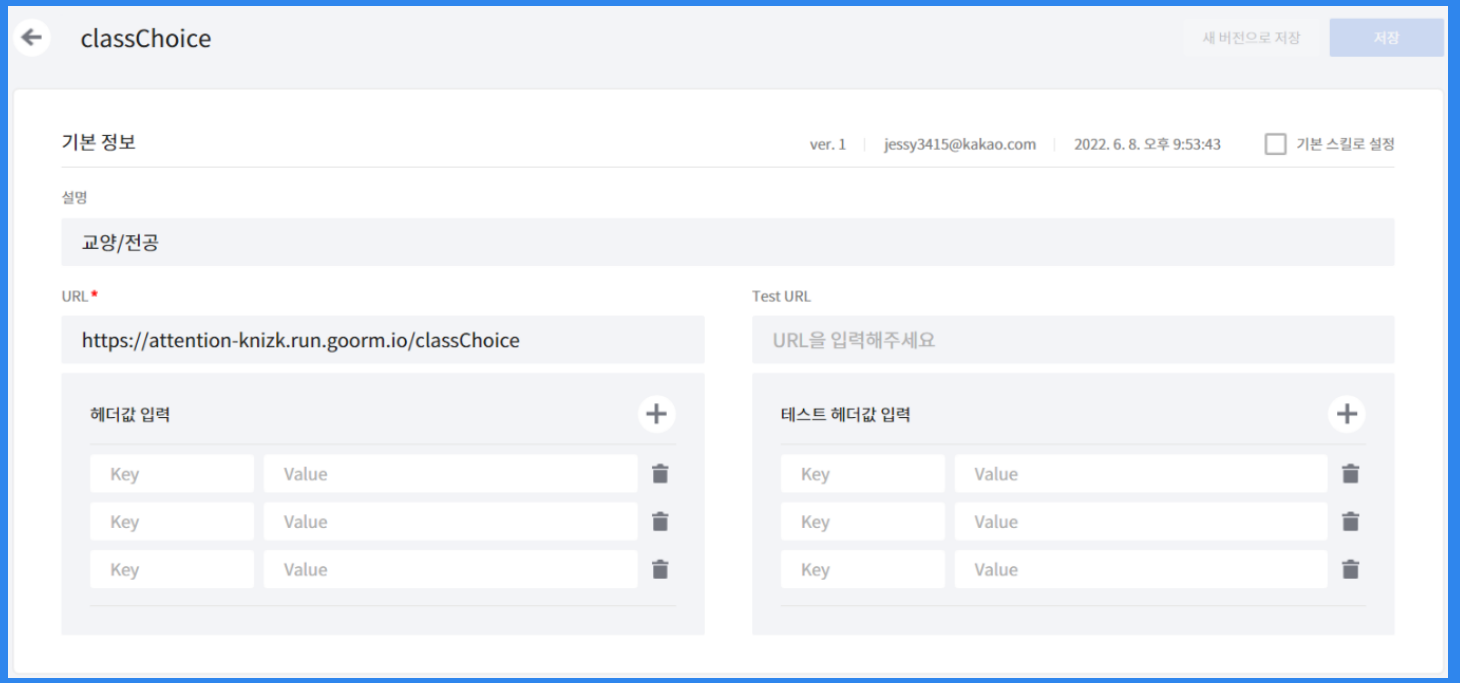
챗봇

api설명


나의 엔티티
시스템 엔티티
jungong
저장
대표 엔트리를 입력한 뒤 엔터를 눌러주세요
18
화학공학과
화공
동의어 입력 후 엔터를 눌러주세요
17
전자통신공학과
전통과
전통
전자통신
전자통신공학
동의어 입력 후 엔터를 눌러주세요
16
정보융합학부
정융
정보융합
동의어 입력 후 엔터를 눌러주세요
15
전자재료공학과
전재
전자재료
전자재료공학
동의어 입력 후 엔터를 눌러주세요
14
전자융합공학과
전융
전자융합공학
동의어 입력 후 엔터를 눌러주세요
13
스포츠융합과학과
스융과
스융
스포츠융합
동의어 입력 후 엔터를 눌러주세요
12
컴퓨터정보공학부
컴정공
컴퓨터정보공학
동의어 입력 후 엔터를 눌러주세요

챗봇

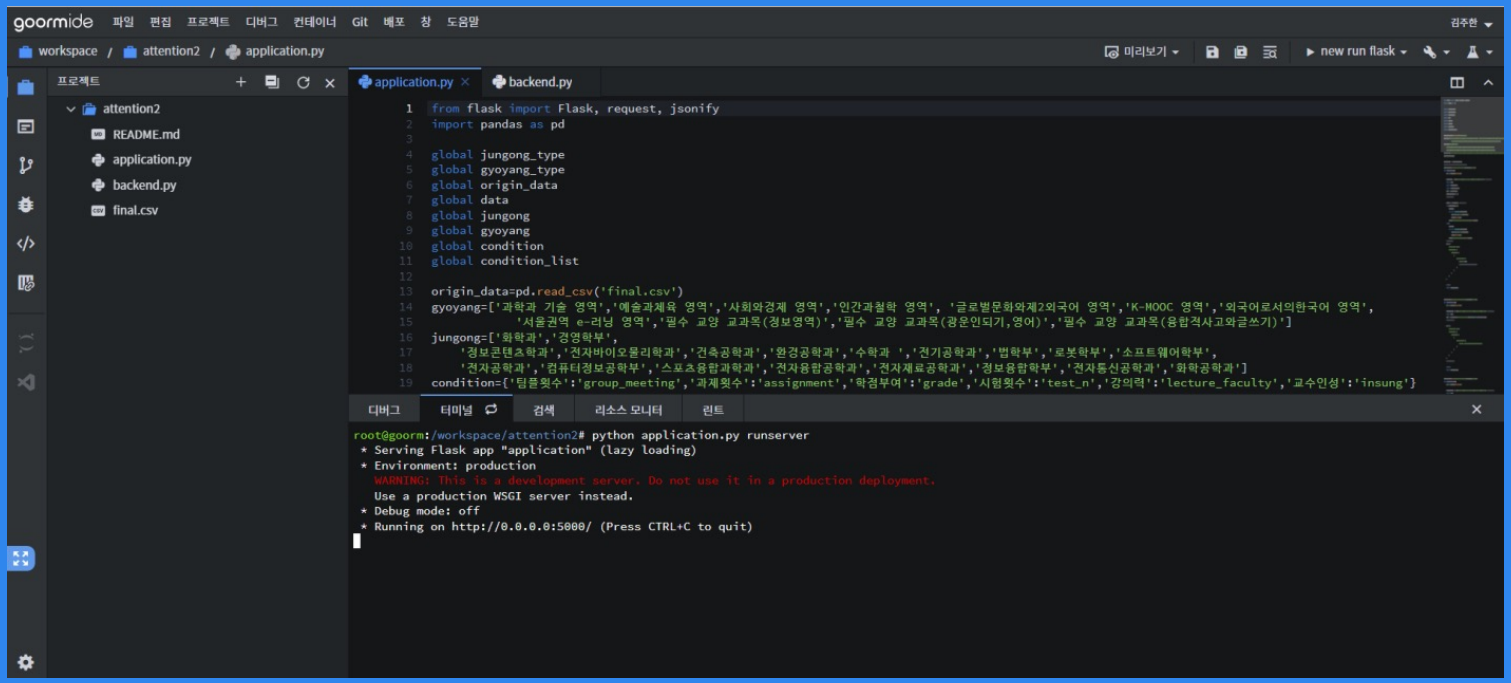
api설명


classChoice
새 버전으로 저장
저장
기본 정보
ver. 1
jessy3415@kakao.com
2022. 6. 8. 오후 9:53:43
기본 스킬로 설정
설명
교양/전공
URL*
https://attention-knizk.run.goorm.io/classChoice
Test URL
URL을 입력해주세요
헤더값 입력
테스트 헤더값 입력
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value
Key
Value

스킬

챗봇

api설명


goormide 파일 편집 프로젝트 디버그 컨테이너 Git 배포 창 도움말
workspace / attention2 / application.py
new run flask
프로젝트
attention2
README.md
application.py
backend.py
final.csv
application.py
backend.py
from flask import Flask, request, jsonify
import pandas as pd
global jungong_type
global gyoyang_type
global origin_data
global data
global jungong
global gyoyang
global condition
global condition_list
origin_data=pd.read_csv('final.csv')
gyoyang=['과학과 기술 영역','예술과체육 영역','사회와경제 영역','인간과철학 영역', '글로벌문화와제2외국어 영역','K-MOOC 영역','외국어로서의한국어 영역',
'서울권역 e-러닝 영역','필수 교양 교과목(정보영역)','필수 교양 교과목(광운인되기,영어)','필수 교양 교과목(융합적사고와글쓰기)']
jungong=['화학과','경영학부',
'정보콘텐츠학과','전자바이오물리학과','건축공학과','환경공학과','수학과 ','전기공학과','법학부','로봇학부','소프트웨어학부',
'전자공학과','컴퓨터정보공학부','스포츠융합과학과','전자융합공학과','전자재료공학과','정보융합학부','전자통신공학과','화학공학과']
condition={'팀플횟수':'group_meeting','과제횟수':'assignment','학점부여':'grade','시험횟수':'test_n','강의력':'lecture_faculty','교수인성':'insung'}
디버그
터미널
검색
리소스 모니터
린트
root@goorm:/workspace/attention2# python application.py runserver
* Serving Flask app "application" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

챗봇

추천시스템

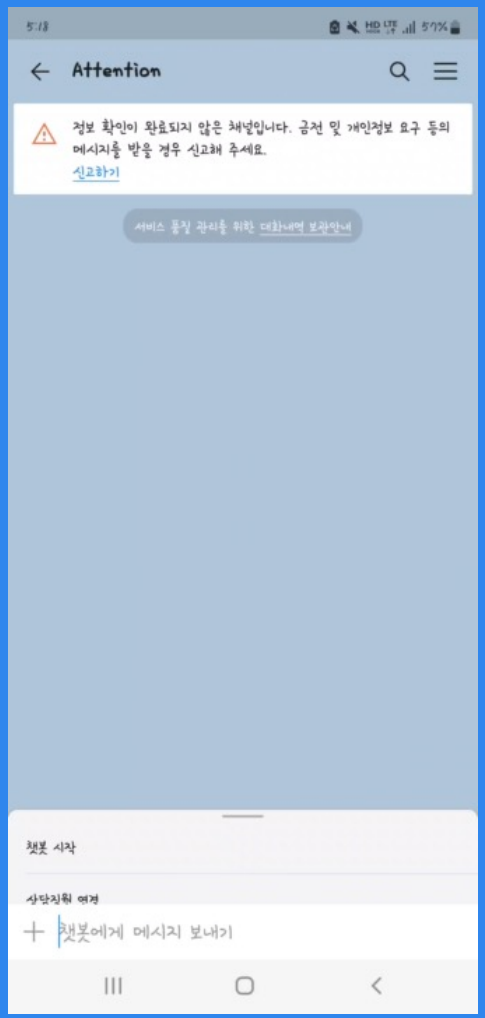
```
data[condition_list[0]]=data[condition_list[0]]*4
data[condition_list[1]]=data[condition_list[1]]*2
data[condition_list[2]]=data[condition_list[2]]*1

data['r_score']=(data['group_meeting']+data['assignment']+data['grade']+data['test_n']+data['lecture_faculty']+data['insung'])
                *(data['score']/2+data['f_score'])/9
dic=data.groupby('lecture_code')['r_score'].mean()
dic = dic.to_dict()

#첫번째 큰값
max_key1 = max(dic, key=dic.get)
recommend1=data[data['lecture_code']==max_key1]
recommend11 = list(recommend1['professor_name'])[0]
recommend12 = list(recommend1['lecture_name'])[0]
recommend13 = list(recommend1['extractive'])[0]
#두번째 큰값
max_key2 =sorted(list(dic.keys()))[1]
print(max_key2)
recommend2=data[data['lecture_code']==max_key2]
recommend21 = list(recommend2['professor_name'])[0]
recommend22 = list(recommend2['lecture_name'])[0]
recommend23 = list(recommend2['extractive'])[0]

#세번째 큰값
max_key3 =sorted(list(dic.keys()))[2]
print(max_key3)
recommend3=data[data['lecture_code']==max_key3]
recommend31 = list(recommend3['professor_name'])[0]
recommend32 = list(recommend3['lecture_name'])[0]
recommend33 = list(recommend3['extractive'])[0]
```

학생이 선택한 우선항목
순서에 따라 해당하는 변수에
가중치를 부여

최종 추천 점수 =
(범주형 score점수들 합)*
감정분석을 통한 rescoring 점수

최종적으로 점수가 가장 높은
2개의 강의를 추천

rescore
feature -> 최종 output계산
[논문]평점과 리뷰 텍스트 감성분석을 결합한 추천시스템 향상 방안 연구 (kisti.re.kr) -2019 참고



챗봇

시연

단계 1. 전공과 교양 선택

단계 2. 전공 혹은 교양의 세부 영역 선택

단계 3. 강의를 선택할 때 중요시하는 요소를 중요한 순서대로 3가지 선택

기대효과

1

학생들의 강의에 대한 객관적인 평가 제공

감성분석 / LDA를 통해
에브리타임 강의평 텍스트로부터
객관적인 평가를 도출하여,
학생들이 강의를 신청/정정할 때
에브리타임보다 더 객관적인 평가를 제공한다.

2

강의 추천 시스템

학생들이 수강할 강의를 선택할 때,
학생들 개개인이 중요시하는 조건들에 맞춰
강의를 추천해주는
강의 추천 시스템을 제공한다.

한계점 및 추후 보완점

1 라벨링된 데이터 부족

2 LDA 파라미터 조정 및 키워드 추가

3 사용자 발화 등록 부족 -> 머신러닝 기능 사용 불가

4 서버 문제

THANK YOU

Attention