구해줘, 나의 자료

TEAM 2조 알파코 규장각

안희성, 이혜진, 전재현, 차재경

목차

01. 프로젝트 개요 – 목적 및 기대효과

대학교에서 처음 교양수업 관련 과제를 하는 학부1~2학년 학생을 대상으로,
그들이 우리에게 과제를 제시하면, 그것과 유사한 서적을 찾아주는 시스템

처음 교양수업 관련하여 과제를 하는 학부1~2학년 학생들에게 과제에 쉽게 접근할 수 있는 자료를 제시하며, 그들의 과제의 방향성을 잡는데 도움이 될 것이다.

01. 프로젝트 개요 - 내용

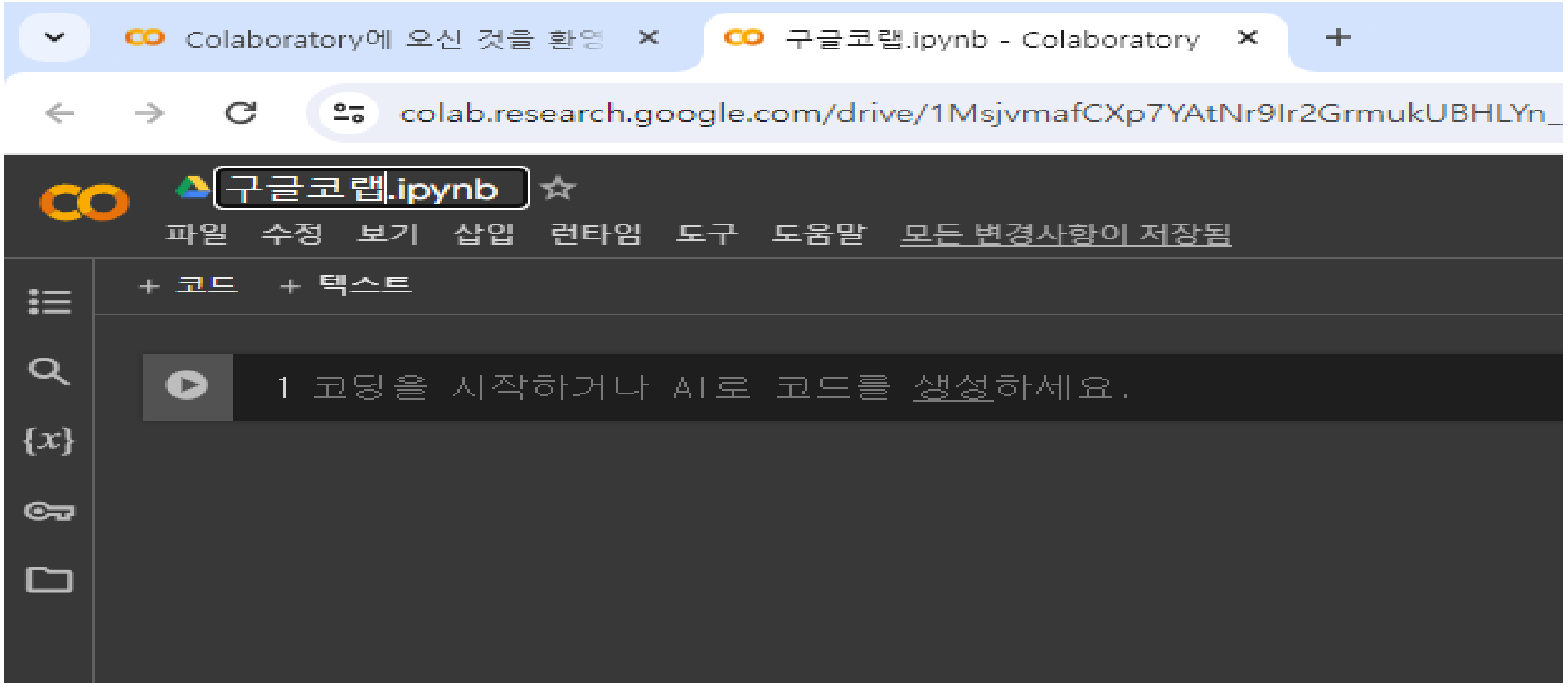
➡ 데이터 수집
- 크롤링 (교보문고 분야별 베스트 셀러)

➡ 데이터 전처리
- 크롤링한 데이터의 누락값과 이상치 제거
- 한나눔 형태소 분석기를 통해서 토큰화, 불용어 제거 및 명사화

➡ TextRank
- 소개글과 목차를 TextRank를 통해서 '목차'와 '소개글' 키워드와 스코어를 추출(제목의 명사는 키워드로 간주)
- 사용자로부터 인풋값을 받아 TextRank를 이용하여 키워드를 추출
- 사용자로부터 받은 값을 통해 추출한 키워드가 우리가 만든 데이터베이스에 가장 겹치는 책이 유사한 책이다 라고 가정 > 이 가정을 통해 유사도를 추출할 방법론으로 자카드 유사도 채택

➡ Jaccard Similarity
- 키워드 점수를 추출한 '목차', '소개글'에 대해서는 가중치를 적용하는 weight Jaccard / 제목에 대해서는 Jaccard를 통해 유사도를 계산, 이를 기반으로 유사도 높은 책을 추천

01. 프로젝트 개요 – 활용 도구(개발환경)



02. 프로젝트 팀 구성 및 역할

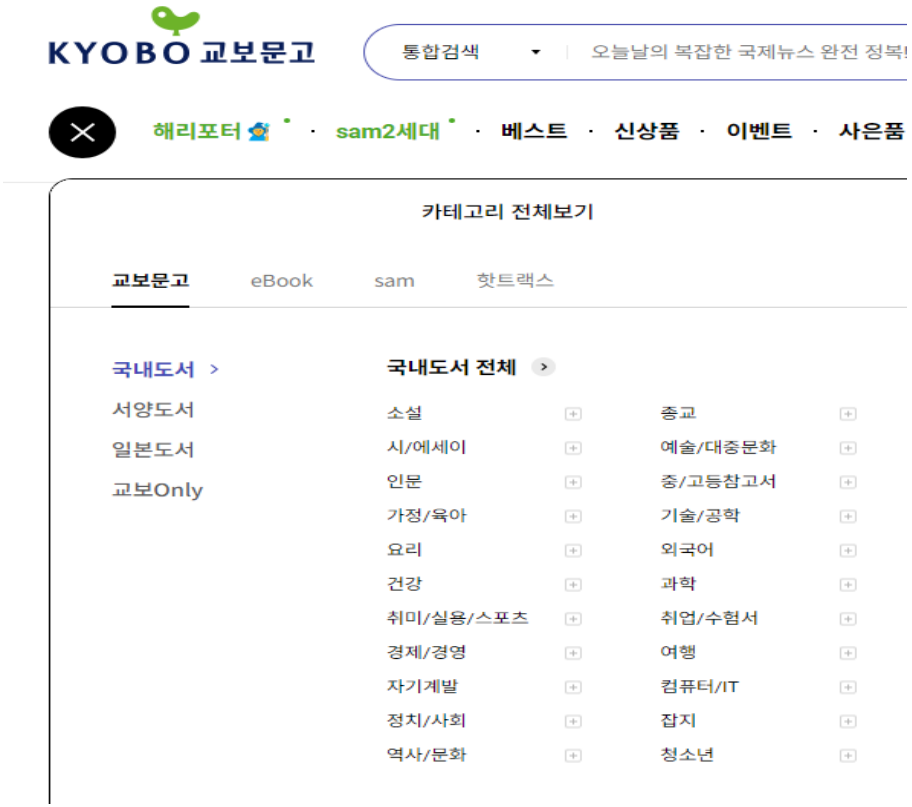
훈련생	역할	담당 업무
차재경	*팀장*	▶ *기획총괄 및 데이터 수집*
안희성	*팀원*	▶ *텍스트 크롤링, 전처리, 모델링, 서비스 구축 및 배포*
전재현	*팀원*	▶ *데이터 정제 및 정규화*
이혜진	*팀원*	▶ *텍스트 전처리 및 분류*

03. 프로젝트 수행 절차 및 방법

구분	기간	활동	비고
사전 기획	▶ 12/7 (목) ~ 12/8(금)	▶ 프로젝트 기획 및 주제 선정 ▶ 기획안 작성	▶ 아이디어 선정
데이터 수집	▶ 12/9(토) ~ 12/12(화)	▶ 필요 데이터 및 수집 절차 정의 ▶ 외부 데이터 수집	▶ 교보문고 데이터
데이터 전처리	▶ 12/13(수) ~ 12/15(금)	▶ 데이터 정제 및 정규화	
모델링	▶ 12/16(토) ~ 12/17(월)	▶ 모형 구현	
서비스 구축	▶ 12/18(일)	▶ 모델 수정 및 배포	▶ 최적화, 오류 수정
총 개발기간	▶ 12/7(월) ~ 12/18(일) (총 12일)	-	-

04. 프로젝트 수행 경과 – 데이터 수집

- 크롤링 : 웹페이지를 그대로 가져와서 거기서 데이터를 추출하는 방식




```
[ ]  1 from selenium import webdriver
     2
     3 options = webdriver.ChromeOptions()
     4 options.add_argument("--headless") # GUI 화면을 띄울지 유무. 보통 코딩 완료후에는 필요없기 때문에 끈다
     5 options.add_argument('--disable-dev-shm-usage') # 크롬 메모리 제한 푸는것
     6 options.add_argument("--no-sandbox")
     7 user_agent = "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0"
     8 options.add_argument('user-agent=' + user_agent)
     9
    10 driver = webdriver.Chrome(options=options)
```

```
[ ]  1 from tqdm import tqdm
     2
     3 book_info = []
     4
     5 for n in tqdm(kyob['판매상품ID'][746:]) :
     6     url = f'https://product.kyobobook.co.kr/detail/{n}' # 책 페이지
     7     driver.get(url)
     8     time.sleep(3)
     9
    10     try :
    11         genre = driver.find_element(By.CLASS_NAME, 'category_list_item').text # 책 장르
    12         contents = driver.find_element(By.CLASS_NAME, 'intro_bottom').text # 책 소개
    13         index = driver.find_element(By.CLASS_NAME, 'book_contents_item').text # 책 목차
    14         book_info.append([n, genre, contents, index])
    15
    16     except :
    17         book_info.append([n])
    18         continue
```

```
[ ]  1 book = pd.DataFrame(book_info)
```

```
[ ]  1 book.columns = ['판매상품ID', '분야', '책 소개', '책 목차']
```

```
[ ]  1 book = book.reset_index()
     2 del book['index'] # 리셋하면서 새로생긴 index열 삭제
     3 book
```

```
[ ]  1 book_final = pd.concat([kyob, book], axis= 1)
```

```
[ ]  1 del book_final[13] # 중복되는 판매상품ID열 삭제
     2 book_final
```

```
[ ]  1 book_final.columns = ['순위', '판매상품ID', '상품코드', '상품명', '저자', '출판사', '분야', '발행(출시)일자', '정가',
     2                       '판매가', '할인율', '적립율', '적립예정포인트', '분야', '책 소개', '책 목차']
```

```
[ ]  1 book_final.to_pickle('book_data_final.pkl')
```

04. 프로젝트 수행 경과 – 데이터 전처리

- 누락값과 이상치 제거
- 한나눔 형태소 분석기를 통해서 토큰화, 불용어 제거 및 명사화

누락값 : 누락된 값
이상치 : 이상한 값
토큰화 : 의미있는 최소 단위인 형태소로 나누는 작업
불용어 : 중요하지 않는 단어
명사화 : 텍스트 데이터에서 명사를 추출하는 작업

Hannanum

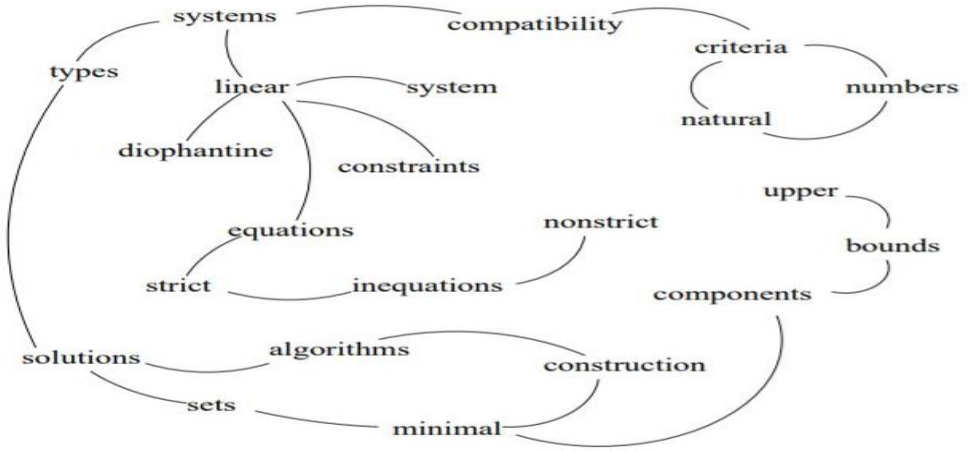
```
[ ]  1 from konlpy.tag import Hannanum
     2 import re
     3
     4 Hannanum = Hannanum()
     5
     6 def Name(string) :
     7     string = ' '.join(re.findall('[가-힣]+|[a-z]+', string.lower())) # 특수문자들 제거 및 모든 영어 소문자 변환
     8     string = re.sub(' +', ' ', string) # 공백을 하나로
     9     string = Hannanum.nouns(string)
    10     return string
    11
    12 a['상품명_Hannanum'] = a['상품명'].apply(lambda x: Name(x))
```

```
[ ]  1 a[['상품명_Hannanum']]
```

	상품명_Hannanum
0	[에디토리얼, 씽킹]
1	[창조적, 행위, 존재, 방식]
2	[방구석, 미술관]
3	[홀리데이, 뮤지엄]
4	[포, 어스]
...	...
10391	[언어, 이해]
10392	[로봇, 의, 인류학]
10393	[내일, 로봇왕, 충격, 인터넷, 방송]

04. 프로젝트 수행 경과 - TextRank

TextRank : 텍스트 내 문장, 단어의 중요도를 계산하여 중요도가 높은 순으로 요약하는 방법론



위 이미지는 상단에 주어진 글에 대해 텍스트 간 관계를 나타낸 그래프로 그린 샘플 이미지이다. 내용과 그래프를 비교해보면 각 문장에서 단어 간의 관계를 선으로 연결하는 모습을 살펴볼 수 있는데, 순위는 다음과 같은 수식에 의해 매긴다고 한다.

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

```
    if len(set(string.split())) < words_n :
        words_n = len(set(string.split())) # 단어 갯수가 적을 경우 뽑을 키워드 갯수를 낮춘다

    string = keywords.keywords(string, words = words_n, scores = True) # 키워드 뽑기

    word = []
    score = []
    for i, j in string :
        i = ''.join(i).split()
        word.append(i)
    word_list = []
    for i in word :
        for j in i :
            word_list.append(j)
    for i in word_list : # 단어들 1개씩
        for ind, j in enumerate(word) :
            if i in j :
                score.append([i, round(string[ind][1], 2)])
    if len(score) > 5 :
        score = score[:5]

    return score
```

```
def DicTrans(keyword) : # 키워드들을 딕셔너리 형태로 변환

    intro = [i[0] for i in keyword]

    dic = {}
    for ind, i in enumerate(intro) :
        dic[i] = keyword[ind][1]
```

04. 프로젝트 수행 경과 - Jaccard Similarity

Jaccard coefficient

(Jaccard index 및 Jaccard similarity) 라고도 함

두 집합 사이의 similarity를 측정하는 방법 중 하나.

0~1 사이의 값을 가지며 두 집합이 동일하면 1의 값을 가지고, 공통 원소가 없으면 0을 가진다.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

```python
def JarccardSimilarity_title (input_key, data_key) :
    score = 0

    in_keys = set(input_key.keys())
    da_keys = set(data_key)
    same = in_keys & da_keys
    for i in list(same) :
        score += (input_key[i] * 1.7) + 1.3
    a = set.union(in_keys, da_keys)
    return score / len(a)
```

```python
def Recommendation(input_text, data_set, Book_Count) : # 사용자가 넣은 문장을 전처리 하여, 데이터 셋과 유사도 분석 실행, 상위 n개 출력

    if type(input_text) != dict :
        input_text = preprocess(input_text)
        input_text = DicTrans(input_text)

    intro_score = data_set['소개글_키워드'].apply(lambda a : JarccardSimilarity(input_text, a))
    index_score = data_set['목차_키워드'].apply(lambda a : JarccardSimilarity(input_text, a))
    title_score = data_set['상품명_Hannanum'].apply(lambda a : JarccardSimilarity_title(input_text, a))

    intro_sim_list = intro_score.sort_values()[-1 : -(Book_Count + 1) : -1] # Book_Count 갯수만큼 책을 뽑아준다
    index_sim_list = index_score.sort_values()[-1 : -(Book_Count + 1) : -1]
    title_sim_list = title_score.sort_values()[-1 : -(Book_Count + 1) : -1]

    for title , t in zip(['소개글 기반 유사도', '목차 기반 유사도', '제목 기반 유사도'], [intro_sim_list, index_sim_list, title_sim_list]) :
        print('==========================================================================================')
        print(f'{title}')
        print('==========================================================================================')
        print('==========================================================================================')
        for key, score in zip(t.keys(), t) :
            if score == 0 :
                break
            for i in ['상품명','저자', '소개글', '목차', '분야', '정가'] :
                print(f'{i} : {data_set.loc[key][i]}')
            print(f'유사도 : {score}')
            print('==========================================================================================')
```

05. 자체 평가 의견

데이터 절대적인 양 늘리기
벡터화를 시키지 않아서, 문맥상 의미가 없더라
실행파일 만들고 싶었는데, 시간 및 기술 부족으로 실패
하이퍼 파라미터 키워드 개수 변경해서 성능 향상이 필요함
가중치의 대한 근거가 부족