

DS데이터톤

2024

덕성여자대학교 도서추천시스템 제안

DSAP

정윤주, 김소연, 김언영, 박주현

DS데이터톤

2024

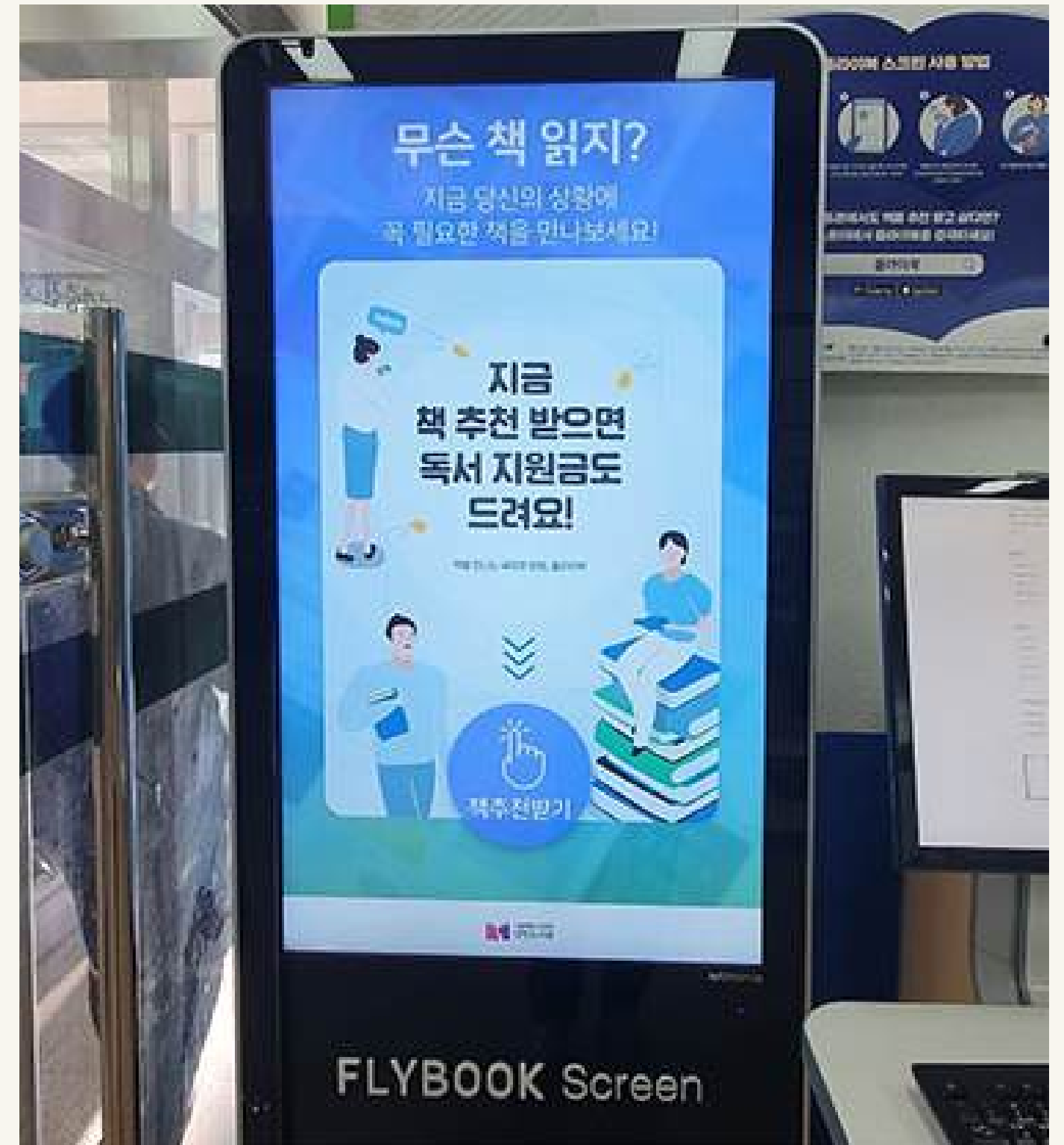
Contents.

상황분석	01
문제정의	02
프로젝트 소개	03
학습 데이터 생성	04
추천 시스템 학습 방법	05
알고리즘 고도화 방안	06



도서관 데이터 활용의 확산

도서관 데이터 활용 서비스는 장서 활용도와 이용자 만족도를 효과적으로 향상시키고 있다. 미대출장서 분석 시스템을 활용한 '숨은 도서 찾기' 프로그램은 대출되지 않던 도서의 활용도를 높였다. 예를 들어, 129권의 미대출도서가 새롭게 대출되었으며, 사서추천도서 대출 건수도 0건에서 14건으로 증가했다. 연령별 대출장서 순위 분석을 통한 수요 중심 수서 정책은 장서 구성 효율성을 높이는 데 기여했다(온정미 외, 2020).



플라이북 스크린 모습(<https://opengov.seoul.go.kr/mediahub/21472090>)

독서 프로파일링

소장자료검색

[🏠](#) > [My ELIS](#) > [맞춤서비스](#) > [독서 프로파일링](#)

이 ** 님의 독서 취향을 반영한 추천 자료 ?

책 ☒ 전자책

최근 1년 ▼

조회

찍지 못한 순간에 관하여
여: 글로 쓴 사진...

전문가와 강적들: 나도
너만큼 알아

사람, 장소, 현대

모두 거짓말을 한다:
EVERYBODY LIES

선량한 차별주의자

시간과 물에 대하여

◀

전체

종류

월학/심리학

종교

사회과학

언어

자연과학

기술과학

▶

이 ★★ 님의 관심 분야 신착도서 ?

전체 신착도서

이 ★★ 님의 관심 저자/시리즈 신착도서 ?

[illegible]

관심 저자



선택 키워드

필수 여행스페어
bauer 화학 이탈리아 극작가
생물학적 기능 박완서 소설전집
제주도관광협회
제주도 버스
인기 여행지 화학 반응
코스별 정보

대학 도서관의 AI 추천 서비스

이화여대 '독서 프로파일링'(<https://news.unn.net/news/articleView.html?idxno=506906>)

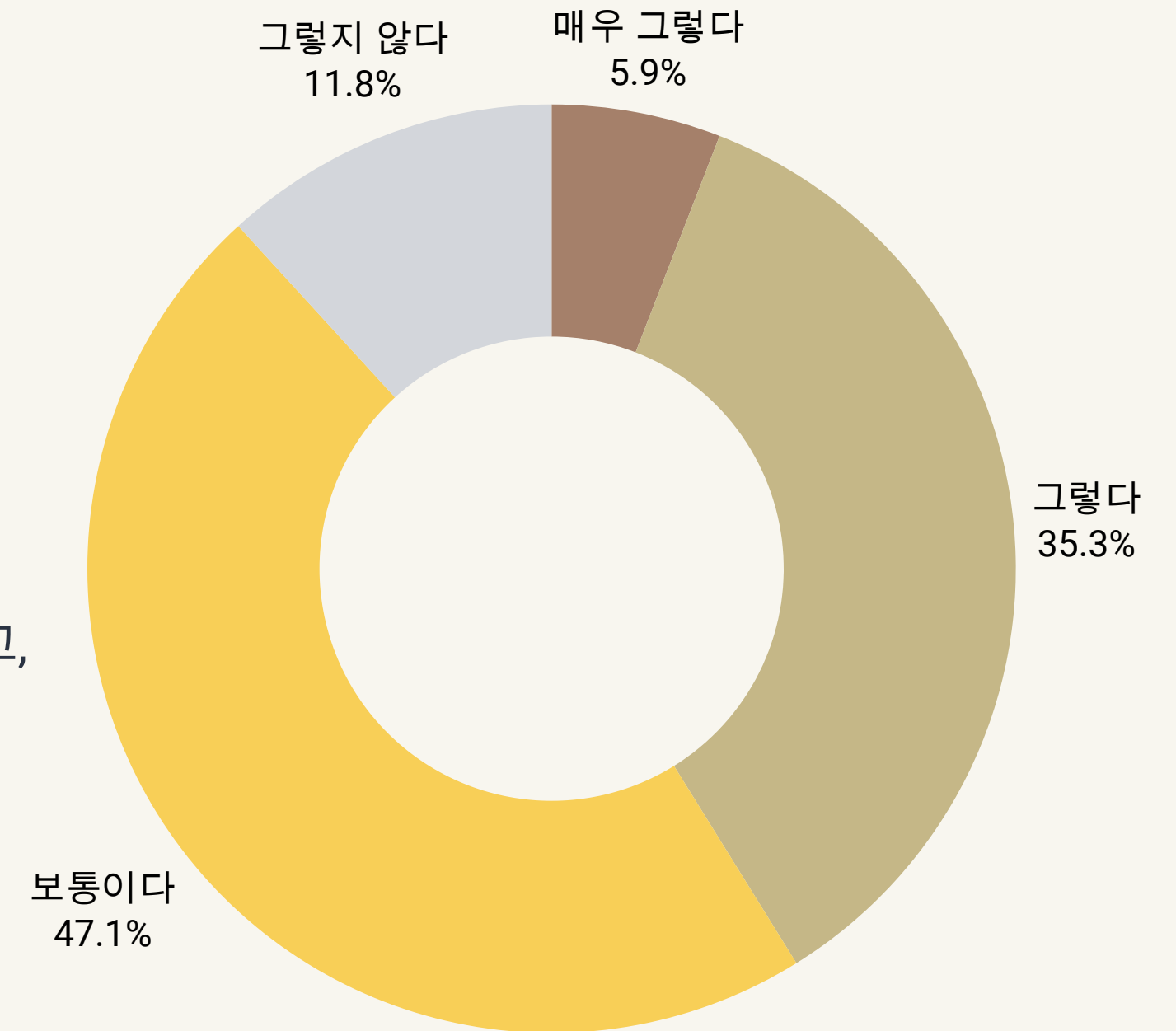
서울대 '대출이력 기반 맞춤형 보고서'(https://now.snu.ac.kr/47/2/1438?utm_source)

맞춤형 도서 추천기능이 필요하다.

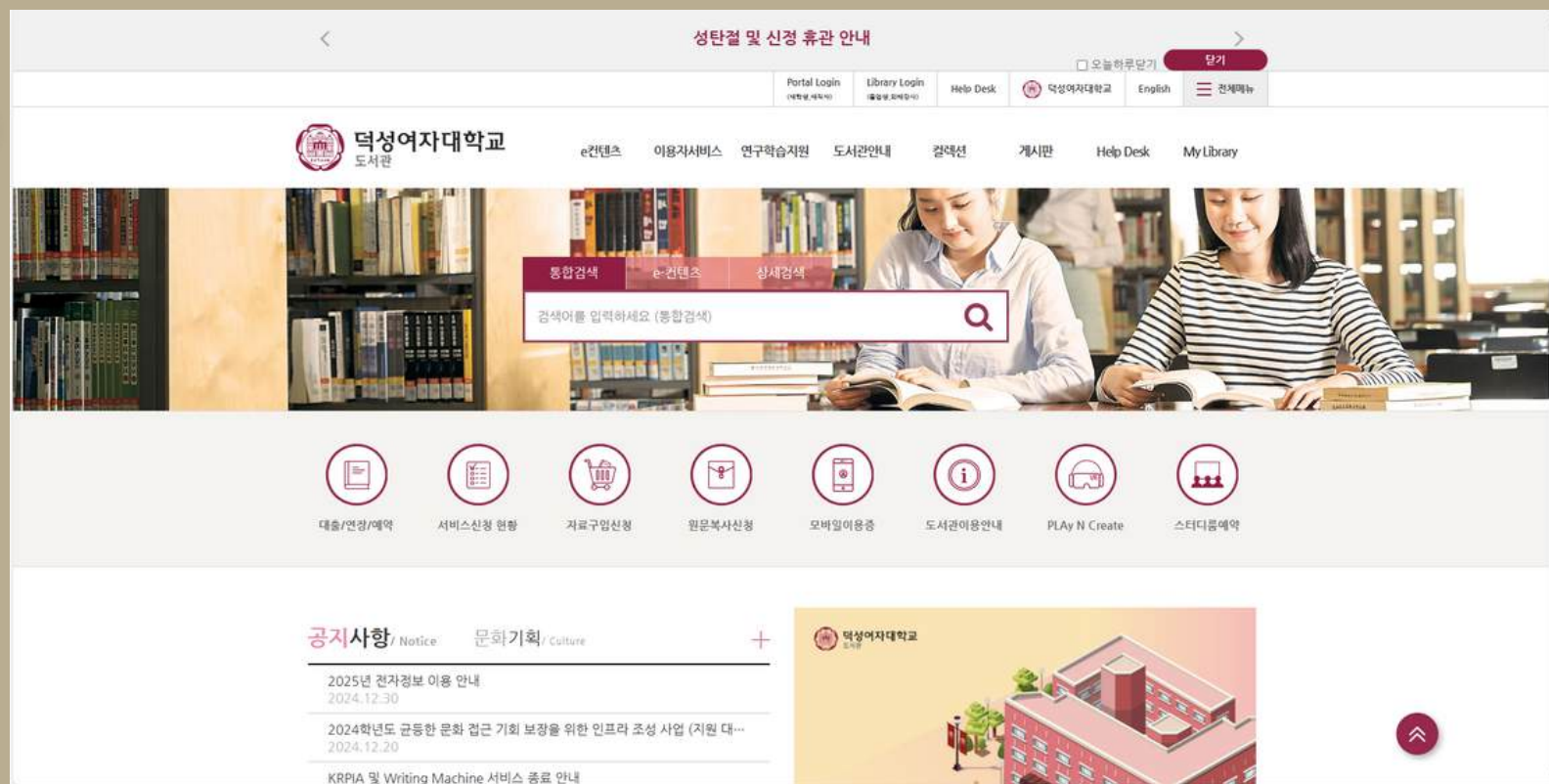
도서관 웹/앱 만족도 조사 결과,

총 34명(재학 31명, 휴학 3명)의 설문 참여자 중 16명의 학생이 '보통이다'라고 응답하였고, 14명의 학생이 필요하다고 답하였다.

또, 맞춤형 도서 추천 기능이 동기부여를 줄 수 있을 것이라고 생각하냐는 질문에, 25명이 '그렇다'와 '매우 그렇다'라고 답하였다.



새로운



덕성여자대학교 도서관
웹페이지

추천시스템

대학생의 학습 요구를 고려한 맞춤형 시스템 개발이 필요하며, 도서 대출 기록과 성향을 분석해 개인화된 추천을 제공하면 대학도서관의 이용률과 만족도를 증대시킬 수 있다(홍연경 외, 2021) 또한, AI 교육 콘텐츠 추천 시스템에 대한 연구 결과에서도 대학생들의 사용 의사가 높아 맞춤형 시스템에 대한 수요가 확인되었다(김성훈 외, 2022).

이에 덕성여자대학교 도서관에서 활용할 수 있는 추천시스템을 제안한다.

학습 데이터 생성

데이터 정제

- 서지 유형 필터링
 - 제공받은 보유 서지 목록 데이터에서, '이북'과 'DVD' 등의 서지유형을 제외하고, '단행본' 데이터만 남겨 학생들이 대여하는 실제 도서 목록만을 대상으로 데이터를 정제
- 언어 필터링
 - 인코딩이 어려운 언어는 제외하고, '언어' 값이 'Korean'과 'English'인 도서만을 선택하여 대상 데이터를 한정

데이터 설정

- 학생수
 - 500명의 학생을 대상으로 설정하였고, id 형태로 생성
- 도서수
 - 총 10,000권의 도서를 대상으로 설정하였으며, 도서 목록 데이터의 인덱스(1부터 1048,001까지의 숫자) 중 100,000개가 랜덤으로 선택되어, 도서 목록 데이터와 매핑

	A	B
1	student_id	book_id
2	4983	616556
3	1219	318094
4	1195	883643
5	3185	759547
6	4843	811699
7	1364	611053
8	4454	93569
9	3314	1014647
10	2797	394683

생성한 대출기록 데이터

	A	B	C	D	E	F	G	H	I	J
1	서지번호	서명	저자	발행처	출판년도	ISBN	청구기호	언어	서지유형	
2	1	뚜껑없는 박설산	三藏		1994		811.8 ㅁ31	Korean	단행본(동양서)	
3	2	오른뇌 방고영희	집현전		1990		151 ㅁ367	Korean	단행본(동양서)	
4	3	잃어버린 진주 당신의심지			1989		151 ㅁ927	Korean	단행본(동양서)	
5	4	두뇌 개발 품천 가야	眞話堂		1985		151 ㄷ23ㄷ	Korean	단행본(동양서)	
6	5	두뇌 훈련 타고 아키	청림출판		1988		151 ㄷ23ㄷ	Korean	단행본(동양서)	
7	6	머리 만들 타고 아키	산하		1989		151 ㄷ23ㄷ	Korean	단행본(동양서)	
8	7	머리 만들 타고 아키	산하		1990		151 ㄷ23ㄷ	Korean	단행본(동양서)	
9	8	머리 만들 타고 아키	산하		1990		151 ㄷ23ㄷ	Korean	단행본(동양서)	
10	9	머리 만들 타고 아키	산하		1990		151 ㄷ23ㄷ	Korean	단행본(동양서)	

도서 목록 데이터. '서지번호'를 대출기록 데이터의 book_id와 매핑하여 사용.

학습 데이터 생성

The screenshot shows the Mockaroo website interface. The top navigation bar is green with the Mockaroo logo and links to SCHEMAS, DATASETS, APIS, DATABASES, SCENARIOS, PROJECTS, and FUNCTIONS. Below the navigation bar, there is a dark grey banner with the text: "Looking to generate fake data based on your production data? Mimic your databases with a trial account from TONIC". Below the banner, there is a dark grey box with the text: "Need some mock data to test your app? Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats. Need more data? Plans start at just \$60/year. Mockaroo is also available as a docker image that you can deploy in your own private cloud." Below the box, there is a table with the following columns: Field Name, Type, and Options. The table contains the following rows:

Field Name	Type	Options
id	Row Number	blank: 0 % Σ X
first_name	First Name	blank: 0 % Σ X
last_name	Last Name	blank: 0 % Σ X
email	Email Address	blank: 0 % Σ X
gender	Gender	blank: 0 % Σ X
ip_address	IP Address v4	blank: 0 % Σ X

Below the table, there are two buttons: "+ ADD ANOTHER FIELD" and "GENERATE FIELDS USING AI...". At the bottom, there is a row of settings: # Rows: 1000, Format: CSV, Line Ending: Unix (LF), Include: ☒ header ☐ BOM.

mockaroo라는 더미 데이터 생성 사이트

더미 데이터

- Mockaroo 사용
 - mockaroo라는 더미 데이터 생성 사이트를 활용하여 데이터를 랜덤으로 생성
 - 이 사이트는 논문 작성을 위한 더미 데이터 생성 등 다양한 용도로 사용되며, 데이터 타입을 정수, 실수, 문자열 이외의 ISBN, 비행기편명, 차종 등 여러 형태를 지원
 - 랜덤 대출 기록 생성:
 - 2번에서 설정한 데이터 형태를 바탕으로, Mockaroo를 통해 20,000건의 대출 기록을 생성했습니다. 각 대출 기록은 학생과 도서의 인덱스를 랜덤으로 매칭하여 생성

학습 데이터 생성

최종 데이터 구조

- 도서 목록 : 10,000권
 - {서명: str}, {저자: str}, {발행처: str}, {청구기호: int}, {서지유형: str}, {대분류: int}, {소분류: int}
- 대출 기록 : 총 20,000건, 특정 학생(ID)이 특정 도서(ID)를 대출한 정보를 담고 있습니다.
 - 학생: 500명 (ID, int)
 - 도서: 10,000권 (ID, int)

```
1번
학생 id 1 ~ 5000 사이 랜덤 생성
도서 id 1 ~ 1048001 사이 랜덤 생성 (학교로부터 제공받은 도서 목록 보유 도서 1048001권임)
총 6000개의 대출 기록

[3]: import pandas as pd

[2]: data = pd.read_csv("MOCK_DATA.csv")
data1 = pd.read_csv("MOCK_DATA (1).csv")
data2 = pd.read_csv("MOCK_DATA (2).csv")
data3 = pd.read_csv("MOCK_DATA (3).csv")
data4 = pd.read_csv("MOCK_DATA (4).csv")
data5 = pd.read_csv("MOCK_DATA (5).csv")

[4]: data.nunique()

[4]: student_id    904
book_id         1000
ISBN           1000
dtype: int64

[5]: data1.nunique()

[5]: student_id    908
book_id         1000
ISBN           1000
dtype: int64

[7]: concat = pd.concat([data, data1, data2, data3, data4, data5])

[9]: concat.nunique()

[9]: student_id    3488
book_id         5987
ISBN           6000
dtype: int64

[22]: concat['student_id'].value_counts()

[22]: student_id
2934     9
3366     7
972       6
2119     6
499       6
```

<concat.ipynb>

Mockaroo로 생성한 데이터를 병합하는 코드.

최종적으로 4번 형태의 데이터를 채택하여 사용

<book_list.ipynb>

대출기록 데이터와 도서 목록 데이터를 인덱스 기반으로 맵핑하여 하나의 데이터로 제작

```
[9]: import pandas as pd
import re

file_path = '학생여대서진_서지.xlsx'
df = pd.read_excel(file_path)

filtered_df = df[df['서지유형'].isin(['단행본(통양서)', '단행본(서양서)'])]
filtered_df = filtered_df[filtered_df['언어'].isin(['Korean', 'English'])]

c:\Users\mycom\AppData\Local\Programs\Python\Python312\Lib\site-packages\openpyxl\styles\stylesheet.py:237: UserWarning: Workbook contains no default style, apply openpyxl's default
warn("Workbook contains no default style, apply openpyxl's default")

[10]: def is_valid_title(title):
    return bool(re.match(r'^[\uac00-\ud7a3a-zA-Z]*$', title))

filtered_df = filtered_df[filtered_df['서명'].apply(is_valid_title)]

[21]: filtered_df['서지유형'].value_counts()

[21]: 서지유형
단행본(통양서)    190483
단행본(서양서)    35796
Name: count, dtype: int64

[22]: sampled_data = []
for 유형, group in filtered_df.groupby('서지유형'):
    sample_size = max(1, len(group) // 20)
    sampled_group = group.sample(n=sample_size, random_state=42)
    sampled_data.append(sampled_group)

final_df = pd.concat(sampled_data, ignore_index=True)

[23]: final_df['서지유형'].value_counts()

[23]: 서지유형
단행본(통양서)    9924
단행본(서양서)    1787
Name: count, dtype: int64

[24]: random_sample = final_df.sample(n=10000, random_state=1)

[25]: random_sample['서지유형'].value_counts()

[25]: 서지유형
단행본(통양서)    8446
단행본(서양서)    1554
Name: count, dtype: int64
```

[2]:

import pandas as pd

[3]:

df = pd.read_csv("data/book_list(2).csv")

[4]:

df = df[['서명', '저자', '발행처', '청구기호', '서지유형']]

[5]:

df

[19]:

	서명	저자	발행처	청구기호	서지유형
0	Poetry	Friedman, Norman	Harper & Row	808	단행본(서양서)
1	Women in cultures of the world	Hammond, Dorothy	Cummings Pub. Co	305	단행본(서양서)
2	이기는 종로 상공하는 PR	김주호	사계절	659	단행본(통양서)
3	위험한 이야기	히로세 타카시	푸른미디어	383	단행본(통양서)
4	리베르의 영혼 카일라스	Thurman, Robert A. F	아름	294	단행본(통양서)
...
9857	알씨구나	윌먼	윌람사	811	단행본(통양서)
9858	그 사랑의 유혹	박영애	푸른사상	915	단행본(통양서)
9859	The telescope handbook and star atlas	Howard, Neale E	Crowell	520	단행본(서양서)
9860	김지 산업화 발전에 관한 연구	한국 식물 공업 협회 식물 연구소	한국식물공업협회 식물연구소	641	단행본(통양서)
9861	영조와 내 개의 죽음	황규진	패이퍼로드	951	단행본(통양서)

9862 rows x 5 columns

[16]:

df['청구기호'] = df['청구기호'].astype(str)

[17]:

'대분류'는 청구기호의 첫 번째 숫자

df['대분류'] = df['청구기호'].str[0]

'소분류'는 청구기호의 첫 두 숫자

df['소분류'] = df['청구기호'].str[:2]

[18]:

df

[19]:

	서명	저자	발행처	청구기호	서지유형	대분류	소분류
0	Poetry	Friedman, Norman	Harper & Row	808	단행본(서양서)	8	80
1	Women in cultures of the world	Hammond, Dorothy	Cummings Pub. Co	305	단행본(서양서)	3	30

<book_type.ipynb>

청구기호에 따른 책의 분류가 영향을 주는지

구체적으로 보기 위해, 대분류, 소분류 열을 추가

추천 시스템

추천 시스템(recommendation system)은 사용자의 과거 행동, 선호도, 인구 통계적 정보를 분석하여 개인화된 콘텐츠, 제품, 정보를 제공하는 기술이다.

이 시스템은 사용자의 만족도와 플랫폼의 효율성을 동시에 높이는 데 활용된다.

도서관, 전자상거래, 미디어 스트리밍 플랫폼 등 다양한 분야에서 널리 사용되고 있으며, 사용자의 선택 과정을 간소화하고 서비스의 질을 높이는 데 기여한다.

사용자 기반
필터링

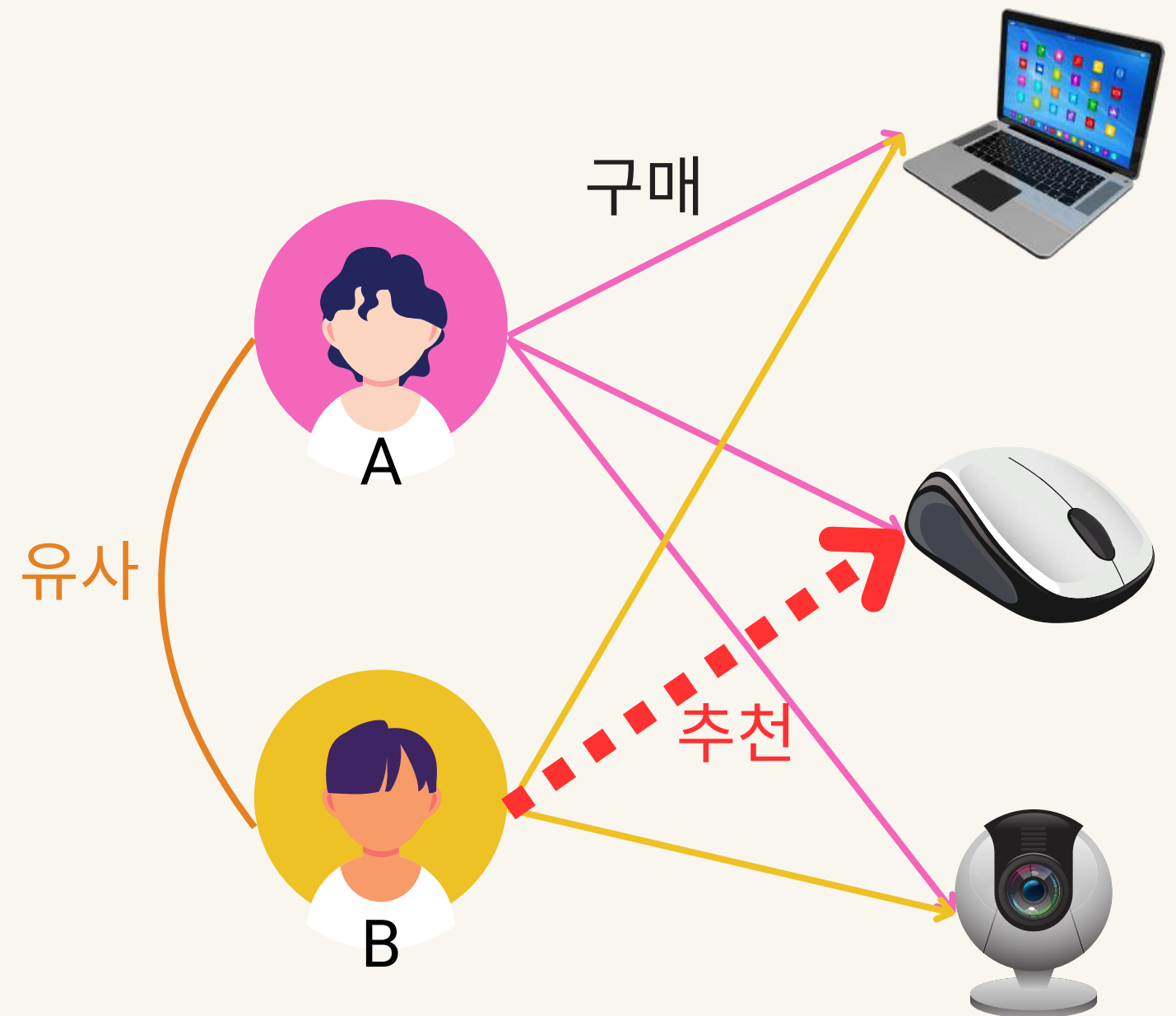
아이템 기반
필터링

콘텐츠 기반
필터링

사용자 기반 협업 필터링

협업필터링이란 간단하게 사람들의 행동 기록을 분석함을 의미한다. 특정 집단에서 발생하는 '유사한 사용 행동'을 파악하여 비슷한 성향의 사람들에게 아이템을 추천하는 기술이다. 협업 필터링은 사용자 기반 협업 필터링(User-based CF), 아이템 기반 협업 필터링(Item-based CF)으로 구분된다.

사용자 기반 협업 필터링이란 비슷한 성향을 가진 사람들이 사용한 아이템을 추천해주는 방식이다. 예를 들어, 사용자 A가 노트북, 마우스, 웹카메라를 구매하고, 사용자 B가 노트북, 웹카메라를 구매하였다고 가정하자. 알고리즘은 구매 목록이 겹치는 두 사용자가 유사하다고 판단하여 사용자 B에게 노트북 파워치를 추천할 것이다.



사용자 기반 협업 필터링

사용자 기반 협업 필터링에서 가장 중요한 부분은 사용자 사이의 유사도를 측정하는 것이다. 사용자와 사용자 간의 유사도를 계산하고 이를 기반으로 알고리즘을 구현한다. 여기서는 대출 여부로 유사도를 계산하기에 코사인 유사도 측정을 사용하였다.

주로 선호도를 기반으로 유사도를 측정하지만, 우리 데이터는 선호도에 대한 정보가 없기에 'student_id', 'book_id'를 기준으로 대여 여부에 따라 유사도를 측정했다. 먼저, Python의 `pivot_table()`을 이용하여 사용자-아이템 행렬 생성하였고 사용자 간 코사인 유사도 계산을 위해 `cosine_similarity()`함수를 사용하였다.

```
# 2. 사용자 간 유사도 계산
user_similarity = cosine_similarity(user_item_matrix)
user_similarity_df = pd.DataFrame(user_similarity, index=user_item_matrix.index, columns=user_item_matrix.index)
```

```
# 3. 추천 함수 정의
def recommend_books(student_id, user_similarity_df, user_item_matrix, top_n=2):
    # 입력 학생과 다른 학생 간의 유사도 가져오기
    similar_users = user_similarity_df[student_id].sort_values(ascending=False).drop(student_id)

    # 유사도가 높은 학생의 대여 기록 합산
    weighted_ratings = pd.Series(0, index=user_item_matrix.columns, dtype=float)
    for other_user, similarity_score in similar_users.items():
        weighted_ratings += user_item_matrix.loc[other_user] * similarity_score

    # 해당 학생이 이미 대여한 책 제외
    student_books = user_item_matrix.loc[student_id]
    weighted_ratings = weighted_ratings[student_books == 0]

    # 추천 책 정렬 후 반환
    return weighted_ratings.sort_values(ascending=False).head(top_n)
```

```
book['student_id'].value_counts()
```

```
student_id
384    4
394    3
68     3
479    3
44     3
..
463    1
459    1
461    1
72     1
74     1
Name: count, Length: 274, dtype: int64

student_id가 60인 학생에게 책 추천
```

<userbasedCF.ipynb>

사용자간 유사도 계산 코드와
추천 함수 생성 코드이다.

사용자 기반 협업 필터링

입력 학생과 다른 학생 간의 유사도를 가져오는 함수와 유사도가 높은 학생의 대여 기록 합산, 해당 학생이 이미 대여한 책 제외하여 추천 책을 정렬하여 반환하는 함수를 만들었으며, 'student_id'가 60번인 학생에게 'book_id'가 219인 <젊은 과학도에게 드리는 조언>, 3183인 <우리 자신 속의 독재자> 책을 추천한다.

그러나, 본 데이터는 사용자-아이템 행렬이 대부분 0이고 1이 거의 없는 것을 확인할 수 있다. 이와 같은 이유 때문에 코사인 유사도를 계산할 때 대부분의 유사도가 0이 나오는 문제를 직면한다. 따라서 사용자 기반 추천 시스템은 적절하지 않다. 그렇지만, '학생아이디', '책정보', '책선호도' 등의 정확한 정보를 담은 데이터를 얻는다면 보다 효과적인 도서 추천 시스템으로 활용할 수 있을 것이라 기대한다.

```
book[book['book_id'] == 219]['서명']
```

```
114    젊은 과학도에게 드리는 조언
393    젊은 과학도에게 드리는 조언
Name: 서명, dtype: object
```

```
book[book['book_id'] == 3183]['서명']
```

```
95    우리 자신 속의 독재자
104   우리 자신 속의 독재자
109   우리 자신 속의 독재자
122   우리 자신 속의 독재자
Name: 서명, dtype: object
```

```
#사용자-아이템 행렬 생성
user_item_matrix = book.pivot_table(index='student_id', columns='book_id', aggfunc='count', fill_value=0)
user_item_matrix
```

	ISBN ...										출판년도										
book_id	33	119	219	261	267	285	355	436	472	488	...	9015	9121	9445	9450	9499	9520	9648	9712	9932	9992
student_id																					
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...
493	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
494	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
496	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
497	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
499	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

274 rows x 1336 columns

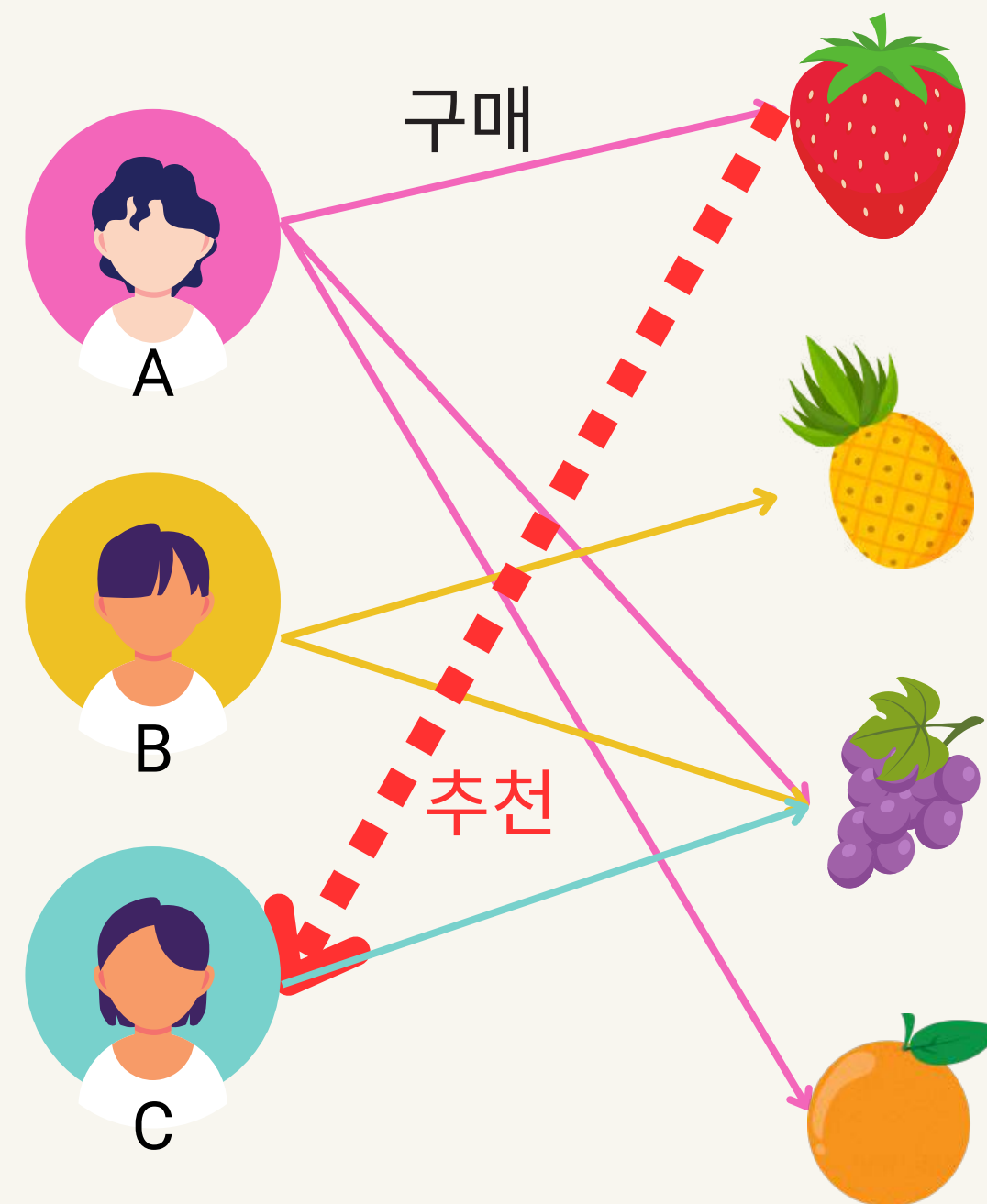
<userbasedCF.ipynb>

사용자간 유사도 계산 코드와
추천 함수 생성 코드이다.

아이템 기반 협업 필터링

아이템 기반 추천 시스템은 최근접 이웃 협업 필터링의 한 종류이며
아이템 기반 최근접 이웃 방식이라고도 한다. 아이템 기반 최근접
이웃 방식은 '아이템 간 속성'이 얼마나 비슷한지를 기반으로 추천
하는 것이 아닌 그 아이템을 좋아하는지 싫어하는지의 평가 척도가
유사한 아이템을 추천하는 시스템이다.

아이템 기반 추천 시스템에 사용되는 데이터는 다차원 희소 행렬이
라는 특징을 가지고 있어 유사도 측정을 위해 주로 코사인 유사도를
사용한다. 이때 코사인 유사도란 벡터와 벡터 간의 유사도를 비교할
때 벡터의 크기보다는 벡터의 상호 방향성이 얼마나 유사한지에 기
반하며 두 벡터 사이의 사잇각을 구해 얼마나 유사한지 수치화한다.



아이템 기반 협업 필터링

본 분석의 목표는 아이템 기반 추천 시스템을 통해 도서관이 보유하고 있는 책과 학생 간의 연관성을 분석하여 추천 시스템을 설계하는 데 있다. 학생 아이디와 책 간의 행렬을 만들고 코사인 유사도를 이용해 수치를 구한 결과, 이 데이터에서는 사용자 기반 협업 필터링과 마찬가지로 아이템 기반 추천 시스템이 성공적으로 작동하지 않는 것을 알 수 있다.

문제 원인으로 유저-아이템 행렬이 대부분 0이고 1이 거의 없는 것을 확인할 수 있었다. 이와 같은 이유 때문에 코사인 유사도를 계산할 때 대부분의 유사도가 0이 나옴을 확인할 수 있었다.

따라서 이 데이터에서는 아이템 기반 추천 시스템은 적절하지 않은 것을 확인할 수 있었다. 이 점을 개선하기 위해서 데이터 자체에 선호도를 나타낼 수 있는 수치를 적용하면 좋을 것이다.

```
from sklearn.metrics.pairwise import cosine_similarity

# 아이템 간(책 간) 코사인 유사도 계산
item_similarity = cosine_similarity(user_item_matrix.T)

item_similarity_df = pd.DataFrame(item_similarity, index=user_item_matrix.columns, columns=user_item_matrix.columns)

# 유사도 상위 5개 확인
item_similarity_df.iloc[:5, :5]
```

book_id	33	119	219	261	267
book_id					
33	1.0	0.0	0.0	0.0	0.0
119	0.0	1.0	0.0	0.0	0.0
219	0.0	0.0	1.0	0.0	0.0
261	0.0	0.0	0.0	1.0	0.0
267	0.0	0.0	0.0	0.0	1.0

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

# 유저-아이템 행렬 생성 (pivot_table 사용)
user_item_matrix = data.pivot_table(index='student_id', columns='book_id', aggfunc='size', fill_value=0)

# 아이템-아이템 유사도 계산
item_user_matrix = user_item_matrix.T # 전치행렬로 아이템-유저 행렬 생성
item_similarity = cosine_similarity(item_user_matrix) # 코사인 유사도 계산

# 데이터프레임으로 변환
item_similarity_df = pd.DataFrame(item_similarity,
                                  index=user_item_matrix.columns,
                                  columns=user_item_matrix.columns)

def recommend_books_for_user(user_id, top_n=5):
    """
    특정 사용자를 위한 책 추천.
    - user_id: 추천을 요청할 사용자 ID
    - top_n: 추천할 책의 개수
    """
    # 사용자가 이미 대출한 책 목록
    user_books = user_item_matrix.loc[user_id]
    user_books = user_books[user_books > 0].index # 대출한 책 ID만 추출

    # 사용자가 대출한 책들과 유사한 책들의 점수 계산
    similar_books = item_similarity_df[user_books].sum(axis=1).sort_values(ascending=False)

    # 이미 대출한 책은 제외
    similar_books = similar_books[~similar_books.index.isin(user_books)]

    # 상위 추천 책 선택
    recommended_books = blist[blist['서지번호'].isin(similar_books.index)].copy()
    recommended_books['score'] = similar_books.loc[recommended_books['서지번호']].values

    return recommended_books.nlargest(top_n, 'score')[['서지번호', '서명', '저자', 'score']]

# 테스트 실행
test_user_id = 140
recommendations = recommend_books_for_user(user_id=test_user_id, top_n=5)
print(recommendations)
```

	서지번호	서명	저자	score
22	6425	영보 창리 기사 글자집	구룡산	0.0
39	9015	너와 나의 사랑을 위하여	김동길	0.0
40	5626	기초 무기화학	Cotton, F. Albert	0.0
84	1314	불로 단나는 세계 여행	손호철	0.0
168	2644	악화 도란 사건	유림문화사	0.0

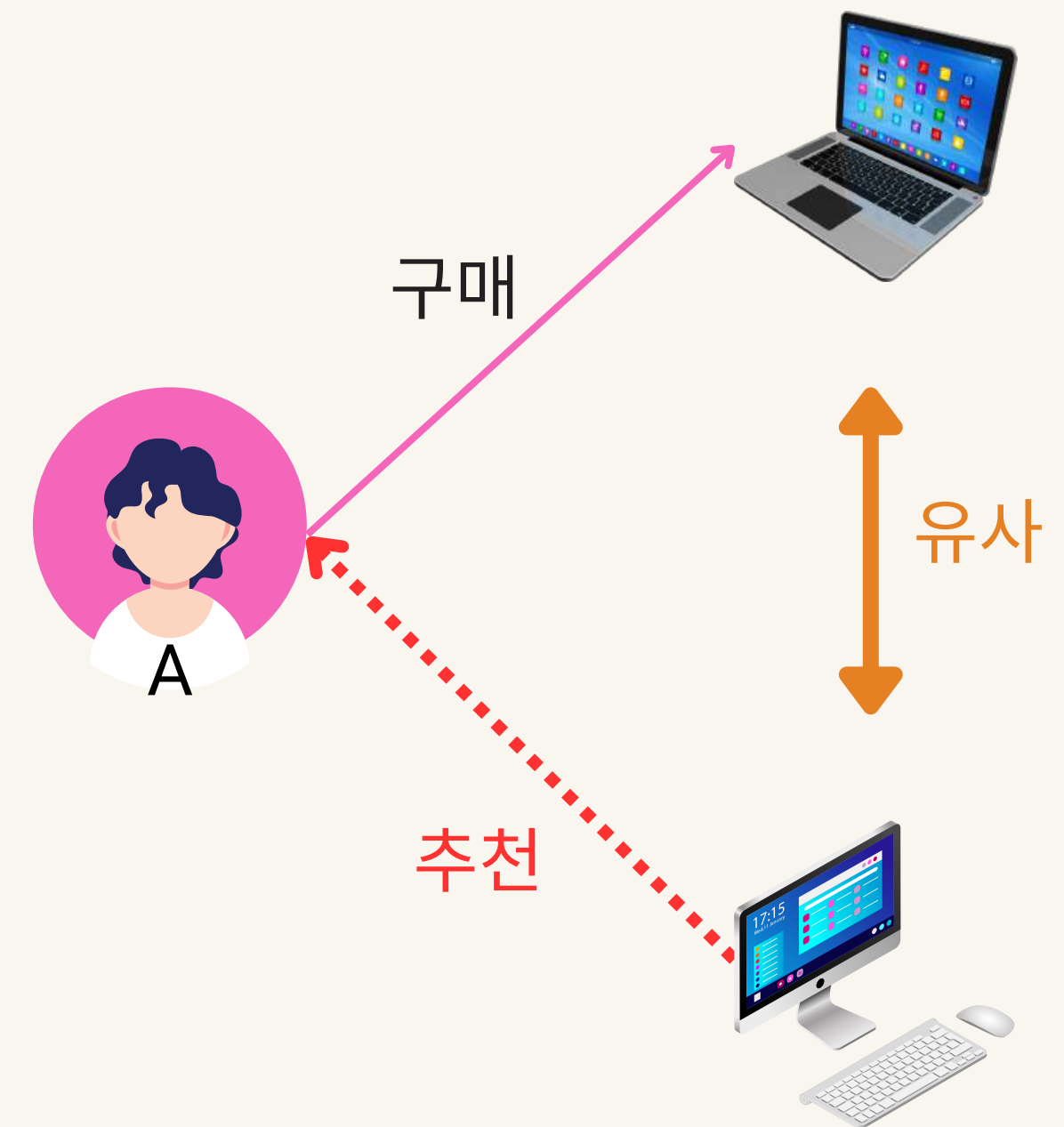
<item.ipynb>

코사인 유사도 계산 코드와
추천 함수 생성 후 출력한
추천 결과 값이다.

콘텐츠 기반 필터링

콘텐츠 기반 필터링은 사용자가 이전에 선호한 아이템의 속성을 분석하여 유사한 아이템을 추천하는 방식이다.

예를 들어, 사용자가 노트북을 구매했다면, 유사한 기능을 하는 컴퓨터를 추천한다. 이 방법은 개별 사용자의 데이터를 중점적으로 활용한다는 점에서 개인화에 유리하지만, 새로운 아이템 추천에 한계가 있다.



콘텐츠 기반 필터링

추천 시스템은 도서의 텍스트 데이터를 기반으로 TF-IDF 벡터화를 통해 각 도서의 특징을 수치화하는 것으로 시작됩니다. TF-IDF 벡터는 단어의 빈도와 중요도를 반영하여 도서의 내용을 정량적으로 표현합니다. 이후, 이렇게 변환된 벡터를 이용해 코사인 유사도를 계산하여 도서 간의 유사성을 측정합니다. 마지막으로, 유사도가 높은 도서들을 찾아 추천 대상 도서와 비슷한 항목을 사용자에게 제시함으로써 개인화된 추천을 제공합니다.

TF-IDF와 코사인 유사도를 활용한 추천 시스템은 구현이 간단하고 효율적이며, 텍스트 데이터를 기반으로 유사한 도서를 추천하는 데 적합하다는 장점이 있습니다. 도서 제목, 저자, 출판사 등 텍스트 정보를 활용하여 직관적이고 빠른 추천이 가능하며, 소규모 데이터셋에서도 높은 성능을 발휘합니다. 특히, 도서 소개 데이터를 추가로 크롤링하면 추천의 질을 더욱 향상시킬 수 있습니다. 그러나, 이 시스템은 단어의 순서나 문법적 구조를 반영하지 못하기 때문에 동일한 의미의 문장이라도 서로 다른 벡터로 처리될 가능성이 있습니다. 또한, 단어의 문맥적 의미를 파악하지 못해 정교한 추천에는 한계가 있으며, 추가적인 텍스트 데이터를 활용하더라도 이 단점은 쉽게 극복되지 않는 어려움으로 남습니다.

<sum.ipynb>

책 정보 관련 칼럼 전처리 과정과 TF-IDF 벡터화, 코사인 유사도 계산 코드이다.

```
# 주요 칼럼 전처리
books_df['저자'] = books_df['저자'].fillna('').astype(str).apply(lambda x: preprocess_text_multilingual(x, language='ko'))
books_df['발행처'] = books_df['발행처'].fillna('').astype(str).apply(lambda x: preprocess_text_multilingual(x, language='ko'))
books_df['출판년도'] = books_df['출판년도'].fillna('').astype(str).apply(lambda x: preprocess_text_multilingual(x, language='ko'))

# 책 속성 결합
books_df['book_features'] = (
    books_df['서명'].fillna('') + ' ' +
    books_df['저자'] + ' ' +
    books_df['발행처'] + ' ' +
    books_df['서지유형'].fillna('') + ' ' +
    books_df['대분류'].fillna('') + ' ' +
    books_df['소분류'].fillna('')
)

# 결합한 데이터에서 결측값 처리
books_df['book_features'] = books_df['book_features'].fillna('')

# TF-IDF 벡터화
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(books_df['book_features'])
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Doc2Vec 벡터 변환
documents = [TaggedDocument(words=content.split(), tags=[str(i)]) for i, content in enumerate(books_df['book_features'])]
doc2vec_model = Doc2Vec(documents, vector_size=200, window=5, min_count=2, epochs=40)
content_vectors = np.array([doc2vec_model.dv[str(i)] for i in range(len(books_df['book_features']))])

# TF-IDF와 Doc2Vec 기반 유사도 계산
article_similarity_tfidf = linear_kernel(tfidf_matrix, tfidf_matrix)
article_similarity_doc2vec = cosine_similarity(content_vectors)
article_similarity_total = 0.5 * article_similarity_tfidf + 0.5 * article_similarity_doc2vec

# 학생-책 활용 생성 (유저-아이템 활용)
students_books_df['interaction'] = 1
user_article_matrix = students_books_df.pivot_table(index='student_id', columns='book_id', values='interaction', fill_value=0).values
```


콘텐츠 기반 필터링

특정 학생이 읽은 책의 데이터를 기반으로 개인화된 추천 도서를 계산하여 상위 몇 개의 도서를 제안하고 있습니다. 학생이 읽은 책 목록이 있을 경우, 해당 책들의 속성 벡터 평균을 계산하여 개인화된 벡터를 생성하며, 읽은 책이 없을 때는 전체 데이터의 평균 벡터를 사용합니다. 이후, 개인화된 벡터와 전체 책 벡터 간의 내적을 계산해 유사도를 점수로 매기고, 가장 높은 점수를 가진 책들을 상위 top_n으로 추출합니다. 예시로, 학생 ID 1의 데이터를 활용한 결과는 상위 5개의 도서를 반환했으며, 각 책의 ID, 저자, 발행처, 출판년도 정보를 제공합니다. 다만, 이 데이터는 시뮬레이션 기반으로 생성된 것이므로 추천의 정확도를 정량적으로 평가하기 어렵다는 한계가 있습니다. 이는 실제 데이터를 활용한 추가적인 분석이 필요함을 시사합니다.

<sum.ipynb>

책 정보 관련 칼럼 전처리 과정과
TF-IDF 벡터화, 코사인 유사도 계산 코드이다.

```
# 개인화된 추천 점수 계산
def personalized_recommend_books(student_id, top_n=5):
    # 학생이 읽은 책 목록 가져오기
    student_books = students_books_df[students_books_df['student_id'] == student_id]['book_id'].tolist()

    # 학생이 읽은 책들의 속성 벡터 평균 계산
    read_books_indices = books_df[books_df['book_id'].isin(student_books)].index
    if len(read_books_indices) > 0:
        # 읽은 책들의 유사도 평균 (TF-IDF, Doc2Vec 기반)
        personalized_vector = np.mean(article_similarity_total[read_books_indices], axis=0)
    else:
        # 읽은 책이 없을 경우 전체 평균 사용
        personalized_vector = np.mean(article_similarity_total, axis=0)

    # 개인화 점수 기반 추천 계산
    scores = np.dot(personalized_vector, article_similarity_total.T)

    # 상위 점수 책 인덱스 추출
    recommended_indices = scores.argsort()[-top_n:][::-1]

    # 추천된 책 정보 반환
    recommended_books = books_df.iloc[recommended_indices]
    return recommended_books[['book_id', '저자', '발행처', '출판년도']]
```

```
# 예시: 학생 ID가 1인 경우 개인화된 상위 5개 추천
student_id = 1
personalized_books = personalized_recommend_books(student_id)
print(personalized_books)
```

	book_id	저자	발행처	출판년도
8212	1372728	김원국	미래	
1947	1803565	김정희	학지사	
2566	44285			
431	1898707	강세황	지식산업사	
1048	156957		책세상	

알고리즘 고도화 방안



도서관 웹사이트 크롤링 칼럼 추가

크롤링이란 웹에서 필요한 데이터를 수집하는 작업이다. 크롤링은 크게 정적 크롤링과 동적 크롤링으로 나뉜다. 정적 크롤링은 정적인 데이터를 수집하는 크롤링이다. 보통 한 페이지에서 원하는 정보가 모두 포함되었을 경우, 정적 크롤링을 사용한다. 동적 크롤링은 동적인 데이터를 수집하는 크롤링이다. 원하는 데이터를 얻기 위해 페이지 이동이 필요한 경우 사용한다.



대출 기록 기반 추천 시스템과 유형 검사 융합의 제안

책 선택의 주요 요소는 책의 내재적 가치이지만, 외적 요소인 두께, 독서 목적, 개인 성향도 중요한 영향을 미치는 요소이다. 기존의 대출 기록 데이터는 사용자 정보를 제한적으로 제공하는 한계를 지니고 있다. 덕성여자대학교에서 독서 취향 유형 검사를 도입하여 개인화된 독서 경험을 제공하는 것이 필요하다. 이 검사는 게임화된 요소와 피드백 시스템을 포함하여 높은 참여도를 유도하며, 기존 알고리즘과 결합할 경우 사용자 만족도를 크게 향상시킬 수 있는 방안이다.

```

import time
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import pandas as pd
import time

# ChromeDriver 실행 파일 경로 설정
driver_path = "chromedriver-win64/chromedriver.exe" # .exe 확장자 확인 필요
service = Service(driver_path)
driver = webdriver.Chrome(service=service)

# 도서관
url = "https://discover.duksung.ac.kr/#/"
driver.get(url)
time.sleep(2)

# 창 최대 크기
driver.maximize_window()
time.sleep(2)

for book in blist['서명'][:5]:
    try:
        # 검색창 찾기
        element = driver.find_element(By.ID, "keyword")
        element.click()
        element.clear() # 검색창 초기화
        element.send_keys(book) # 도서관 책 제목 입력
        element.send_keys("\n") # Enter키 입력하여 검색 시작

        # 페이지 로딩 대기 (최대 5초)
        WebDriverWait(driver, 5).until(
            EC.presence_of_element_located((By.CSS_SELECTOR, ".result-item"))) # 결과 항목이 로드될 때까지
        )

        # 첫 번째 검색 결과의 링크를 클릭 (정확한 XPath로 수정)
        first_result_link = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH, f"//a[span[contains(text(), '{book}')]"))
        )

        # 클릭 가능한 상태인지 확인한 후 스크롤
        driver.execute_script("arguments[0].scrollIntoView(true);", first_result_link)
        first_result_link.click()
        # 페이지 로딩 대기 (새로운 페이지가 로드될 때까지 대기)
        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.TAG_NAME, "body"))
        )
        print(f"'{book}'의 상세 페이지로 이동 완료!")

    except Exception as e:
        print(f"Error with book '{book}': {e}")

# 작업 완료 후 드라이버 종료
# driver.quit()

```

교내 도서관 웹사이트 기반 크롤링 칼럼 추가

해당 데이터 분석을 위해서 학교 도서관 페이지의 검색창에 도서를 검색한 후 상세 정보로 이동을 해야 하기 때문에 동적 크롤링 방법을 사용하였다.

웹 브라우저를 자동으로 제어하여 웹페이지를 테스트하고 조작할 수 있게 해주는 프레임워크인 Selenium 라이브러리를 사용한다. 분석의 단계는 다음과 같다.

1. 도서관 홈페이지에서 검색창을 찾은 뒤 도서 정보가 들어있는 리스트에서
2. 도서명 정보가 들어있는 컬럼의 정보를 입력한다.
3. 상세 페이지의 도서 정보를 클릭한다.
4. 해당 도서의 서평 정보를 끌어온다.
5. 이 과정을 반복한다.

이 과정을 통해 도서관에 등록되어있는 서평 정보를 얻어올 수 있다.


대출 기록 기반 추천 시스템의 유효성과 유형 검사 융합의 제안



현재 대학 도서관의 추천 시스템은 대출 목록 기반 알고리즘을 활용하고 있다. 이는 정성적 분석 부족과 같은 한계를 가지고 있다.


이를 해결하기 위해서는 유형 검사 기반 추천 시스템을 결합하는 것이 필요하다.

이 방식은 개인의 독서 성향을 분석해 추천의 정확도와 몰입도를 높일 수 있는 방안이다(조현양, 2017).




내 독서 유형은?

DSAP



테스트 시작하기




2/5

Q2.

책을 읽을 때 가장 중요한 건?

힐링! 위로와 감성이 필요해.

속도! 알고 부담 없는 책이 최고.




3/5

Q5.

전공 서적을 볼 때 드는 생각?

내 전문성에 도움이 되니까 참아본다.

이것만으로도 충분하니, 다른 책은 안 읽어.




창문 넘어 도망친

짧고 굵게 책 읽는 효율주의자

실속 간결형

일상에서 가볍게 읽을 수 있는 책을 통해
짧지만 강렬한 메시지를 받으며, 간결한
효율성을 추구하는 당신!

책 두께가 부담스럽다면 얇고 간결한 책으로도
충분하다고 생각하는 타입입니다! 한 권이라도
끝내는 게 중요한 당신은 "질보다 양"이라는
생각으로 속도감 있게 독서를 즐기죠.



COSMOS

학문 탐구에 몰입하는 지식형 인간

전공 생존형

"공부가 내 삶의 중심이다"라는 생각으로 시험
기간에는 다른 책은 잠시 미뤄두는 실용적인
당신!

전공 서적과 논문 읽기에 대부분의 시간을
쏟는 타입입니다! 한정된 시간 안에서 지식을
깊이 있게 쌓으려 노력하며, 전공을 통해
스스로의 전문성을 확립하고 싶어하네요.

당신을 위한 맞춤 도서
칼 세이건 - 《코스모스》

감사합니다.

DSAP