

추천 시스템 (Recommender Systems) Part1

학습 목표

- 추천 시스템 모델의 개념과 구현을 알아본다.

주요 내용

1. 추천시스템
2. 사용자 기반 협업 필터링
3. 아이템 기반 협업 필터링



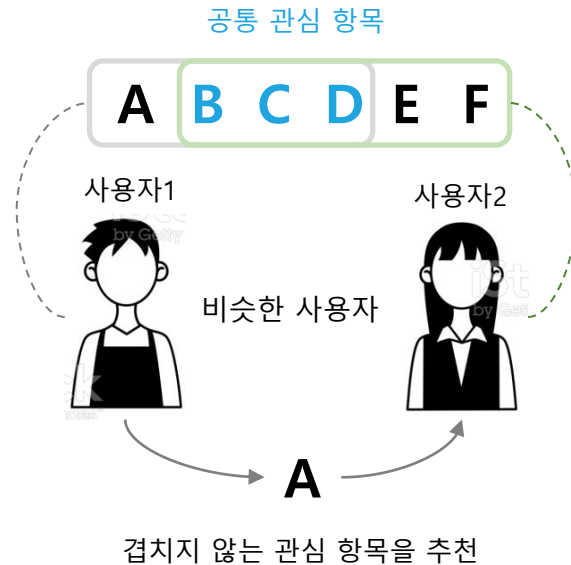
1. 추천 시스템



추천시스템 (Recommender Systems)

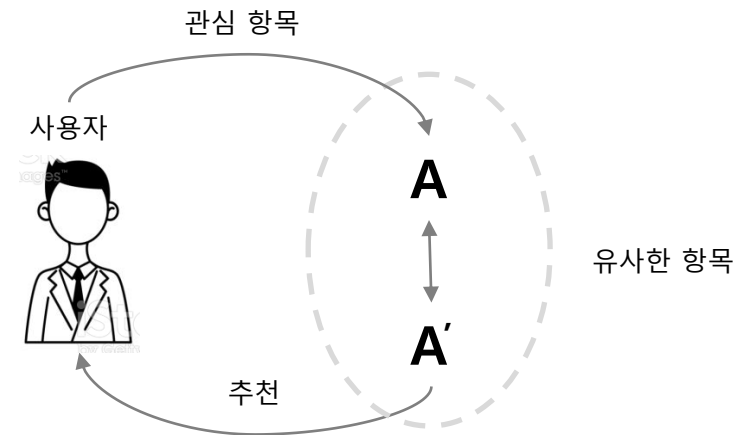
컨텐츠 또는 사용자의 행동을 토대로 사용자가 관심을 가질 만한 정보를 추천하는 시스템

협업 필터링 (Collaborative Filtering)



- 사람들의 행동을 기반으로 추천해주는 방식
- 나와 관심사가 비슷한 사용자의 관심 항목을 추천
- 나의 관심 항목과 사용자 군이 비슷한 관심 항목을 추천

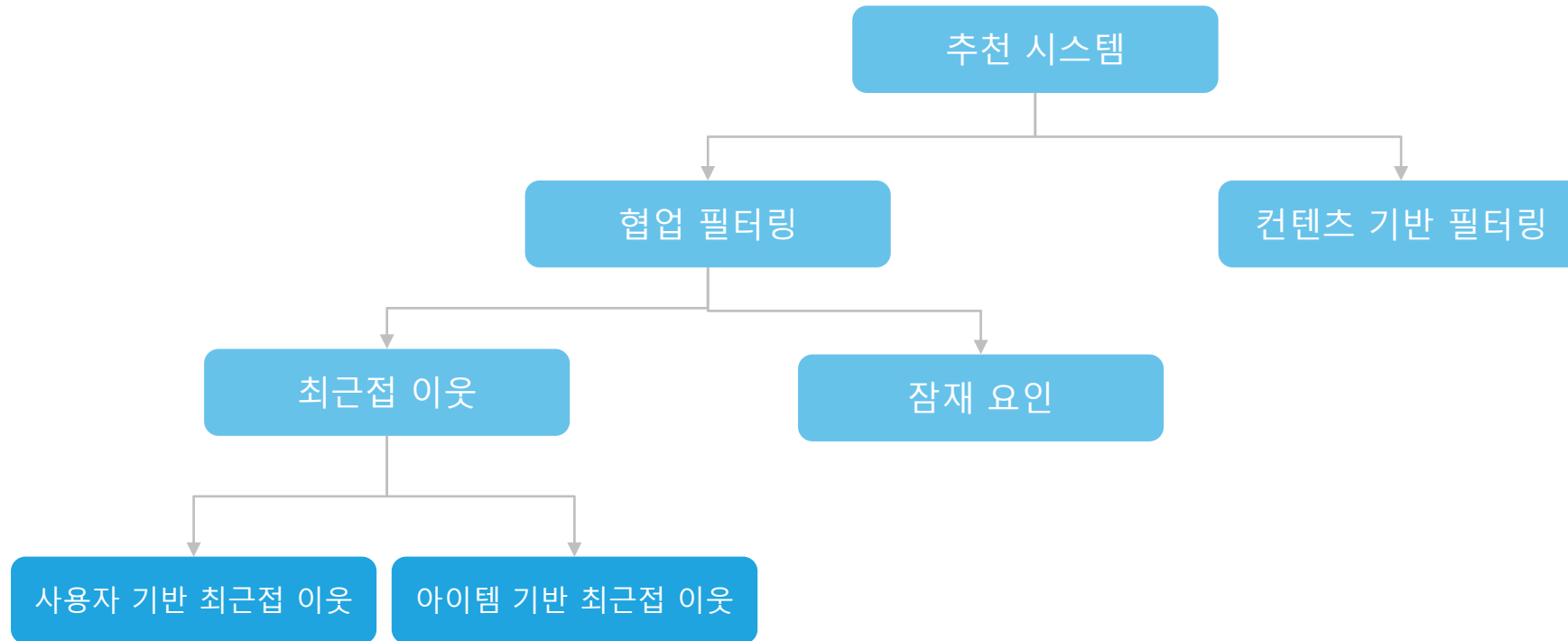
컨텐츠 기반 필터링 (Content-Based Filtering)



- 관심 항목의 컨텐츠를 분석해서 유사한 항목을 추천
- 예를 들어, 유튜브의 경우 동영상 자막이나 설명을 컨텐츠로 해서 유사한 컨텐츠를 찾을 수 있다.

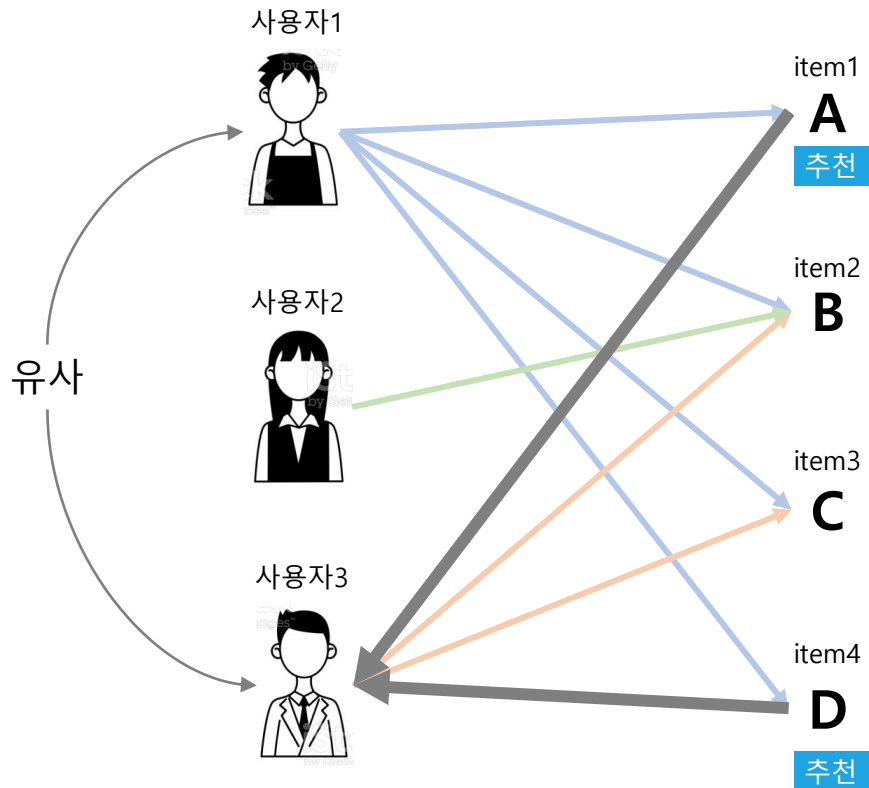
추천시스템 (Recommender Systems)

컨텐츠 또는 사용자의 행동을 토대로 사용자가 관심을 가질 만한 정보를 추천하는 시스템



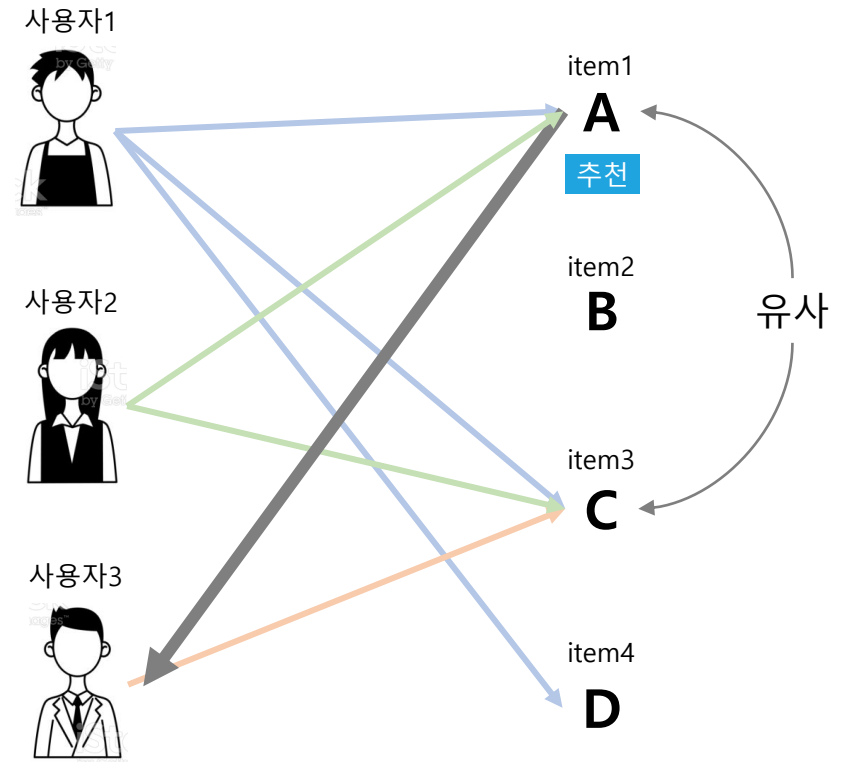
협업 필터링 (Collaborative Filtering)

사용자 기반 협업 필터링 (User-Based CF)



- 사용자3은 사용자1과 관심사가 유사
- 사용자3에게 사용자1의 관심사 A와 D를 추천

아이템 기반 협업 필터링 (Item-Based CF)



- 사용자3이 구매한 C는 A와 사용자가 유사
- 사용자3에게 A를 추천

데이터셋 정의

사용자 별 관심 목록

```
users_interests = [  
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],  
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],  
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],  
    ["R", "Python", "statistics", "regression", "probability"],  
    ["machine learning", "regression", "decision trees", "libsvm"],  
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],  
    ["statistics", "probability", "mathematics", "theory"],  
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],  
    ["neural networks", "deep learning", "Big Data", "artificial intelligence"],  
    ["Hadoop", "Java", "MapReduce", "Big Data"],  
    ["statistics", "R", "statsmodels"],  
    ["C++", "deep learning", "artificial intelligence", "probability"],  
    ["pandas", "R", "Python"],  
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],  
    ["libsvm", "regression", "support vector machines"]  
]
```

인기 순위로 추천

관심 종목 별 인기 순위

```
from collections import Counter

popular_interests = Counter(interest
                             for user_interests in users_interests
                             for interest in user_interests)
```

Counter({'Python': 4, 'R': 4, 'Big Data': 3, 'HBase': 3, 'Java': 3, 'statistics': 3, 'regression': 3, 'probability': 3, 'Hadoop': 2, 'Cassandra': 2, 'MongoDB': 2, 'Postgres': 2, 'scikit-learn': 2, 'statsmodels': 2, 'pandas': 2, 'machine learning': 2, 'libsvm': 2, 'C++': 2, 'neural networks': 2, 'deep learning': 2, 'artificial intelligence': 2, 'Spark': 1, 'Storm': 1, 'NoSQL': 1, 'scipy': 1, 'numpy': 1, 'decision trees': 1, 'Haskell': 1, 'programming languages': 1, 'mathematics': 1, 'theory': 1, 'Mahout': 1, 'MapReduce': 1, 'databases': 1, 'MySQL': 1, 'support vector machines': 1})

인기 순위로 추천

사용자에 관심 목록에 없는 인기 높은 관심 종목 추천

```
from typing import Dict, List, Tuple

def most_popular_new_interests(
    user_interests: List[str],
    max_results: int = 5) -> List[Tuple[str, int]]:
    suggestions = [(interest, frequency)
                    for interest, frequency in popular_interests.most_common()
                    if interest not in user_interests]
    return suggestions[:max_results]
```

사용자 1 추천 목록

```
print(most_popular_new_interests(users_interests[1]))
```

관심 목록 ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],

추천 목록 [('Python', 4), ('R', 4), ('Big Data', 3), ('Java', 3), ('statistics', 3)]

2. 사용자 기반 협업 필터링



사용자 행동 분석

관심사가 비슷한 사용자의 행동을 분석해서 새로운 행동을 추천

관심 항목

	Item1	Item2	Item3	Item4	Item5
User1	O	O		O	O
User2		O		O	
User3		O	O	O	O
User4	O		O		O

사용자

1 사용자-아이템 행렬 만들기

2 사용자 별로 관심사의 유사도 측정



사용자3이 사용자1과 관심사가 가장 비슷함

3 관심사가 유사한 사용자의 관심사를 추천

사용자 1에게 새로운 item3을 추천

아이템 목록

아이템 목록 생성

```
unique_interests = sorted({interest
                           for user_interests in users_interests
                           for interest in user_interests})
```

- 집합으로 만들어서 중복을 제거한 후 정렬

['Big Data', 'C++', 'Cassandra', 'HBase', 'Hadoop', 'Haskell', 'Java', 'Mahout', 'MapReduce', 'MongoDB', 'MySQL', 'NoSQL', 'Postgres', 'Python', 'R', 'Spark', 'Storm', 'artificial intelligence', 'databases', 'decision trees', 'deep learning', 'libsvm', 'machine learning', 'mathematics', 'neural networks', 'numpy', 'pandas', 'probability', 'programming languages', 'regression', 'scikit-learn', 'scipy', 'statistics', 'statsmodels', 'support vector machines', 'theory']


사용자-아이템 행렬

사용자 관심 목록을 사용자-아이템 행렬로 변환

사용자 관심 목록

```
users_interests = [  
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],  
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],  
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],  
    ["R", "Python", "statistics", "regression", "probability"],  
    ["machine learning", "regression", "decision trees", "libsvm"],  
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],  
    ["statistics", "probability", "mathematics", "theory"],  
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],  
    ["neural networks", "deep learning", "Big Data", "artificial intelligence"],  
    ["Hadoop", "Java", "MapReduce", "Big Data"],  
    ["statistics", "R", "statsmodels"],  
    ["C++", "deep learning", "artificial intelligence", "probability"],  
    ["pandas", "R", "Python"],  
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],  
    ["libsvm", "regression", "support vector machines"]  
]
```

사용자-아이템 행렬 (user_interest_vectors)



	Item1	Item2	Item3	Item4	Item5
User1	1	1	0	1	1
User2	0	1	0	1	0
User3	0	1	1	1	1
User4	1	0	1	0	1

사용자-아이템 행렬

```
user_interest_vectors = [make_user_interest_vector(user_interests)  
    for user_interests in users_interests]
```

사용자-아이템 행렬

사용자 별 관심 목록을 이진 벡터로 변환

["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"]



[1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

사용자 별 관심 목록 벡터

```
def make_user_interest_vector(user_interests: List[str]) -> List[int]:  
    """  
    Given a list of interests, produce a vector whose ith element is 1  
    if unique_interests[i] is in the list, 0 otherwise  
    """  
    return [1 if interest in user_interests else 0  
            for interest in unique_interests]
```

사용자 유사도 행렬

사용자 별로 다른 모든 사용자와의 유사도를 측정

사용자-아이템 행렬 (user_interest_vectors)

	Item1	Item2	Item3	Item4	Item5
User1	1	1	0	1	1
User2	0	1	0	1	0
User3	0	1	1	1	1
User4	1	0	1	0	1



사용자 유사도 행렬 (user_similarities)

	User1	User2	User3	User4
User1	1.0	0.3	0.8	0.1
User2	0.3	1.0	0.22	0.9
User3	0.8	0.22	1.0	0.4
User4	0.1	0.9	0.4	1.0

- 대칭 행렬
- 대각은 자기 자신과의 유사도로 1

```
from scratch.nlp import cosine_similarity

user_similarities = [[cosine_similarity(interest_vector_i, interest_vector_j)
                       for interest_vector_j in user_interest_vectors]
                      for interest_vector_i in user_interest_vectors]
```

- 사용자 별 관심 벡터의 코사인 유사도 측정

참고 코사인 유사도 (cosine similarity)

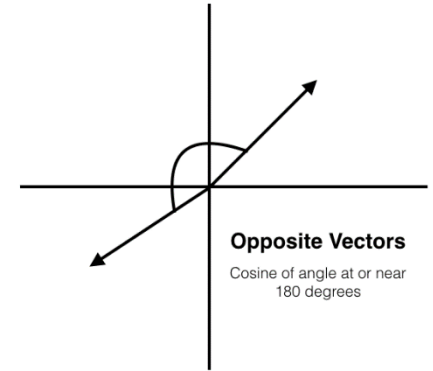
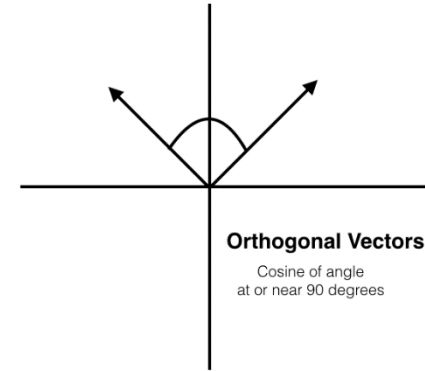
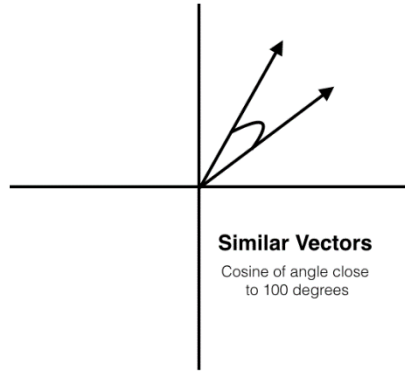
내적 (Dot product)

$$v \cdot w = \|v\| \|w\| \cos \theta$$

코사인 유사도 (cosine similarity)

$$\cos \theta = \frac{v \cdot w}{\|v\| \|w\|}$$

$$\|v\| = \sqrt{v \cdot v}, \|w\| = \sqrt{w \cdot w}$$



```
from scratch.linear_algebra import dot, Vector
import math

def cosine_similarity(v1: Vector, v2: Vector) -> float:
    return dot(v1, v2) / math.sqrt(dot(v1, v1) * dot(v2, v2))
```

관심사가 유사한 사용자 목록

사용자 유사도 행렬을 이용해서 관심사가 유사한 사용자 목록을 구성

```
def most_similar_users_to(user_id: int) -> List[Tuple[int, float]]:
    pairs = [(other_user_id, similarity)                # Find other
              for other_user_id, similarity in          # users with
                  enumerate(user_similarities[user_id]) # nonzero
              if user_id != other_user_id and similarity > 0] # similarity.

    return sorted(pairs,                               # Sort them
                  key=lambda pair: pair[-1],           # most similar
                  reverse=True)                       # first.
```

user_id →

	User1	User2	User3	User4
User1	1.0	0.3	0.8	0.1
User2	0.3	1.0	0.22	0.9
User3	0.8	0.22	1.0	0.4
User4	0.1	0.9	0.4	1.0

사용자 유사도 행렬 (user_similarities)

다른 사용자와의 유사도 목록

pairs = [(user1, 0.3) , (user3, 0.22), (user4, 0.9)]

↓ 유사도 순으로 정렬

[(user4, 0.9), (user1, 0.3), (user3, 0.22)]

관심사가 유사한 사용자 목록

사용자 0과 가장 유사한 사용자는 누구일까?

```
most_similar_to_zero = most_similar_users_to(0)
```

```
users_interests = [
    0 ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],
    1 ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],
    2 ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],
    3 ["R", "Python", "statistics", "regression", "probability"],
    4 ["machine learning", "regression", "decision trees", "libsvm"],
    5 ["Python", "R", "Java", "C++", "Haskell", "programming languages"],
    6 ["statistics", "probability", "mathematics", "theory"],
    7 ["machine learning", "scikit-learn", "Mahout", "neural networks"],
    8 ["neural networks", "deep learning", "Big Data", "artificial intelligence"],
    9 ["Hadoop", "Java", "MapReduce", "Big Data"],
    10 ["statistics", "R", "statsmodels"],
    11 ["C++", "deep learning", "artificial intelligence", "probability"],
    12 ["pandas", "R", "Python"],
    13 ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],
    14 ["libsvm", "regression", "support vector machines"]
]
```

```
[(9, 0.5669467095138409),
 (1, 0.3380617018914066),
 (8, 0.1889822365046136),
 (13, 0.1690308509457033),
 (5, 0.1543033499620919)]
```

사용자 9와 가장 유사하며
유사도는 56%


사용자 기반 추천

관심사가 유사한 사용자의 관심사 목록에서 추천 목록을 생성

관심사가 유사한 사용자 목록

[(user4, 0.9), (user1, 0.3), (user3, 0.22)]

사용자-아이템 행렬
(users_interests)



	Item1	Item2	Item3	Item4	Item5
User1	0.3	0.3		0.3	0.3
User2					
User3		0.22	0.22	0.22	0.22
User4	0.9		0.9		0.9
합산	1.2	0.52	1.12	0.52	1.64

관심 item에 유사도 0.3을 적용

관심 item에 유사도 0.22을 적용

관심 item에 유사도 0.9을 더함

추천 목록

suggestions = {item1 : 1.2, item2 : 0.52, item3 : 1.12, item4 : 0.52, item5 : 1.64}

suggestions = {item5 : 1.64 , item1 : 1.2, item3 : 1.12, item2 : 0.52, item4 : 0.52,}

정렬

사용자 기반 추천

관심사가 유사한 사용자의 관심사 목록에서 추천 목록을 생성

관심사가 유사한 사용자의 관심 목록 작성

```
from collections import defaultdict

def user_based_suggestions(user_id: int,
                           include_current_interests: bool = False):
    # Sum up the similarities.
    suggestions: Dict[str, float] = defaultdict(float)
    for other_user_id, similarity in most_similar_users_to(user_id):
        for interest in users_interests[other_user_id]:
            suggestions[interest] += similarity
```

- include_current_interests : user_id의 관심 목록에 있는 것도 포함해서 추천할지 여부
- User_id와 가장 유사한 사용자 목록을 구함
 - 유사한 사용자들의 관심 목록을 하나의 추천 목록에 추가하면서 사용자 유사도를 더함

사용자 기반 추천

관심 목록을 유사도 순으로 정렬

```
# Convert them to a sorted list.
suggestions = sorted(suggestions.items(),
                      key=lambda pair: pair[-1], # weight
                      reverse=True)
```

사용자의 관심사와 중복된 관심사를 포함할지 여부

```
# And (maybe) exclude already-interests
if include_current_interests:
    return suggestions
else:
    return [(suggestion, weight)
            for suggestion, weight in suggestions
            if suggestion not in users_interests[user_id]]
```

- include_current_interests = True : 현재 suggestion을 그대로 전달
- include_current_interests = False : user_id의 관심 목록에 없는 것만 골라서 전달

사용자 기반 추천

사용자 0에게 추천할 관심사는?

```
ubs0 = user_based_suggestions(0)
```

```
[('MapReduce', 0.5669467095138409),  
(('MongoDB', 0.50709255283711),  
(('Postgres', 0.50709255283711),  
(('NoSQL', 0.3380617018914066),  
(('neural networks', 0.1889822365046136),  
(('deep learning', 0.1889822365046136),  
(('artificial intelligence', 0.1889822365046136),  
(('databases', 0.1690308509457033),  
(('MySQL', 0.1690308509457033),  
(('Python', 0.1543033499620919),  
(('R', 0.1543033499620919),  
(('C++', 0.1543033499620919),  
(('Haskell', 0.1543033499620919),  
(('programming languages', 0.1543033499620919))]
```

사용자0은 Big Data와 DB에 관심이 있으므로
추천 목록이 적절해 보임!


3. 아이템 기반 협업 필터링



사용자 기반 추천 방식의 한계

아마존과 같이 아이템 수 매우 많다면 사용자 기반 추천이 잘 작동하지 않는다.

아이템 개수가 매우 많은 경우 관심 항목 벡터가 희소해짐



	Item1	Item2	Item3	Item4	Item5
User1	O	O		O	O
User2		O		O	
User3		O	O	O	O
User4	O		O		O

아이템이 많아지면 관심 항목 벡터가 희소해져서
공통의 아이템 수가 작아지게 되고
사용자 유사도가 아주 작은 값으로 계산됨

아이템 기반 추천

비슷한 관심을 갖는 사용자의 행동을 찾을 때
사용자 관심사의 유사도가 아닌 아이템 사용자 군의 유사도로 계산해보자!

사용자-아이템 행렬 (user_interest_vectors)

	Item1	Item2	Item3	Item4	Item5
User1	1	1	0	1	1
User2	0	1	0	1	0
User3	0	1	1	1	1
User4	1	0	1	0	1

사용자-아이템
행렬을 전치

아이템-사용자 행렬 (interest_user_matrix)

	User1	User2	User3	User4
Item1	1	0	0	1
Item2	1	1	1	0
Item3	0	0	1	1
Item4	1	1	1	0
Item5	1	0	1	1

아이템-사용자 행렬

아이템-사용자 행렬

```
interest_user_matrix = [[user_interest_vector[j]  
                          for user_interest_vector in user_interest_vectors]  
                          for j, _ in enumerate(unique_interests)]
```

- user_interest_vectors 행렬을 전치하기 위해 j번째 컬럼을 모아서 하나의 열을 만듦

아이템 0의 사용자 벡터

```
print(interest_user_matrix[0])
```

[1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0]

아이템 유사도 행렬

아이템 별로 다른 모든 아이템과의 유사도를 측정한 행렬

아이템-사용자 행렬 (interest_user_matrix)

	User1	User2	User3	User4
Item1	1	0	0	1
Item2	1	1	1	0
Item3	0	0	1	1
Item4	1	1	1	0
Item5	1	0	1	1



아이템 유사도 행렬 (interest_similarities)

	Item1	Item2	Item3	Item4	Item5
Item1	1.0	0.12	0.42	0.5	0.7
Item2	0.12	1.0	0.37	0.83	0.2
Item3	0.42	0.37	1.0	0.38	0.29
Item4	0.5	0.83	0.38	1.0	0.84
Item5	0.7	0.2	0.29	0.84	1.0

- 대칭 행렬
- 대각은 자기 자신과의 유사도로 1

아이템 유사도 행렬

```
interest_similarities = [[cosine_similarity(user_vector_i, user_vector_j)
                           for user_vector_j in interest_user_matrix]
                           for user_vector_i in interest_user_matrix]
```

사용자 군이 유사한 아이템 목록

아이템 유사도 행렬을 이용해서 사용자 군이 유사한 아이템 목록을 구성

```
def most_similar_interests_to(interest_id: int):  
    similarities = interest_similarities[interest_id]  
    pairs = [(unique_interests[other_interest_id], similarity)  
             for other_interest_id, similarity in enumerate(similarities)  
             if interest_id != other_interest_id and similarity > 0]  
    return sorted(pairs,  
                  key=lambda pair: pair[-1],  
                  reverse=True)
```

- interest_id의 유사도 중 유사도가 0이상인 것만 골라 냄
- 유사도 순으로 정렬

interest_id →

	Item1	Item2	Item3	Item4	Item5
Item1	1.0	0.12	0.42	0.5	0.7
Item2	0.12	1.0	0.37	0.83	0.2
Item3	0.42	0.37	1.0	0.38	0.29
Item4	0.5	0.83	0.38	1.0	0.84
Item5	0.7	0.2	0.29	0.84	1.0

아이템 유사도 행렬 (similarities)

다른 항목과의 유사도 목록

pairs = [(item1, 0.12) , (item3, 0.37), (item4, 0.83) , (item5, 0.2)]

↓ 유사도 순으로 정렬

[((item4, 0.83), (item3, 0.37), (item5, 0.2), (item1, 0.12))]

사용자 군이 유사한 아이템 목록

아이템 0과 가장 유사한 아이템은 무엇일까?

```
msit0 = most_similar_interests_to(0)
```

```
[('Hadoop', 0.8164965809277261),  
( 'Java', 0.6666666666666666),  
( 'MapReduce', 0.5773502691896258),  
( 'Spark', 0.5773502691896258),  
( 'Storm', 0.5773502691896258),  
( 'Cassandra', 0.4082482904638631),  
( 'artificial intelligence', 0.4082482904638631),  
( 'deep learning', 0.4082482904638631),  
( 'neural networks', 0.4082482904638631),  
( 'HBase', 0.3333333333333333)]
```

Hadoop와 가장 유사하며
유사도는 81%

아이템 기반 추천

사용자의 관심사와 유사한 관심 아이템으로 추천 목록을 생성

사용자-관심 벡터
(user_interest_vectors)

user-id →

	Item1	Item2	Item3	Item4	Item5
User1	1	1	0	1	1
User2	0	1	0	1	0
User3	0	1	1	1	1
User4	1	0	1	0	1

유사한 관심 목록
(similar_interests)

[((item4, 0.83), (item3, 0.37), (item5, 0.2), (item1, 0.12))]
 [((item5, 0.84), (item2, 0.83), (item1, 0.5), (item3, 0.38))]

	Item1	Item2	Item3	Item4	Item5
Item1	1.0	0.12	0.42	0.5	0.7
Item2	0.12	1.0	0.37	0.83	0.2
Item3	0.42	0.37	1.0	0.38	0.29
Item4	0.5	0.83	0.38	1.0	0.84
Item5	0.7	0.2	0.29	0.84	1.0

Item2와 유사한 아이템 목록 구하기
 item4와 유사한 아이템 목록 구하기

↓ 같은 항목의 유사도는 더함

추천 목록 suggestions = {item1 : 1.7, item2 : 0.83, item3 : 0.76, item4 : 0.83, item5 : 1.04}

suggestions = {item1 : 1.7, item5 : 1.04, item2 : 0.83, item4 : 0.83, item3 : 0.76,}

정렬

1 5 3 순으로 추천

아이템 기반 추천

사용자 군이 유사한 아이템으로 추천 목록 작성

```
def item_based_suggestions(user_id: int,
                           include_current_interests: bool = False):
    # Add up the similar interests
    suggestions = defaultdict(float)
    user_interest_vector = user_interest_vectors[user_id]
    for interest_id, is_interested in enumerate(user_interest_vector):
        if is_interested == 1:
            similar_interests = most_similar_interests_to(interest_id)
            for interest, similarity in similar_interests:
                suggestions[interest] += similarity
```

→ 인덱스 변화와
경향성 변화를 tuple로
피로 변환

- include_current_interests : user_id의 관심 목록에 있는 것도 포함해서 추천할지 여부
- user_id의 관심 목록에 있는 아이템 별로 유사한 아이템 목록(similar_interests)을 구함
 - 유사한 아이템 목록을 추천 목록에 추가하면서 아이템의 유사도를 더함

아이템 기반 추천

추천 목록을 유사도 순으로 정렬

```
# Sort them by weight
suggestions = sorted(suggestions.items(),
                     key=lambda pair: pair[-1],
                     reverse=True)
```

user_id 추천 목록 필터링

```
if include_current_interests:
    return suggestions
else:
    return [(suggestion, weight)
            for suggestion, weight in suggestions
            if suggestion not in users_interests[user_id]]
```

- include_current_interests = True : 현재 suggestion을 그대로 전달
- include_current_interests = False : user_id의 관심 목록에 없는 것만 골라서 전달

아이템 기반 추천

사용자 0에게 추천할 관심사는?

```
ibs0 = item_based_suggestions(0)
```

```
[('MapReduce', 1.861807319565799),  
(('MongoDB', 1.3164965809277263),  
(('Postgres', 1.3164965809277263),  
(('NoSQL', 1.2844570503761732),  
(('MySQL', 0.5773502691896258),  
(('databases', 0.5773502691896258),  
(('Haskell', 0.5773502691896258),  
(('programming languages', 0.5773502691896258),  
(('artificial intelligence', 0.4082482904638631),  
(('deep learning', 0.4082482904638631),  
(('neural networks', 0.4082482904638631),  
(('C++', 0.4082482904638631),  
(('Python', 0.2886751345948129),  
(('R', 0.2886751345948129))]
```

```
ubs0 = user_based_suggestions(0)
```

```
[('MapReduce', 0.5669467095138409),  
(('MongoDB', 0.50709255283711),  
(('Postgres', 0.50709255283711),  
(('NoSQL', 0.3380617018914066),  
(('neural networks', 0.1889822365046136),  
(('deep learning', 0.1889822365046136),  
(('artificial intelligence', 0.1889822365046136),  
(('databases', 0.1690308509457033),  
(('MySQL', 0.1690308509457033),  
(('Python', 0.1543033499620919),  
(('R', 0.1543033499620919),  
(('C++', 0.1543033499620919),  
(('Haskell', 0.1543033499620919),  
(('programming languages', 0.1543033499620919))]
```

아이템 기반의 추천과 사용자 기반의 추천과 유사함

Thank you!

