협업 필터링을 활용한 추천 시스템 개발

협업 필터링(Collaborative Filtering)은 추천 시스템에서 가장 널리 사용되는 방법 중 하나로, 사용자들의 과거 행동이나 선호도를 바탕으로 다른 사용자가 좋아할 만한 항목을 추천하는 방법입니다. 협업 필터링은 크게 두 가지 방식으로 나뉩니다: 사용자 기반 협업 필터링과 아이템 기반 협업 필터링입니다.

1. 코사인 유사도와 협업 필터링

코사인 유사도는 두 벡터 간의 유사도를 측정하는 방법으로, 협업 필터링에서 사용자나 아이템 간의 유사도를 계산하는 데 자주 사용됩니다. 코사인 유사도는 벡터 간의 각도를 기반으로 유사도를 측정하며, 두 벡터가 완전히 동일한 방향일 때 1에 가까운 값을 가지며, 완전히 다른 방향일 때 -1에 가까운 값을 가집니다.
Cosine Similarity = (\cos(\theta)) = (\frac{A \cdot B}{|A| \times |B|})

2. 사용자 기반 협업 필터링 (User-based Collaborative Filtering)

이 방식에서는 사용자의 평점 데이터를 기반으로 유사한 사용자 그룹을 찾습니다. 예를 들어, Kim 사용자가 특정 영화를 아직 보지 않았다고 할 때, Kim과 유사한 다른 사용자들이 해당 영화를 좋아했다면, 그 영화를 Kim에게 추천할 수 있습니다.
유사한 사용자 찾기: Kim과 다른 사용자들의 평점을 코사인 유사도를 이용해 비교하여 Kim과 유사한 사용자들을 찾습니다. 추천 계산: 유사한 사용자들이 높은 점수를 준 아이템(영화 등)을 Kim에게 추천합니다.

3. 아이템 기반 협업 필터링 (Item-based Collaborative Filtering)

이 방식에서는 아이템 간의 유사도를 바탕으로 추천을 수행합니다. 예를 들어, Kim이 특정 영화를 좋아했다면, 그 영화와 유사한 다른 영화들을 추천할 수 있습니다.
유사한 아이템 찾기: 각 영화 간의 유사도를 코사인 유사도를 통해 계산합니다. 추천 계산: 사용자가 평가한 아이템들과 유사한 아이템을 찾아 추천합니다

4. 코사인 유사도를 이용한 추천 프로세스

1. 데이터 수집: 사용자들이 아이템에 대해 평가한 평점 데이터가 필요합니다.
2. 유사도 계산: 코사인 유사도를 이용해 사용자 간 또는 아이템 간의 유사도를 계산합니다.
3. 가중 평균 계산: 유사도를 가중치로 하여 결측값(즉, 사용자가 아직 평가하지 않은 아이템에 대한 예측값)을 계산합니다.
4. 추천: 예측된 평점이 높은 아이템을 사용자에게 추천합니다.

5. 예시 데이터의 추천 프로세스 설계

1. 로딩되는 예시 데이터

사용자	아이템 1	아이템 2	아이템 3	아이템 4	아이템 5	아이템 6
Kim	5	1	4	4	?	?

사용자	아이템 1	아이템 2	아이템 3	아이템 4	아이템 5	아이템 6
Lee	3	1	2	2	3	2
Park	4	2	4	5	5	1
Choi	3	3	1	5	4	3
Kwon	1	5	5	2	1	4

2. 계산된 코사인 유사도

	A	B	C	D	E	F	코사인 유사도
kim	5	1	4	4			
lee	3	1	2	2	3	2	0.990375137
park	4	2	4	5	5	1	0.975099827
choi	3	3	1	5	4	3	0.831397962
kwon	1	5	5	2	1	4	0.67280352

3. 결측치에 대한 결측 대상

사용자	아이템 1	아이템 2	아이템 3	아이템 4	아이템 5	아이템 6
Kim	5	1	4	4	?	?

4. Kim의 평점 예측치

	A	B	C	D	E	F
kim	5	1	4	4	**3.397**	**2.36254**
lee	3	1	2	2	3	2
park	4	2	4	5	5	1
choi	3	3	1	5	4	3
kwon	1	5	5	2	1	4

5. 결과 출력

```
Kim의 다섯 번째 값: 3.39702127399391
Kim의 여섯 번째 값: 2.3625384640129212
추천: E
```

1. Kim의 아이템 5와 6에 대한 평점을 데이터를 로딩합니다.
2. Kim과 다른 사용자들 (Lee, Park, Choi, Kwon) 간의 유사도를 계산합니다.
3. 계산된 유사도를 가중치로 사용하여 결측값을 계산합니다.
4. Lee, Park, Choi, Kwon의 아이템 5와 6에 대한 평가를 바탕으로 Kim의 평점을 예측합니다.
5. Kim의 다섯 번째와 여섯 번째 값에 대한 레이블 결정하여 결과를 출력하도록 합니다.

6. 협업 필터링을 활용한 추천 시스템 프로그래밍

```
# 코사인 유사도에 의한 추천 시스템
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# 1. 사용자 벡터 정의 (Kim의 다섯 번째, 여섯 번째 값은 0으로 초기화)
kim = np.array([5, 1, 4, 4, 0, 0])
lee = np.array([3, 1, 2, 2, 3, 2])
park = np.array([4, 2, 4, 5, 5, 1])
choi = np.array([3, 3, 1, 5, 4, 3])
kwon = np.array([1, 5, 5, 2, 1, 4])

# 모든 사용자 벡터를 모아서 행렬로 만듦
users = np.array([kim, lee, park, choi, kwon])

# 2. Kim과 다른 사용자 간의 코사인 유사도 계산
cos_sim = cosine_similarity(users)[0, 1:]

# 3. 각 항목의 값을 유사도로 가중 평균하여 예측
def predict_missing_value(kim, others, cos_sim):
    predictions = []
    for i in range(len(kim)):
        if kim[i] == 0:  # 값이 비어 있는 항목에 대해서만 예측
            weighted_sum = np.sum(others[:, i] * cos_sim)
            sum_of_weights = np.sum(cos_sim)
            prediction = weighted_sum / sum_of_weights if sum_of_weights != 0 else 0
            predictions.append(prediction)
        else:
            predictions.append(kim[i])
    return predictions

# 다른 사용자의 벡터 모음 (Kim 제외)
other_users = users[1:]

# 4. Kim의 빈 값을 예측
kim_filled = predict_missing_value(kim, other_users, cos_sim)

# 5. Kim의 다섯 번째와 여섯 번째 값에 대한 레이블 결정
fifth_value = kim_filled[4]
sixth_value = kim_filled[5]

if fifth_value > sixth_value:
    label = "E"
else:
    label = "F"

# 결과 출력
print(f"Kim의 다섯 번째 값: {fifth_value}")
print(f"Kim의 여섯 번째 값: {sixth_value}")
print(f"추천: {label}")
```

```
Kim의 다섯 번째 값: 3.39702127399391
Kim의 여섯 번째 값: 2.3625384640129212
추천: E
```

7. KNN(K 최근접 이웃) 에 의한 추천 시스템 프로그래밍

```
# KNN(최근접이웃) 에 의한 추천 시스템
import numpy as np
from sklearn.impute import KNNImputer

# 사용자 벡터 정의 (Kim의 다섯 번째, 여섯 번째 값은 결측값으로 처리)
kim = np.array([5, 1, 4, 4, np.nan, np.nan])
lee = np.array([3, 1, 2, 2, 3, 2])
park = np.array([4, 2, 4, 5, 5, 1])
choi = np.array([3, 3, 1, 5, 4, 3])
kwon = np.array([1, 5, 5, 2, 1, 4])

# 모든 사용자 벡터를 모아서 행렬로 만듦
users = np.array([kim, lee, park, choi, kwon])

# KNNImputer를 사용해 결측값 채우기
knn_imputer = KNNImputer(n_neighbors=2, weights='uniform')  # 가까운 2명의 이웃을 사용
users_filled = knn_imputer.fit_transform(users)

# Kim의 채워진 벡터 확인
kim_filled = users_filled[0]

# Kim의 다섯 번째와 여섯 번째 값에 대한 레이블 결정
fifth_value = kim_filled[4]
sixth_value = kim_filled[5]

if fifth_value > sixth_value:
    label = "E"
else:
    label = "F"

# 결과 출력
print(f"Kim의 다섯 번째 값: {fifth_value}")
print(f"Kim의 여섯 번째 값: {sixth_value}")
print(f"추천: {label}")
Kim의 다섯 번째 값: 4.0
Kim의 여섯 번째 값: 1.5
추천: E
```