Machine Learning HW4

202211167 장지원

[Setting]

1. **Arrange the Result**

   해당 링크의 코드를 활용해 result를 준비해 보았다.

   recommender-tutorial/part-1-item-item-recommender.ipynb at master · topspinj/recommender-tutorial (github.com)

2. **Show how the results differ based on the distance metrics.**

   Distance metric을 Cosine similarity, Euclidean distance, Manhattan distance으로 바꿔가면서 결과를 출력해 보았다. 결과는 다음 표와 같았다. 각 metric마다 결과가 조금씩 다르게 나옴을 확인해 볼 수 있다.

Distance Meric	Result
Cosine Similarity	Because you watched Toy Story (1995) Toy Story 2 (1999) Jurassic Park (1993) Independence Day (a.k.a. ID4) (1996) Star Wars: Episode IV - A New Hope (1977) Forrest Gump (1994) Lion King, The (1994) Star Wars: Episode VI - Return of the Jedi (1983) Mission: Impossible (1996) Groundhog Day (1993) Back to the Future (1985)
Euclidean distance	Because you watched Toy Story (1995): Toy Story 2 (1999) Mission: Impossible (1996) Independence Day (a.k.a. ID4) (1996) Bug's Life, A (1998) Nutty Professor, The (1996) Willy Wonka & the Chocolate Factory (1971) Babe (1995) Groundhog Day (1993) Mask, The (1994) Honey, I Shrunk the Kids (1989)
Manhattan distance	Because you watched Toy Story (1995): Toy Story 2 (1999) Bug's Life, A (1998) Groundhog Day (1993) Nutty Professor, The (1996) Willy Wonka & the Chocolate Factory (1971) Mission: Impossible (1996) Babe (1995) Monsters, Inc. (2001) Toy Story 3 (2010) Honey, I Shrunk the Kids (1989)

3. **Instead of K-NN, implement matrix factorization using Singular Value Decomposition (SVD) for**

collaborative filtering. Show how the results differ. If SVD does not work due to the large size of the matrix, check the results using only a subset of the matrix.

KNN 대신 SVD를 이용하여 collaborative filtering을 진행해 보았다. GitHub의 코드를 다음과 같이 수정하였다. *(matrix를 너무 크게 하면 연산 오류가 생긴다. 따라서 subset을 가지고 연산을 진행했다)*

```
def apply_svd(X, n_components=100):
    svd = TruncatedSVD(n_components=n_components, random_state=42)
    X_svd = svd.fit_transform(X)
    return X_svd

X_svd = apply_svd(X)

def calculate_similarity(X_svd):
    sim_matrix = cosine_similarity(X_svd)
    return sim_matrix

similarity_matrix = calculate_similarity(X_svd)

def find_similar_movies(movie_id, similarity_matrix, movie_mapper,
movie_inv_mapper, k=10):
    movie_idx = movie_mapper[movie_id]
    similar_movies = np.argsort(similarity_matrix[movie_idx])[::-1][:k]
    similar_movies = [movie_inv_mapper[i] for i in similar_movies]
    return similar_movies
```

이 때는 다음과 같은 결과가 나왔다.

```
Because you watched Toy Story (1995):
Toy Story (1995)
My Crazy Life (Mi vida loca) (1993)
Harriet the Spy (1996)
Very Brady Sequel, A (1996)
Bloodsport 2 (a.k.a. Bloodsport II: The Next Kumite) (1996)
Fear (1996)
Beverly Hills Ninja (1997)
Beavis and Butt-Head Do America (1996)
Only Yesterday (Omohide poro poro) (1991)
It's Such a Beautiful Day (2012)
```

4. Implement matrix factorization using Alternating Least Square (ALS) for collaborative filtering. Show how the results differ.

ALS를 이용하여 collaborative filtering을 진행해 보았다.

GitHub의 코드를 다음과 같이 수정하였다.

```
def apply_als(X, factors=20, regularization=0.1, iterations=15):
    model = AlternatingLeastSquares(factors=factors, regularization=regularization,
iterations=iterations)
    model.fit(X)
    return model

model = apply_als(X)

def calculate_similarity(model, X):
```

```
    user_factors = model.user_factors
    item_factors = model.item_factors
    sim_matrix = cosine_similarity(item_factors)
    return sim_matrix

similarity_matrix = calculate_similarity(model, X)

def find_similar_movies(movie_id, similarity_matrix, movie_mapper,
movie_inv_mapper, k=10):
    movie_idx = movie_mapper[movie_id]
    similar_movies = np.argsort(similarity_matrix[movie_idx])[::-1][:k]
    similar_movies = [movie_inv_mapper[i] for i in similar_movies]
    return similar_movies
```

이 때는 다음과 같은 결과가 나왔다.

```
Because you watched Toy Story (1995):
Toy Story (1995)
To Catch a Thief (1955)
Friday (1995)
Ran (1985)
African Queen, The (1951)
Army of Darkness (1993)
Quiz Show (1994)
Ace Ventura: When Nature Calls (1995)
Faster Pussycat! Kill! Kill! (1965)
First Knight (1995)
```

ALS는 대규모 모델에서 잘 작동한다. 추천이 잘 되었는 지에 대한 직관적인 판단을 불가능하지만, SVD, KNN서 보다 조금 더 잘 작동되었을 것임을 예측해 볼 수는 있다.