CptS 315 - HW 2

Zach Fechko (011711215)

October 16, 2022

Q1

Consider the following ratings matrix with 3 users and 6 items. Ratings are on a 1-5 star scale. Compute the following from this matrix:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	4	5		5	1	
User 2		3	4	3	1	2
User 3	2		1	3		4

a. Treat missing values as 0. Compute the jaccard similarity between each pair of users
b. Treat missing values as 0. Compute the cosine similarity between each pair of users
c. Normalize the matrix by subtracting from each non-zero rating, the average value for its user. Show the normalized matrix.
d. Compute the (centered) cosine similarity between each pair of users using the normalized matrix.

Q1 Solution

a.

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

$J(1,2) = \frac{3}{6}$
$J(1,3) = \frac{1}{6}$
$J(2,3) = \frac{3}{6}$

b.

$C(A,B) = \frac{A \cdot B}{\|A\| \|B\|}$

```python
from numpy.linalg import norm

def cosine(a: list, b: list):
    """
    Calculates the cosine similarity between two lists
    """
```

```
    A = np.array(a)
    B = np.array(b)
    return np.dot(A, B) / (norm(A) * norm(B))

print("Users 1 & 2: ", cosine(list(ratings.loc["User 1"]), list(ratings.loc["User 2"]))) #cosine simila
print("Users 1 & 3: ", cosine(list(ratings.loc["User 1"]), list(ratings.loc["User 3"]))) #cosine simila
print("Users 2 & 3: ", cosine(list(ratings.loc["User 2"]), list(ratings.loc["User 3"]))) #cosine simila
```

```
## Users 1 & 2:  0.6064457948612227
## Users 1 & 3:  0.5130146972572911
## Users 2 & 3:  0.6139406135149204
```

c.

```
# Normalize the matrix by subtracting from each non-zero rating, the average value for its user. Show t
ratings = ratings.replace(0, np.nan)
ratings = ratings.subtract(ratings.mean(axis=1), axis=0)
norm_ratings = ratings.fillna(0)

print(norm_ratings)
```

```
##         Item 1  Item 2  Item 3  Item 4  Item 5  Item 6
## User
## User 1    0.25    1.25     0.0    1.25   -2.75     0.0
## User 2    0.00    0.40     1.4    0.40   -1.60    -0.6
## User 3   -0.50    0.00    -1.5    0.50    0.00     1.5
```

d.

Using the `norm_ratings` df from part c and the `cosine` function from part b, we can calculate the cosine similarity between each pair of users.

```
# Compute the (centered) cosine similarity between each pair of users using the normalized matrix.
print("Users 1 & 2: ", cosine(list(norm_ratings.loc["User 1"]), list(norm_ratings.loc["User 2"]))) #cos
print("Users 1 & 3: ", cosine(list(norm_ratings.loc["User 1"]), list(norm_ratings.loc["User 3"]))) #cos
print("Users 2 & 3: ", cosine(list(norm_ratings.loc["User 2"]), list(norm_ratings.loc["User 3"]))) #cos
```

```
## Users 1 & 2:  0.7222505078325668
## Users 1 & 3:  0.06819943394704735
## Users 2 & 3:  -0.5491251783869152
```

Q2

Read the following two papers and write a brief summary of the main points in at most **TWO** pages.
Two Decades of Recommender Systems at Amazon.com
Industry Report: Amazon.com Recommendations: Item-to-Item

Q2 Solution

Two Decades of Recommender Systems at Amazon.com

This article talks about how Amazon has been using recommender systems for over 20 years. It starts out by describing the first collaborative filtering algorithm that Amazon used, which wasn't too accurate. The algorithm at the time was generally user based, meaning that it would search across what other users had ordered and recommend items that those similar users had ordered. Nowadays the algorithm recommends on an item basis, for example if you ordered a set of headphones, the algorithm might recommend other pairs of headphones. As time went on, other online services and retatilers started using an algorithm similar to Amazon's, youtube used it to recommend videos, and Netflix used it to recommend movies to their users. As the years went on, the recommendendations were used so extensively by users that Microsoft estimated that 30% of Amazon's page views came from recommendations. Through use of machine learning, Amazon and other companies were able to improve the accuracy of their recommendations, constantly refining the parameters that influence what gets recommended to the user. Time is also an important fact in recommendations. For example, if a user buys a camera, it makes sense to recommend a memory card to go with it, but if they buy the memory card first, then it doesn't make sense to recommend the camera.

Industry Report: Amazon.com Recommendations: Item-to-Item

This article describes what goes into making a recommendation on Amazon. It describes what the algorithm is, and the pieces that make it work. The algorithm makes use of collaborative filtering and cluster models. Collaborative filtering represents a customer as an n-dimensional vector, where n is the number of items in the catalog. The components of this vector are the ratings that the customer has given to each item, positive for positvely rated items, and negative for negatively rated items. The algorithm then uses this vector to find similar customers, and recommends items through methods like cosine similarity, which is the dot product of the two vectors divided by the product of the norms of the vectors.

$$C(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Using collaborative filtering is pretty expensive computationally, running in $O(MN)$ time, where M is the number of users and N is the number of items. To reduce the computational cost, the algorithm makes use of a cluster model. Which clusters users into segments, these have better online scalability because you're comparing a user to a controlled number of segments rather than the entire user base. Amazon's recommendations have evolved from comparing between users to comparing between items. Instead of matching the user to similar customers, the algorithm matches each of the purchased items to similar items and computes the similarity between each item pair.