University of Salford MANCHESTER

Artificial Intelligent – Week 12

Dr Salem Ameen

S.A.AMEEN1@SALFORD.AC.UK

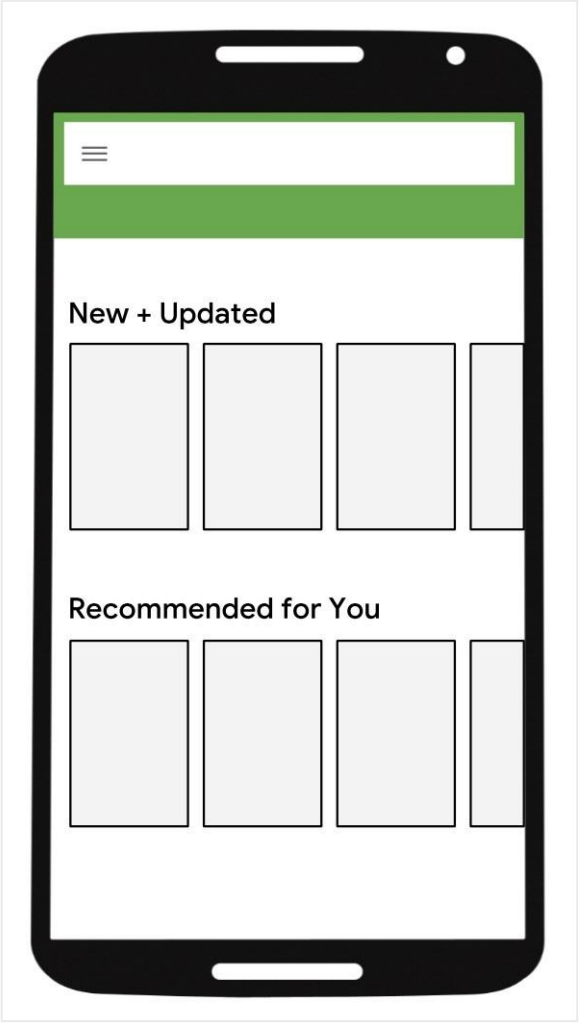
Last week :

MAB
Contextual Bandits
RL

Week 12 :Recommendation System

University of Salford MANCHESTER

Personalization is transforming our experience of the world



Week 12 :Recommendation System

University of Salford MANCHESTER

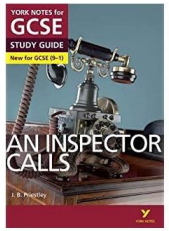
Personalization is transforming our experience of the world

Gift ideas inspired by your shopping history

Page 1 of 7

PowerA Joy-Con Comfort Grip for...
PowerA
4.6 out of 5 stars 20,112
Nintendo Switch
#1 Best Seller in Nintendo Switch Hand Grips
£6.99
prime Today by 7PM

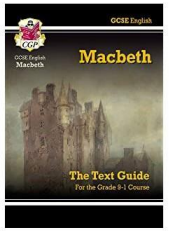
PowerA Charging Station for Nintendo Switch...
PowerA
4.6 out of 5 stars 10,435
Nintendo Switch
#1 Best Seller in Nintendo Switch Chargers
£14.89
prime Today by 7PM



An Inspector Calls: York Notes for GCSE (9-1)
› John Scicluna
4.6 out of 5 stars 991
Paperback
£3.99
prime FREE One-Day

Mario Kart 8 Deluxe (Nintendo Switch)
Nintendo
4.8 out of 5 stars 25,493
Nintendo Switch
£36.99
prime Today by 7PM

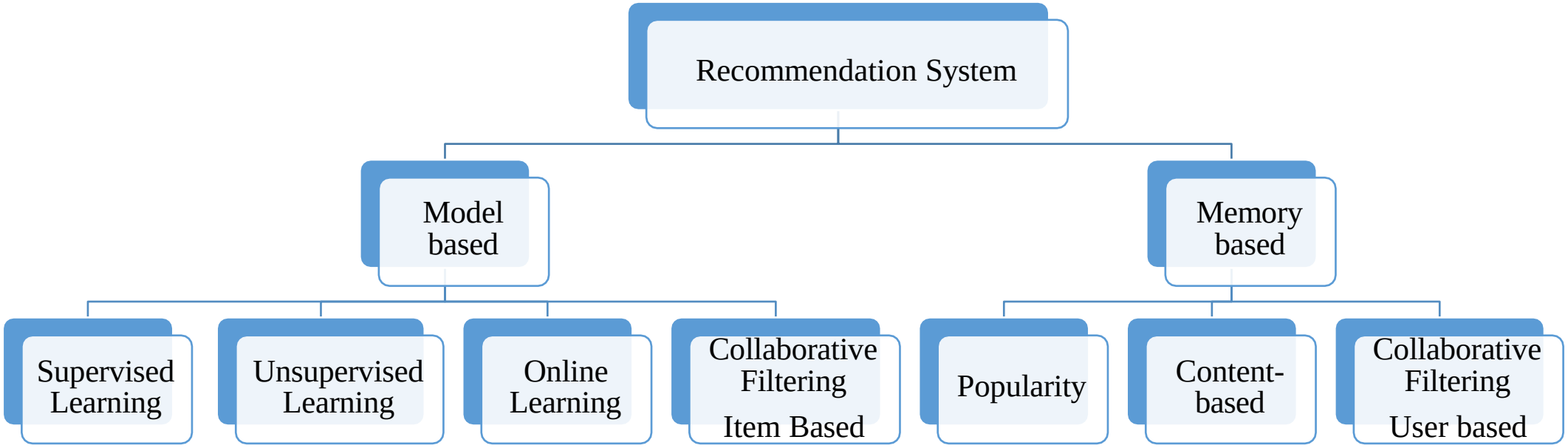
Carolina Reaper Peanuts - Hot as Hell Seasoned Peanuts 80g
4.6 out of 5 stars 1,982
£2.69



Grade 9-1 GCSE English Shakespeare Text...
› CGP Books
4.6 out of 5 stars 2,483
Paperback
#1 Best Seller in Welsh
£3.41
prime Today by 7PM

Joy-Con Pair Green/Pink (Nintendo Switch)
Nintendo
4.8 out of 5 stars 53,031
Nintendo Switch
#1 Best Seller in Nintendo Switch Controllers
£58.00
prime Today by 7PM

Recommendation System



Week 12 :Content-based Recommendation System

University of Salford MANCHESTER

What is a "**Content-based**" recommender system?

Content-based technique tries to figure out what a user's favourite aspects of an item is, and then recommends items that present those aspects.

Week 12 :Content-based Recommendation System

Week 12 :Content-based Recommendation System

Week 12 :Content-based Recommendation System - Example

(Comedy, Super Hero)

Batman begins

(Super Hero)

Week 12 :Content-based Recommendation System - Example

Name of the movie

Genre - category

(Comedy, Super Hero)

Batman begins

(Super Hero)

Week 12 :Content-based Recommendation System

Batman v Superman  (Adventure, Super Hero)

Guardians of the Galaxy  (Comedy, Adventure, Super Hero, Sci-Fi)

Captain America: Civil War  (Comedy, Super Hero)

Hitchhiker's guide to the galaxy  (Comedy, Adventure, Sci-Fi)

Batman begins  (Super Hero)

Spiderman  (Comedy, Super Hero)

3

Week 12 :Content-based Recommendation System

Week 12 :Content-based Recommendation System

Week 12 :Content-based RS- Budling User Profile

Week 12 :Content-based RS Budling User Profile / Analysis

Week 12 :Content-based RS Budling User Profile / Analysis

University of Salford MANCHESTER

```
#Storing the movie information into a pandas dataframe
movies_df = pd.read_csv('movies.csv')
#Storing the user information into a pandas dataframe
ratings_df = pd.read_csv('ratings.csv')
```

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

```
movies_df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
ratings_df.head()
```

	userId	movieId	rating	timestamp
0	1	169	2.5	1204927694
1	1	2471	3.0	1204927438
2	1	48516	5.0	1204927435
3	2	2571	3.5	1436165433
4	2	109487	4.0	1436165496

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

```
movies_df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
#Using regular expressions to find a year stored between parentheses
#We specify the parantheses so we don't conflict with movies that have years in their titles
movies_df['year'] = movies_df.title.str.extract('(\(\d\d\d\d\))',expand=False)
```

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

```
movies_df.head()
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

	movieId	title	genres	year
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	(1995)
1	2	Jumanji (1995)	Adventure|Children|Fantasy	(1995)
2	3	Grumpier Old Men (1995)	Comedy|Romance	(1995)
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	(1995)
4	5	Father of the Bride Part II (1995)	Comedy	(1995)

```
#Using regular expressions to find a year stored between parentheses
#We specify the parantheses so we don't conflict with movies that have years in their titles
movies_df['year'] = movies_df.title.str.extract('(\(\d\d\d\d\))',expand=False)
```

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

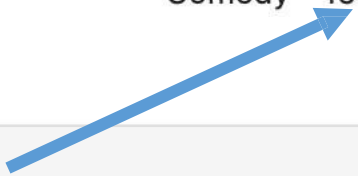
	movieId	title	genres	year
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	(1995)
1	2	Jumanji (1995)	Adventure|Children|Fantasy	(1995)
2	3	Grumpier Old Men (1995)	Comedy|Romance	(1995)
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	(1995)
4	5	Father of the Bride Part II (1995)	Comedy	(1995)

```
#Removing the parentheses
movies_df['year'] = movies_df.year.str.extract('(\d\d\d\d)',expand=False)
movies_df.head()
```

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

	movieId	title	genres	year
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji (1995)	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men (1995)	Comedy|Romance	1995
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II (1995)	Comedy	1995



```
#Removing the parentheses
movies_df['year'] = movies_df.year.str.extract('(\d\d\d\d)',expand=False)
movies_df.head()
```

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

```
#Removing the years from the 'title' column
movies_df['title'] = movies_df.title.str.replace('(\(\d\d\d\d\))', '')
movies_df.head()
```

```
<ipython-input-15-be57492d82d5>:2: FutureWarning: The default value of regex will change from True t
  movies_df['title'] = movies_df.title.str.replace('(\(\d\d\d\d\))', '')
```

	movieId	title	genres	year
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men	Comedy|Romance	1995
3	4	Waiting to Exhale	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II	Comedy	1995

```
#Applying the strip function to get rid of any ending whitespace characters that may have appeared
movies_df['title'] = movies_df['title'].apply(lambda x: x.strip())
movies_df.head()
```

	movieId	title	genres	year
0	1	Toy Story	Adventure|Animation|Children|Comedy|Fantasy	1995
1	2	Jumanji	Adventure|Children|Fantasy	1995
2	3	Grumpier Old Men	Comedy|Romance	1995
3	4	Waiting to Exhale	Comedy|Drama|Romance	1995
4	5	Father of the Bride Part II	Comedy	1995

Week 12 :Content-based RS UP/Weighting the Genre/Coding

University of Salford MANCHESTER

```
#Every genre is separated by a | so we simply have to call the split function on |
movies_df['genres'] = movies_df.genres.str.split('|')
movies_df.head()
```

	movieId	title	genres	year
0	1	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
1	2	Jumanji	[Adventure, Children, Fantasy]	1995
2	3	Grumpier Old Men	[Comedy, Romance]	1995
3	4	Waiting to Exhale	[Comedy, Drama, Romance]	1995
4	5	Father of the Bride Part II	[Comedy]	1995

Week 12 :Content-based RS
Hot Encoding technique / coding

University of Salford MANCHESTER

```
#Copying the movie dataframe into a new one since we won't need to use the genre information in our first case.
moviesWithGenres_df = movies_df.copy()

#For every row in the dataframe, iterate through the list of genres and place a 1 into the corresponding column
for index, row in movies_df.iterrows():
    for genre in row['genres']:
        moviesWithGenres_df.at[index, genre] = 1
#Filling in the NaN values with 0 to show that a movie doesn't have that column's genre
moviesWithGenres_df = moviesWithGenres_df.fillna(0)
```

genres
[Adventure, Animation, Children, Comedy, Fantasy]
[Adventure, Children, Fantasy]
[Comedy, Romance]
[Comedy, Drama, Romance]
[Comedy]

Week 12 :Content-based RS
Hot Encoding technique / coding

genres
[Adventure, Animation, Children, Comedy, Fantasy]
[Adventure, Children, Fantasy]
[Comedy, Romance]
[Comedy, Drama, Romance]
[Comedy]

Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IMAX	Documentary	War	Musical	Western	Film-Noir	(no genres listed)
1.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Week 12 :Content-based RS User Profile

Input User Ratings

```
userInput = [
            {'title':'Breakfast Club, The', 'rating':5},
            {'title':'Toy Story', 'rating':3.5},
            {'title':'Jumanji', 'rating':2},
            {'title':"Pulp Fiction", 'rating':5},
            {'title':'Akira', 'rating':4.5}
        ]
inputMovies = pd.DataFrame(userInput)
inputMovies
```

	title	rating
0	Breakfast Club, The	5.0
1	Toy Story	3.5
2	Jumanji	2.0
3	Pulp Fiction	5.0
4	Akira	4.5

Week 12 :Content-based RS Hot Encoding technique

	2
	10
	8

Input User Ratings

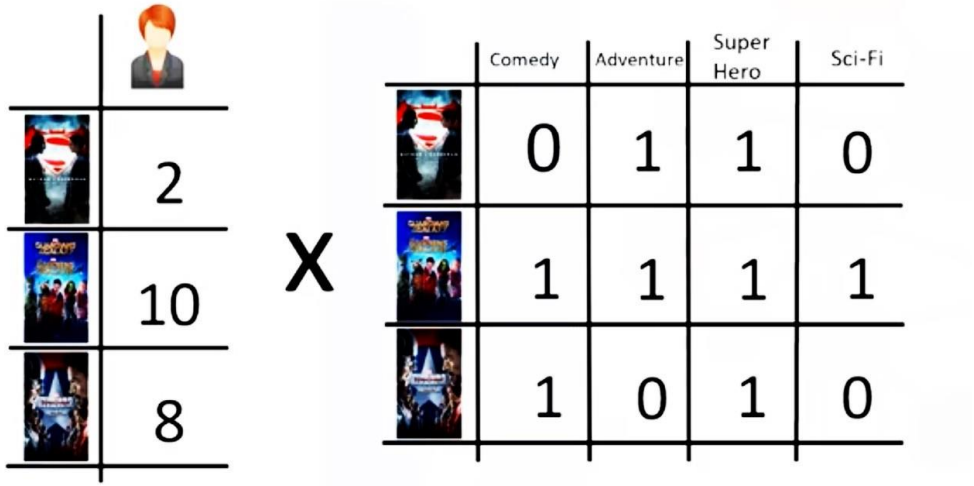
	Comedy	Adventure	Super Hero	Sci-Fi
	0	1	1	0
	1	1	1	1
	1	0	1	0

Movies Matrix

title	rating
Toy Story (1995)	2.5
Jumanji (1995)	3.0
Grumpier Old Men (1995)	5.0
Waiting to Exhale (1995)	3.5
Father of the Bride Part II (1995)	4.0

Adventure	Animation	Children	Comedy	Fantasy	Romance	...	Horror	Mystery	Sci-Fi	IMAX	Documentary	War	Musical	Western	Film-Noir	(no genres listed)
1.0	1.0	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Week 12 :Content-based RS Item-Item recommendation systems



	2
	10
	8

Input User Ratings

X

	Comedy	Adventure	Super Hero	Sci-Fi
	0	1	1	0
	1	1	1	1
	1	0	1	0

Movies Matrix

```
inputMovies['rating']
```

```
0    3.5
1    2.0
2    5.0
3    4.5
4    5.0
```

```
#Resetting the index to avoid future issues
userMovies = userMovies.reset_index(drop=True)
#Dropping unnecessary issues due to save memory and to avoid issues
userGenreTable = userMovies.drop('movieId', 1).drop('title', 1).drop('genres', 1).drop('year', 1)
userGenreTable
```

	Adventure	Animation	Children	Comedy	Fantasy	Romance	Drama	Action	Crime	Thriller	Horror	Mystery	Sci-Fi	IMAX	Documentary	War	Musical	Western	Film-Noir	(no genres listed)
0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	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	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Week 12 :Content-based RS Multiply

University of Salford MANCHESTER

	2
	10
	8

Input User Ratings

X

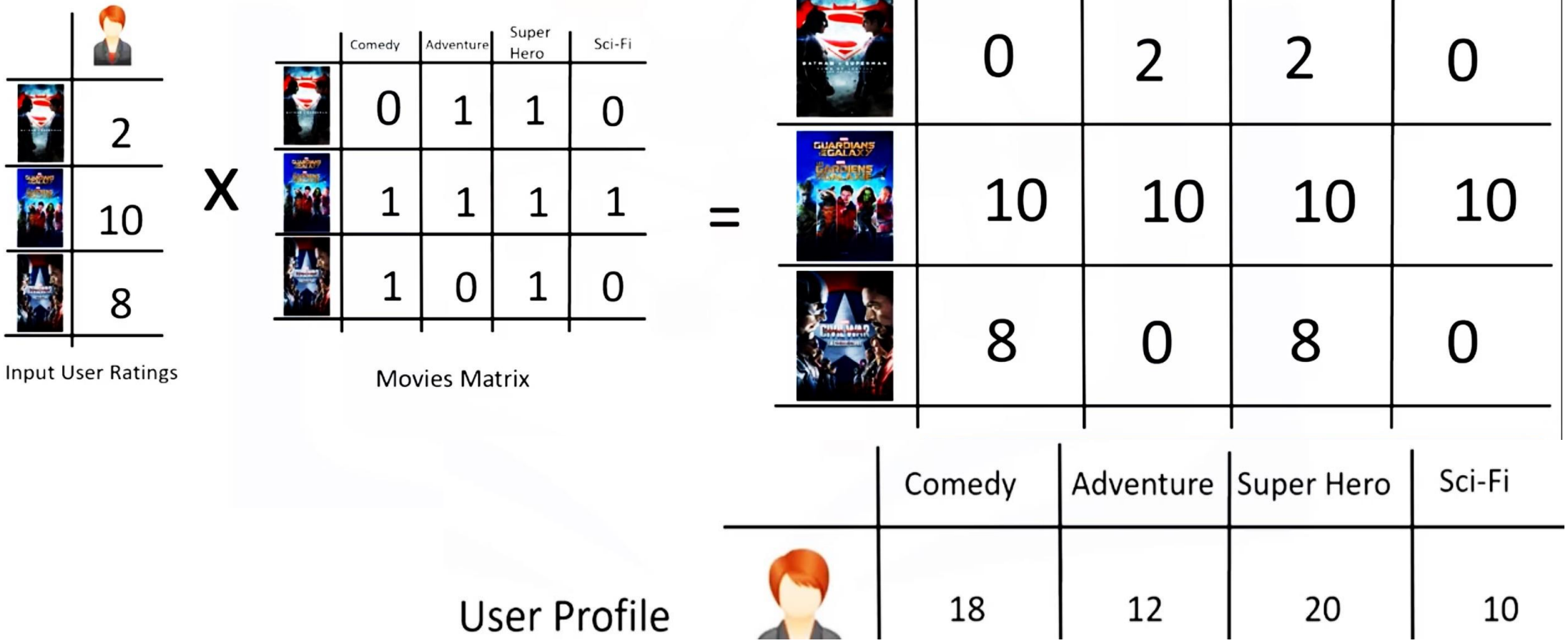
	Comedy	Adventure	Super Hero	Sci-Fi
	0	1	1	0
	1	1	1	1
	1	0	1	0

Movies Matrix

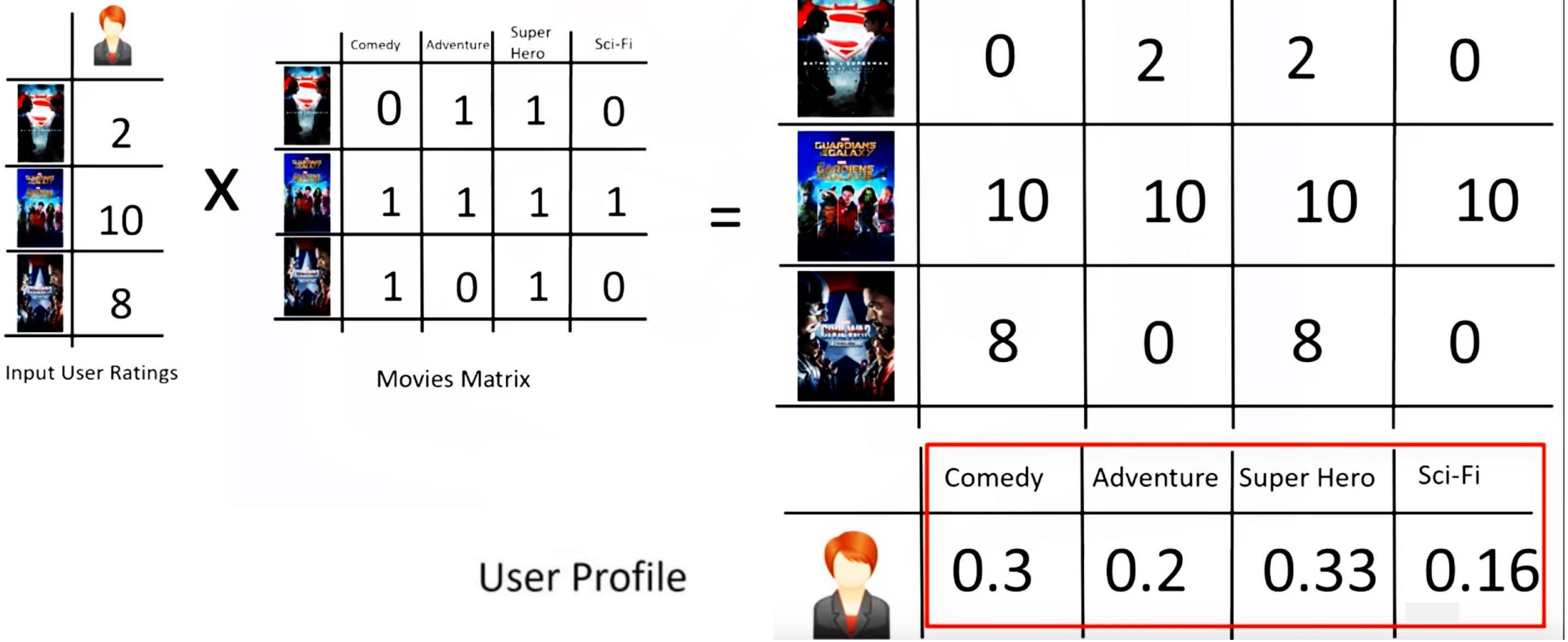
=

	Comedy	Adventure	Super Hero	Sci-Fi
	0	2	2	0
	10	10	10	10
	8	0	8	0

Week 12 :Content-based RS
Multiply – aggregate or sum



Week 12 :Content-based RS
Multiply – aggregate – Normalize



Week 12 :Content-based RS
Multiply – aggregate – Normalize

University of Salford MANCHESTER

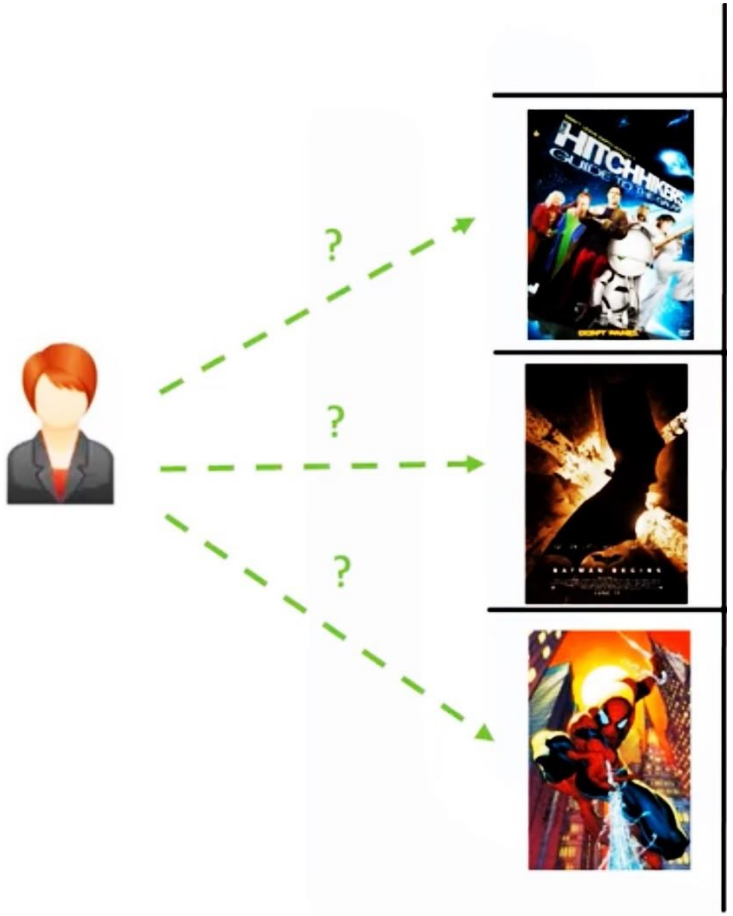
```
#Dot produt to get weights
userProfile = userGenreTable.transpose().dot(inputMovies['rating'])
#The user profile
userProfile
```

```
Adventure            10.0
Animation             8.0
Children              5.5
Comedy               13.5
Fantasy               5.5
Romance               0.0
Drama                10.0
Action                4.5
Crime                 5.0
Thriller              5.0
Horror                0.0
Mystery               0.0
Sci-Fi                4.5
IMAX                  0.0
Documentary           0.0
War                   0.0
Musical               0.0
Western               0.0
Film-Noir             0.0
(no genres listed)    0.0
```

User Profile

Comedy	Adventure	Super Hero	Sci-Fi
0.3	0.2	0.33	0.16

Week 12 :Content-based RS Movies for recomm. / Predict Rating



Week 12 :Content-based RS Movies for recomm. / Predict Rating

?

?

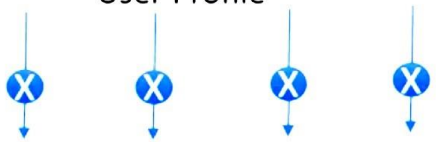
?

	Comedy	Adventure	Super Hero	Sci-Fi
	1	1	0	1
	0	0	1	0
	1	0	1	0

Week 12 :Content-based RS
Weighting the Genre

	Comedy	Adventure	Super Hero	Sci-Fi
	0.3	0.2	0.33	0.16

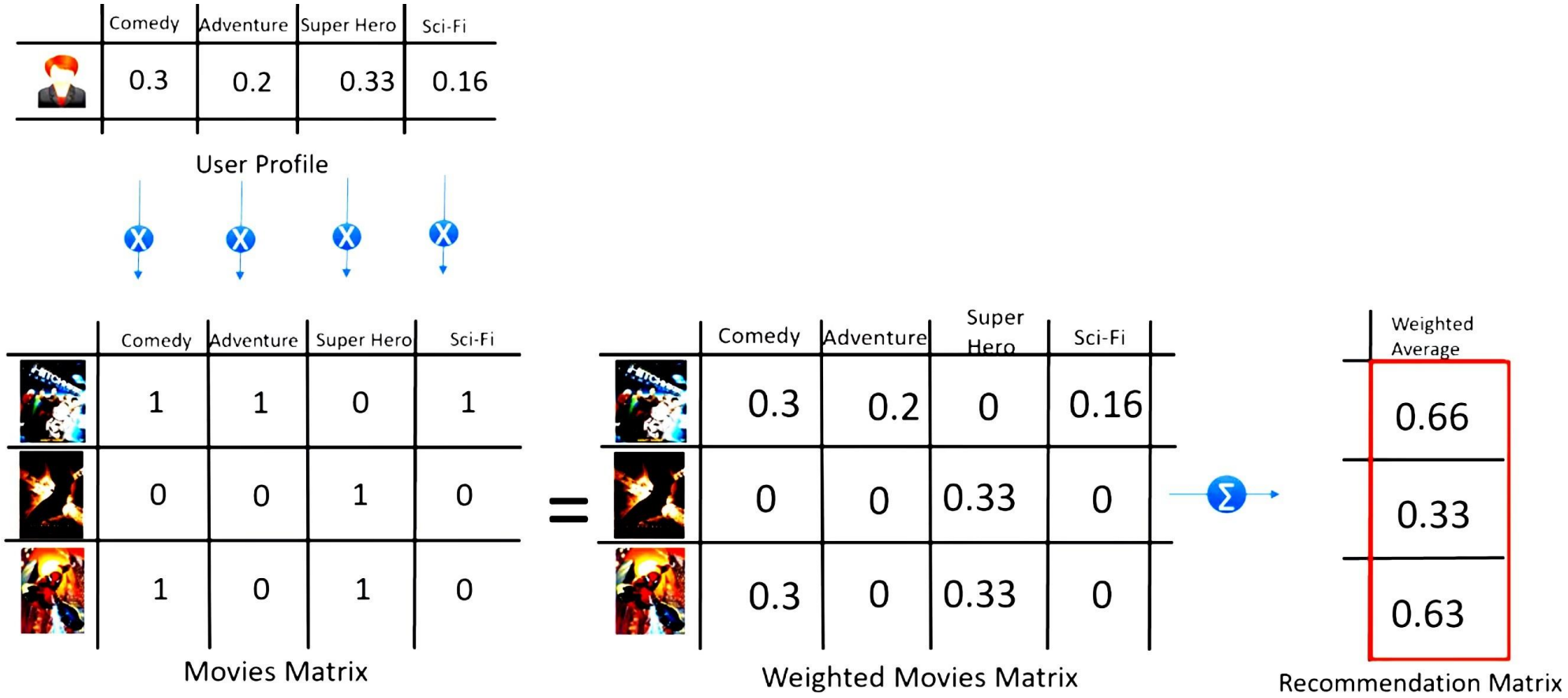
User Profile



	Comedy	Adventure	Super Hero	Sci-Fi
	1	1	0	1
	0	0	1	0
	1	0	1	0

Movies Matrix

Week 12 :Content-based RS
Weighting the Genre



Week 12 :Content-based RS
Weighting the Genre

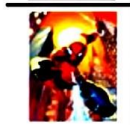
	Comedy	Adventure	Super Hero	Sci-Fi
	0.3	0.2	0.33	0.16

User Profile

	Comedy	Adventure	Super Hero	Sci-Fi
	1	1	0	1
	0	0	1	0
	1	0	1	0

Movies Matrix

=

	Comedy	Adventure	Super Hero	Sci-Fi
	0.3	0.2	0	0.16
	0	0	0.33	0
	0.3	0	0.33	0

Weighted Movies Matrix

Σ

Weighted Average
0.66
0.33
0.63

Recommendation Matrix

Week 12 :Content-based RS
UP/Weighting the Genre

Week 12 :Content-based RS User Profile

University of Salford MANCHESTER

```
#Multiply the genres by the weights and then take the weighted average
recommendationTable_df = ((genreTable*userProfile).sum(axis=1))/(userProfile.sum())
recommendationTable_df.head()
```

```
movieId
1    0.594406
2    0.293706
3    0.188811
4    0.328671
5    0.188811
```

Week 12 :Content-based RS User Profile

University of Salford MANCHESTER

```
#Sort our recommendations in descending order
recommendationTable_df = recommendationTable_df.sort_values(ascending=False)
#Just a peek at the values
recommendationTable_df.head()
```

```
movieId
5018      0.748252
26093     0.734266
27344     0.720280
148775    0.685315
6902      0.678322
```

Week 12 :Content-based RS User Profile - predict

University of Salford MANCHESTER

```
#The final recommendation table
movies_df.loc[movies_df['movieId'].isin(recommendationTable_df.head(5).keys())]
```

	movieId	title	genres	year
4923	5018	Motorama	[Adventure, Comedy, Crime, Drama, Fantasy, Mys...	1991
6793	6902	Interstate 60	[Adventure, Comedy, Drama, Fantasy, Mystery, S...	2002
8605	26093	Wonderful World of the Brothers Grimm, The	[Adventure, Animation, Children, Comedy, Drama...	1962
9296	27344	Revolutionary Girl Utena: Adolescence of Utena...	[Action, Adventure, Animation, Comedy, Drama, ...	1999
33509	148775	Wizards of Waverly Place: The Movie	[Adventure, Children, Comedy, Drama, Fantasy, ...	2009

Week 12 :Content-based RS predict

Week 12 :Content-based RS predict – limitation

Week 12 :Content-based RS Advantages & Disadvantages

University of Salford MANCHESTER

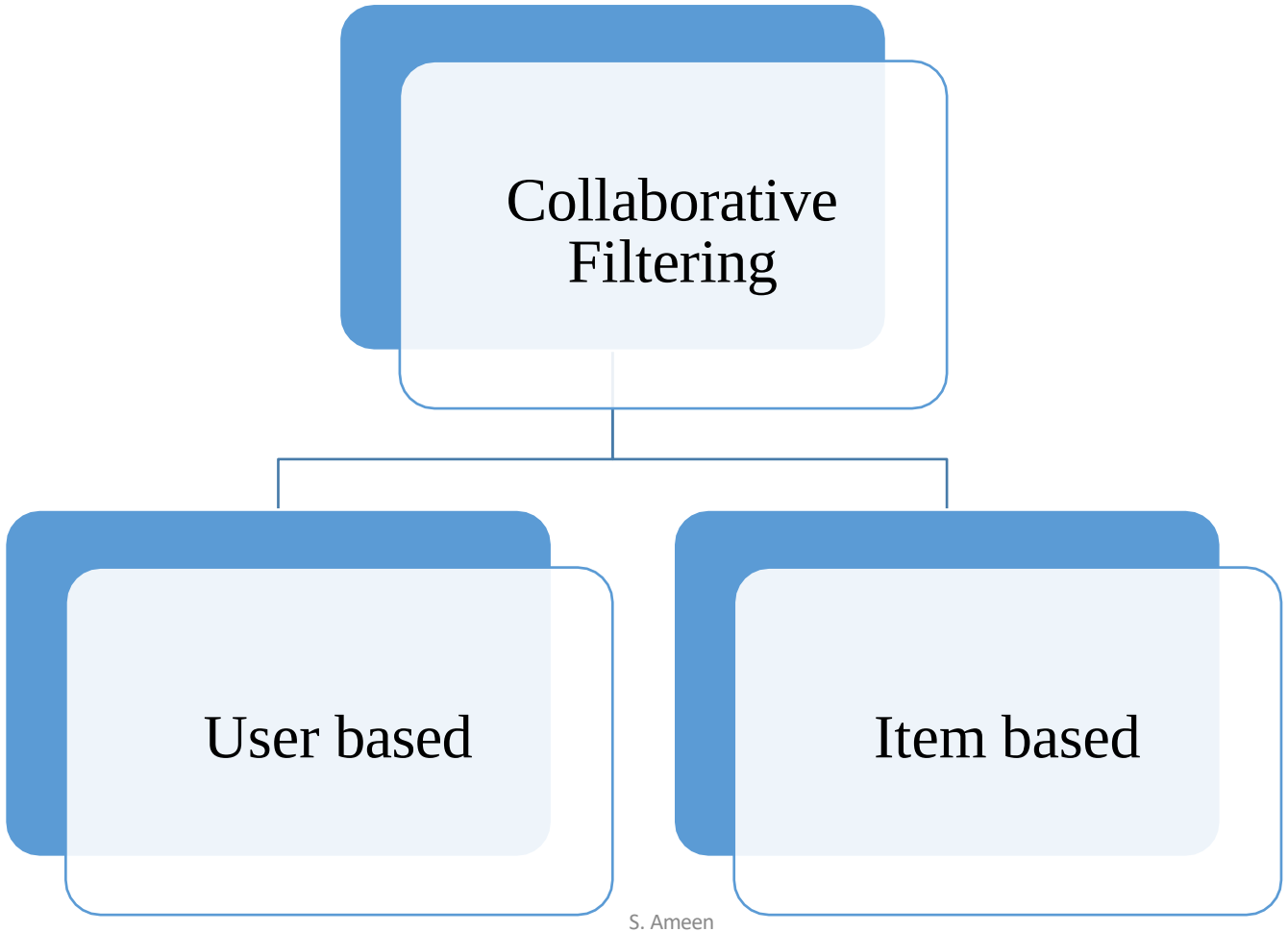
Advantages

- The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

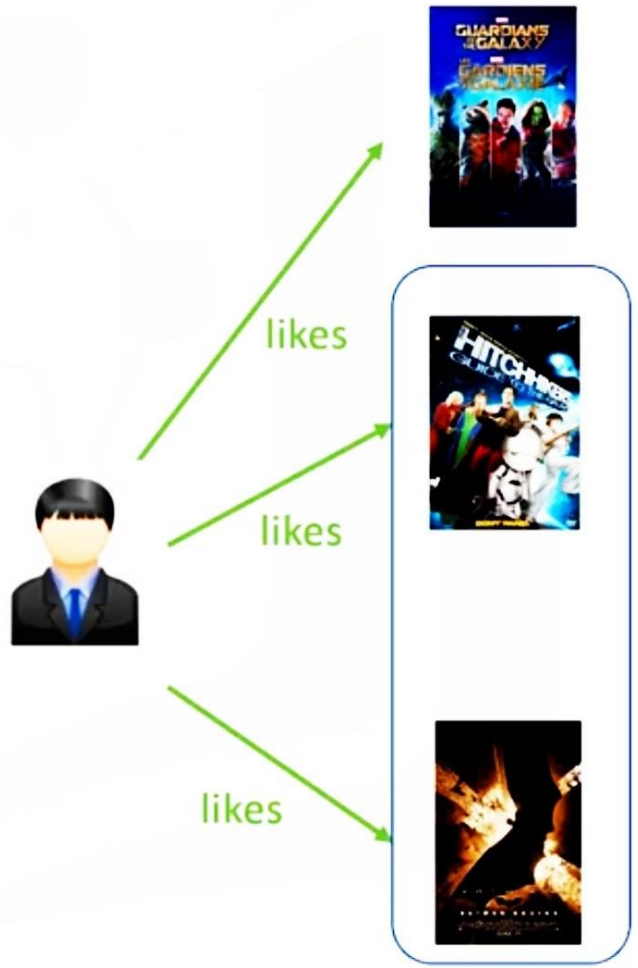
Disadvantages

- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

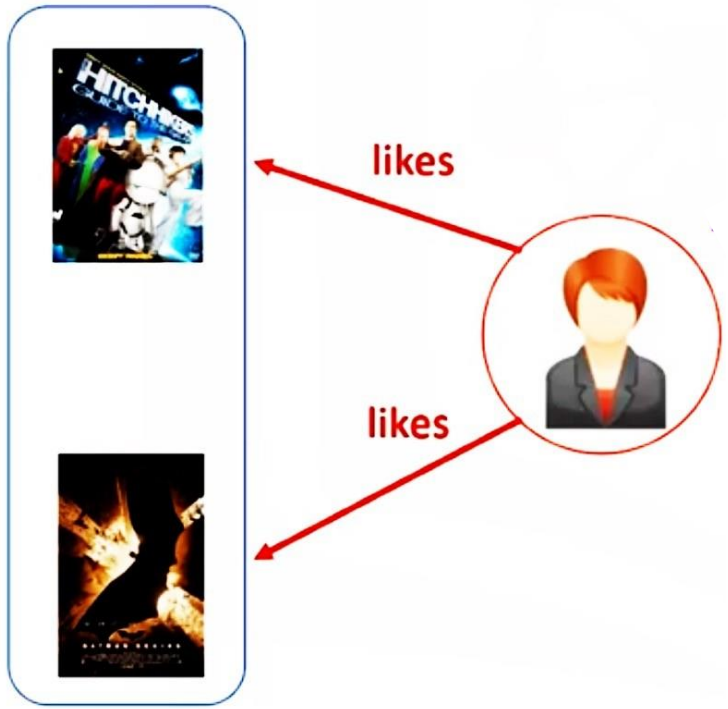
Week 12 :Collaborative Filtering RS - Low rank matrix factorization

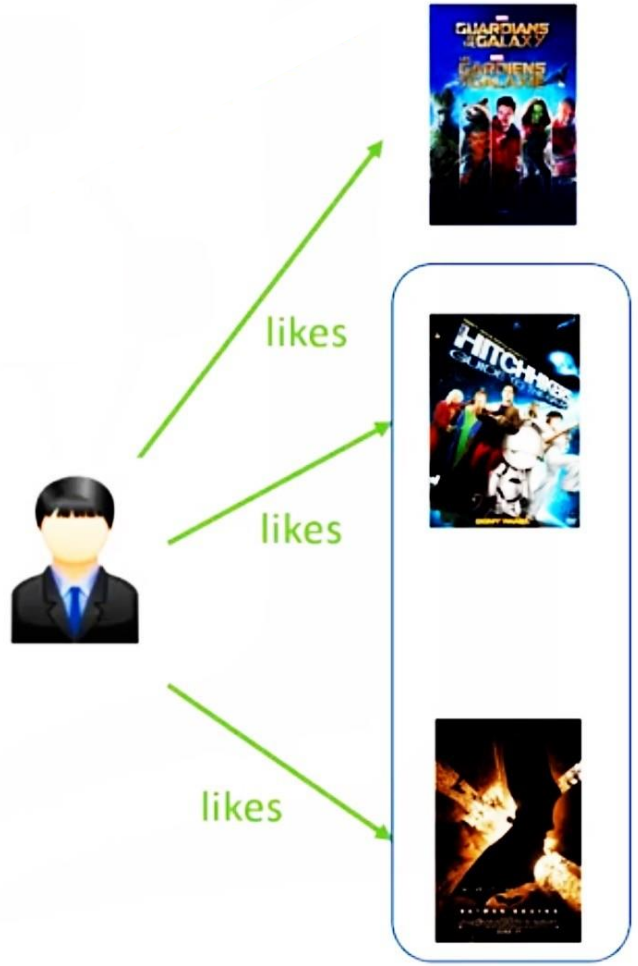


Week 12 :Collaborative Filtering RS - User based

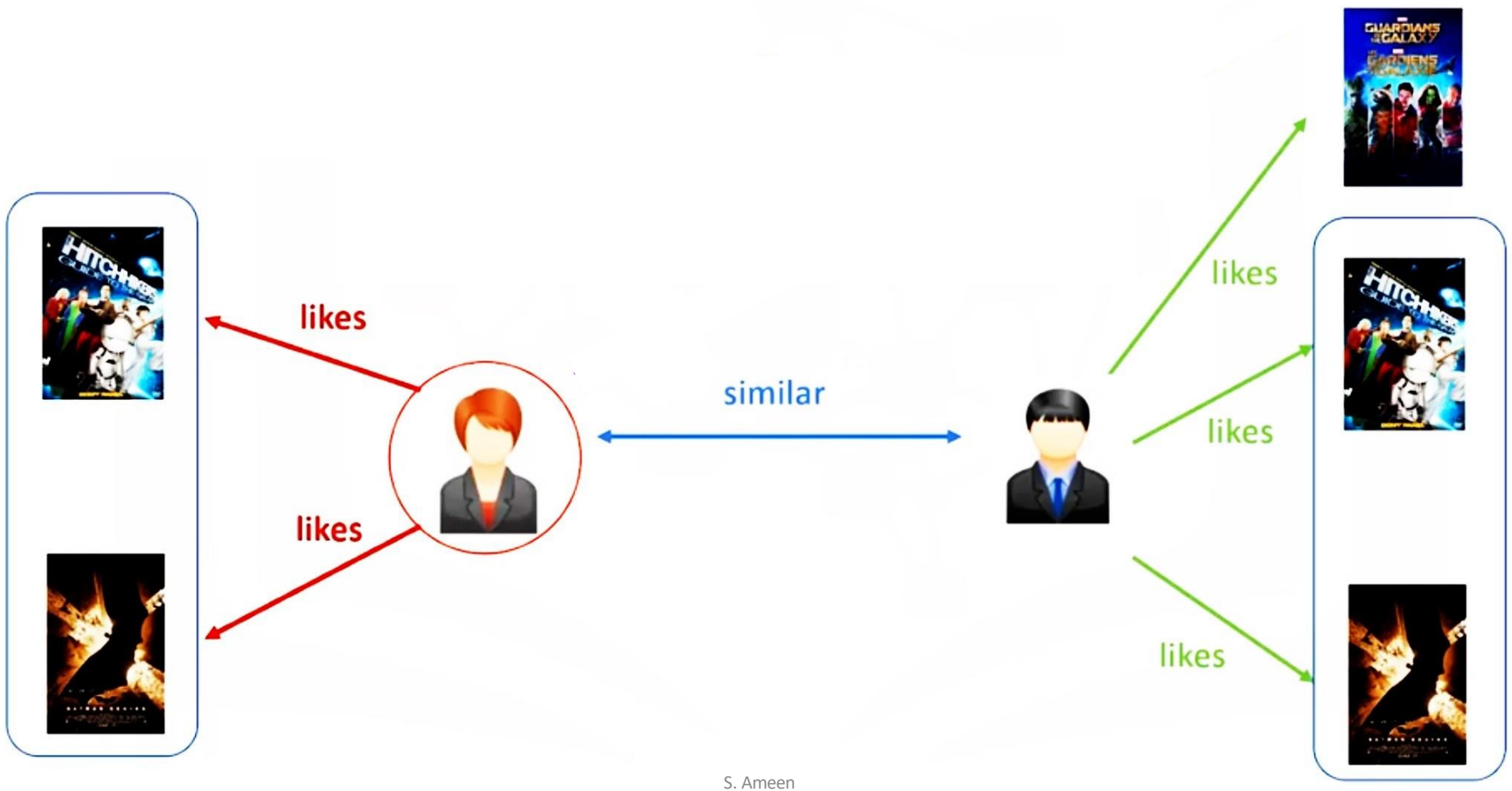


Week 12 :Collaborative Filtering RS - User based

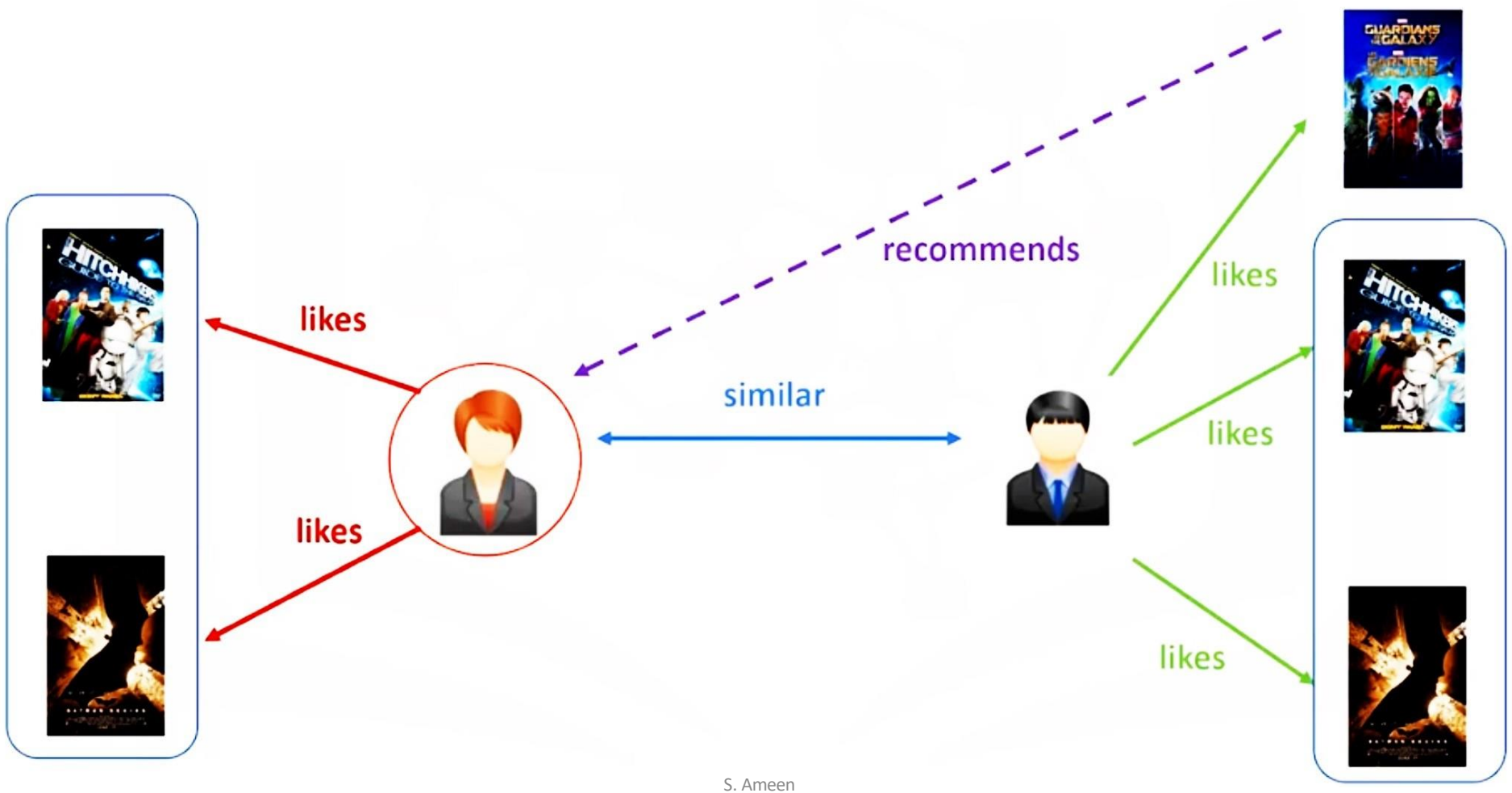




Week 12 :Collaborative Filtering RS - User based



Week 12 :Collaborative Filtering RS - User based



Week 12 :CF RS - / Algorithm /

User based

Active user			10	7	8

Ratings Matrix

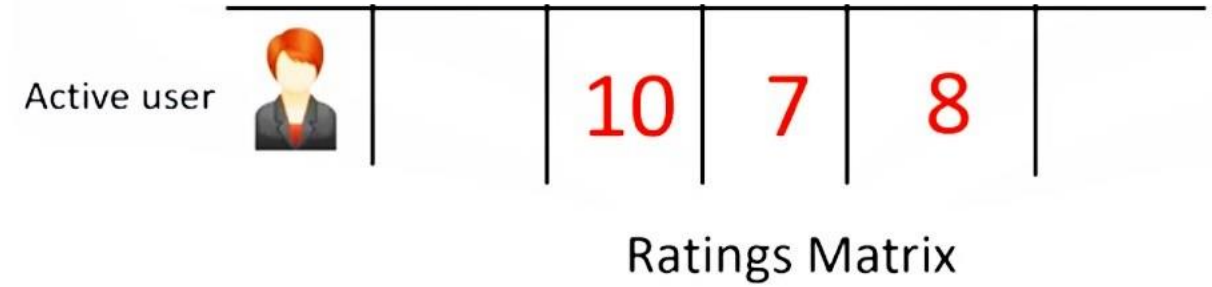
Week 12 :CF RS - / Algorithm /

User based

University of Salford MANCHESTER

```
userInput = [
            {'title':'Breakfast Club, The', 'rating':5},
            {'title':'Toy Story', 'rating':3.5},
            {'title':'Jumanji', 'rating':2},
            {'title':"Pulp Fiction", 'rating':5},
            {'title':'Akira', 'rating':4.5}
        ]
inputMovies = pd.DataFrame(userInput)
inputMovies
```

	rating	title
0	5.0	Breakfast Club, The
1	3.5	Toy Story
2	2.0	Jumanji
3	5.0	Pulp Fiction
4	4.5	Akira



Week 12 :CF RS - / Algorithm /

User based

University of Salford MANCHESTER

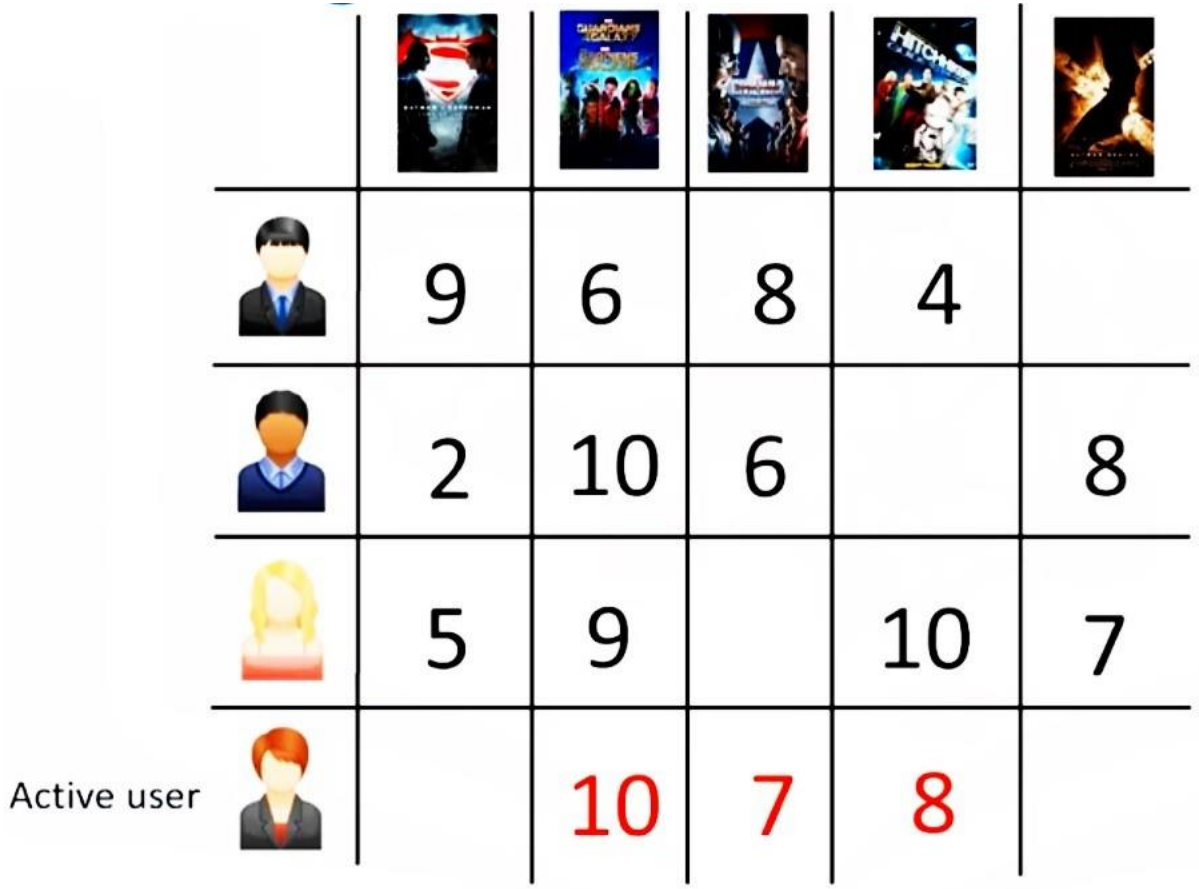
```
#Filtering out the movies by title
inputId = movies_df[movies_df['title'].isin(inputMovies['title'].tolist())]
#Then merging it so we can get the movieId. It's implicitly merging it by title.
inputMovies = pd.merge(inputId, inputMovies)
#Dropping information we won't use from the input dataframe
inputMovies = inputMovies.drop('year', 1)
#Final input dataframe
#If a movie you added in above isn't here, then it might not be in the original
#dataframe or it might spelled differently, please check capitalisation.
inputMovies
```

	movieId	title	rating
0	1	Toy Story	3.5
1	2	Jumanji	2.0
2	296	Pulp Fiction	5.0
3	1274	Akira	4.5
4	1968	Breakfast Club, The	5.0

Ratings Matrix

Week 12 :CF RS - / Algorithm /

User based



Ratings Matrix

Week 12 :CF RS - / Algorithm /

User based

University of Salford MANCHESTER

```
#Filtering out users that have watched movies that the input has watched and storing it
userSubset = ratings_df[ratings_df['movieId'].isin(inputMovies['movieId'].tolist())]
userSubset.head()
```

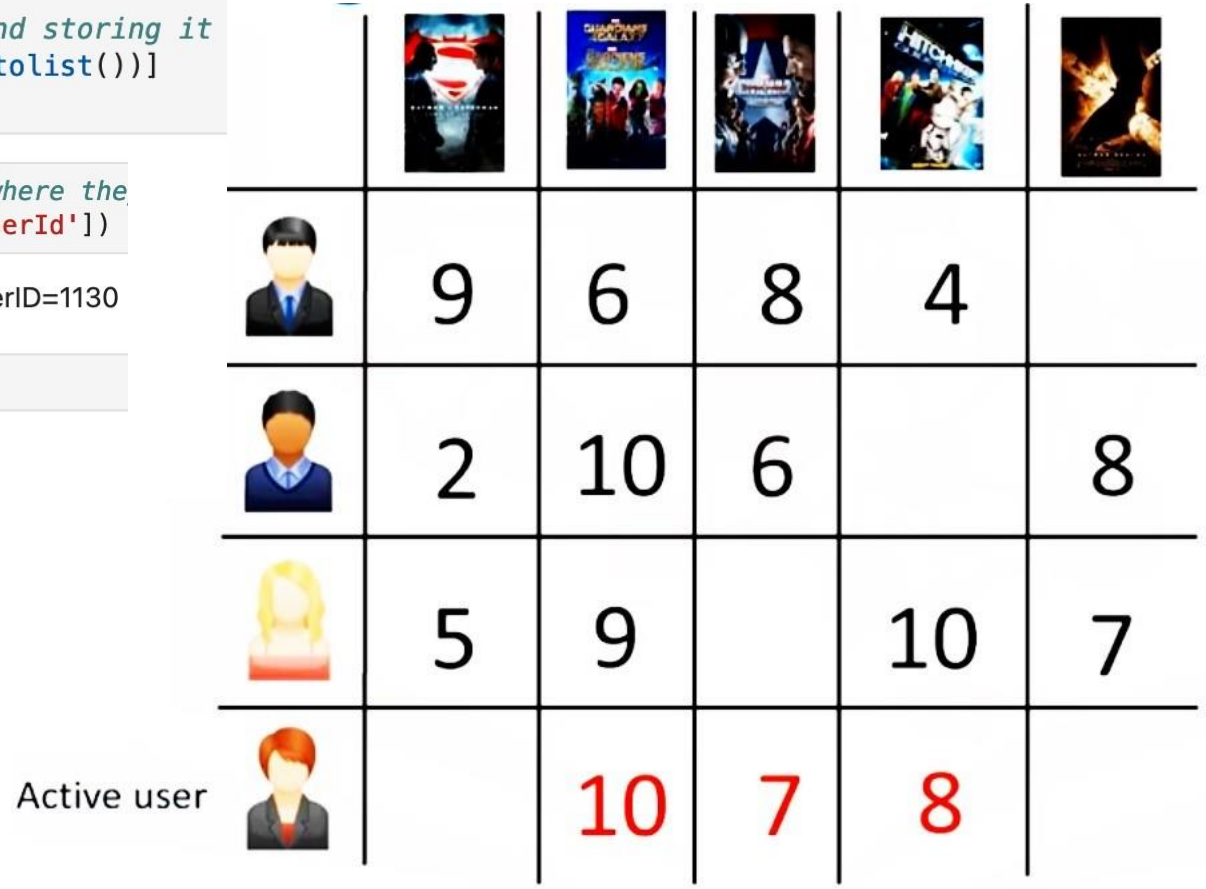
	userId	movieId	rating
19	4	296	4.0
441	12	1968	3.0
479	13	2	2.0
531	13	1274	5.0
681	14	296	2.0

```
#Groupby creates several sub dataframes where the
userSubsetGroup = userSubset.groupby(['userId'])
```

lets look at one of the users, e.g. the one with userID=1130

```
userSubsetGroup.get_group(1130)
```

	userId	movieId	rating
104167	1130	1	0.5
104168	1130	2	4.0
104214	1130	296	4.0
104363	1130	1274	4.5
104443	1130	1968	4.5



Ratings Matrix

Week 12 :CF RS - / Algorithm /

User based

```
#Sorting it so users with movie most in common with the input will have priority
userSubsetGroup = sorted(userSubsetGroup,  key=lambda x: len(x[1]), reverse=True)
```

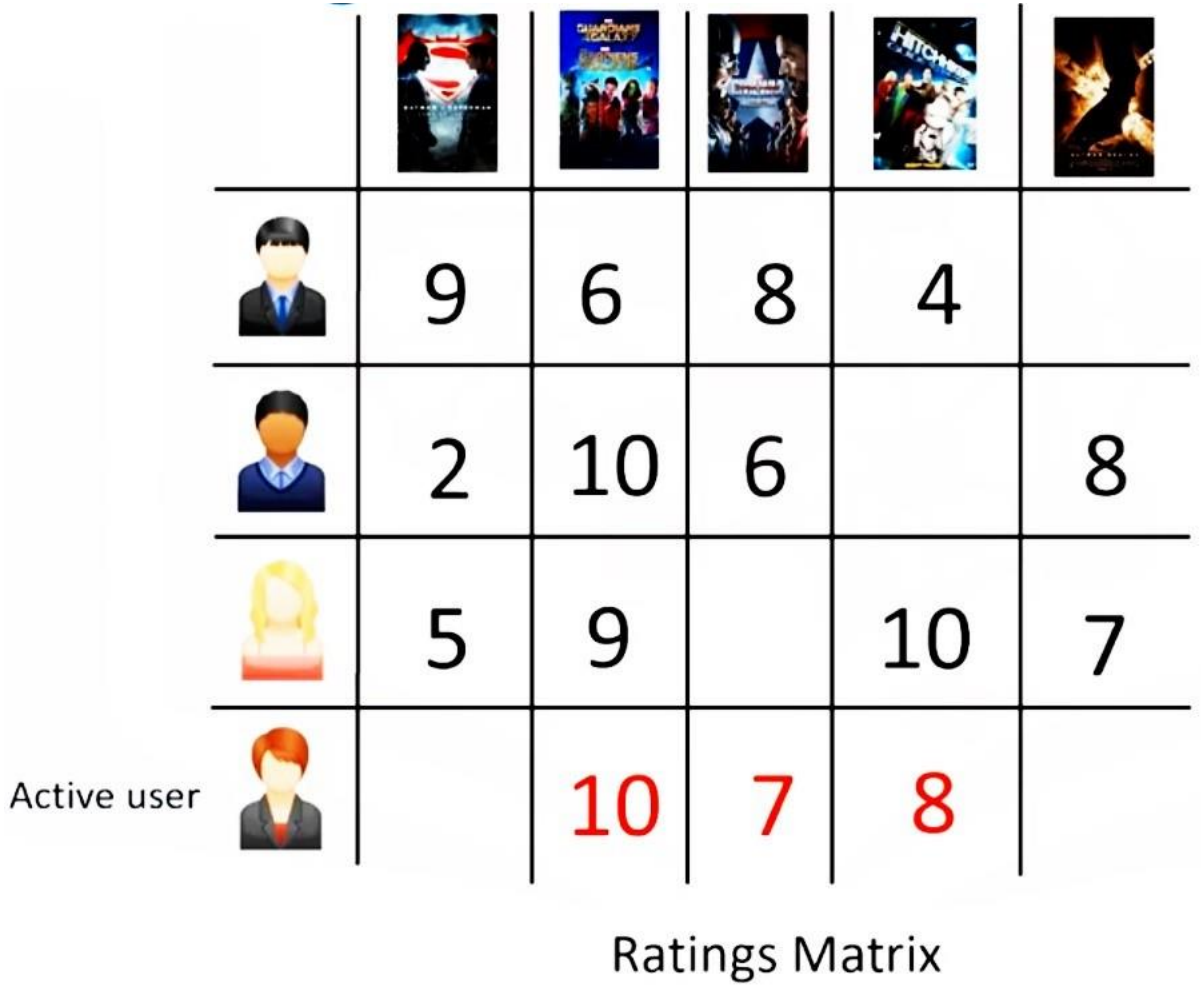
Now lets look at the first user

```
userSubsetGroup[0:3]
```

```
[(75,
        userId  movieId  rating
  7507      75        1     5.0
  7508      75        2     3.5
  7540      75      296     5.0
  7633      75     1274     4.5
  7673      75     1968     5.0),
 (106,
        userId  movieId  rating
  9083     106        1     2.5
  9084     106        2     3.0
  9115     106      296     3.5
  9198     106     1274     3.0
  9238     106     1968     3.5),
 (686,
         userId  movieId  rating
  61336     686        1     4.0
  61337     686        2     3.0
  61377     686      296     4.0
  61478     686     1274     4.0
  61569     686     1968     5.0)]
```

```
userId  movieId  rating
```

```
userId  movieId  rating
```



Week 12 :CF RS - / Algorithm /

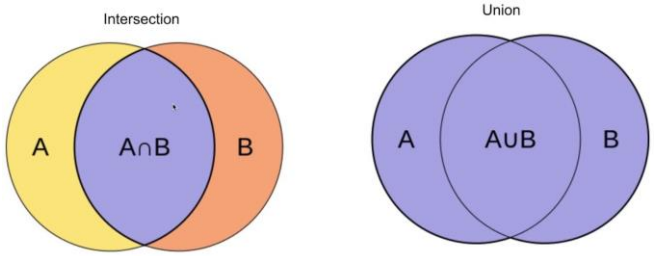
User based

					
	9	6	8	4	
	2	10	6		8
	5	9		10	7
	?	10	7	8	?

Ratings Matrix

Week 12 :CF RS - / Algorithm / Similarity

User based



- Jaccard similarity:
  - normalizes by popularity − Who purchased *A* and *B* *divided by*
  - who purchased *A* or *B*

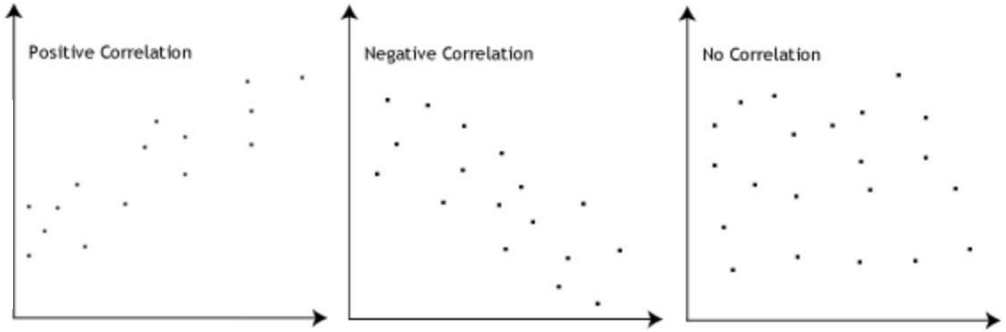
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- Cosine similarity

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

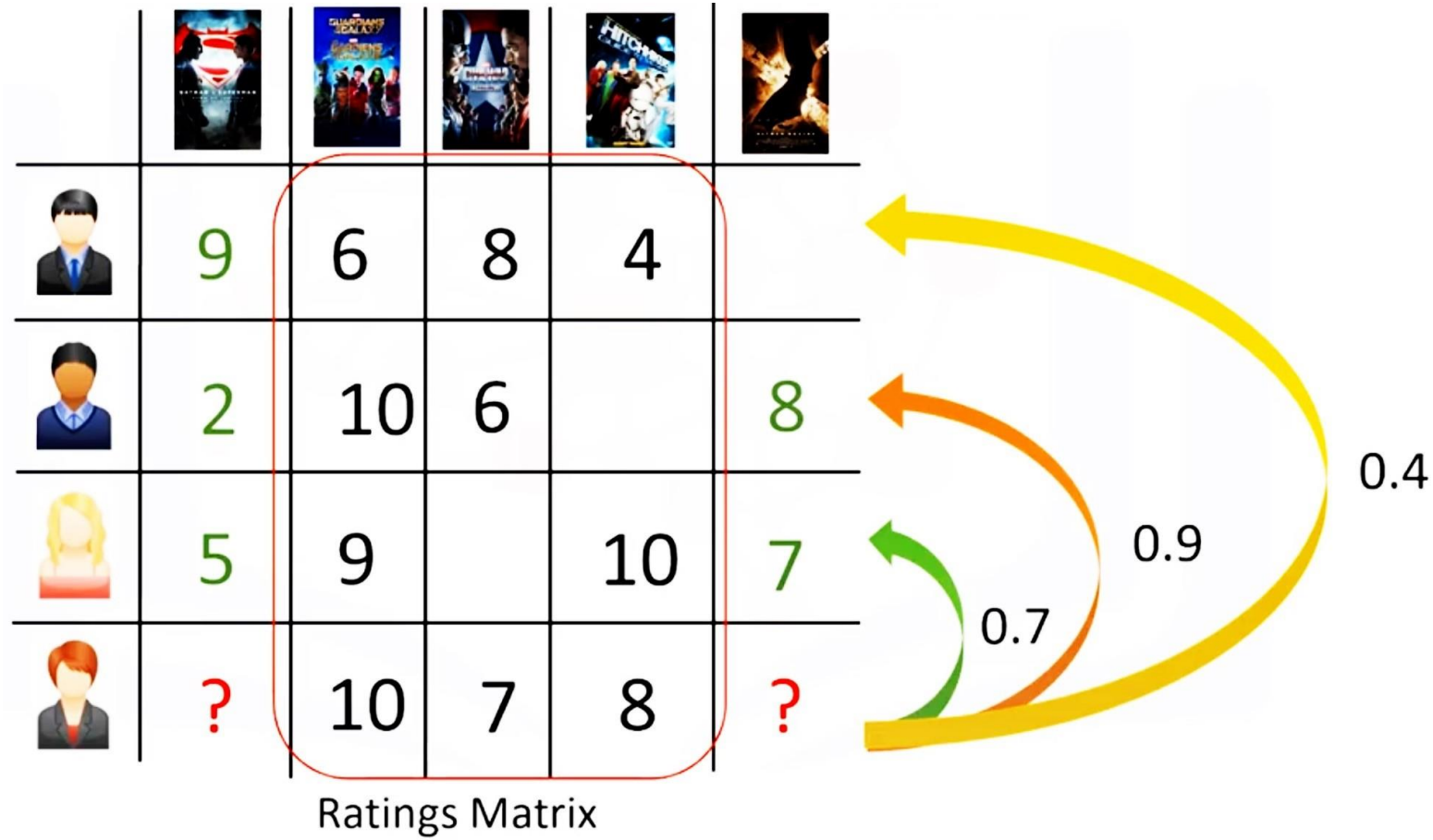
- Pearson correlation

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$



Week 12 :CF RS - / Algorithm / Similarity

User based



Week 12 :CF RS - / Algorithm / Similarity

User based

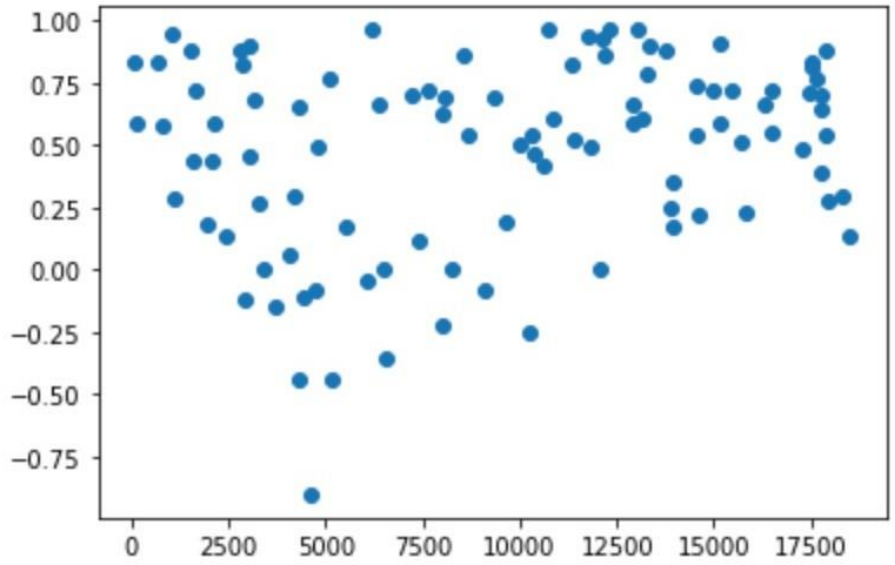
University of Salford MANCHESTER

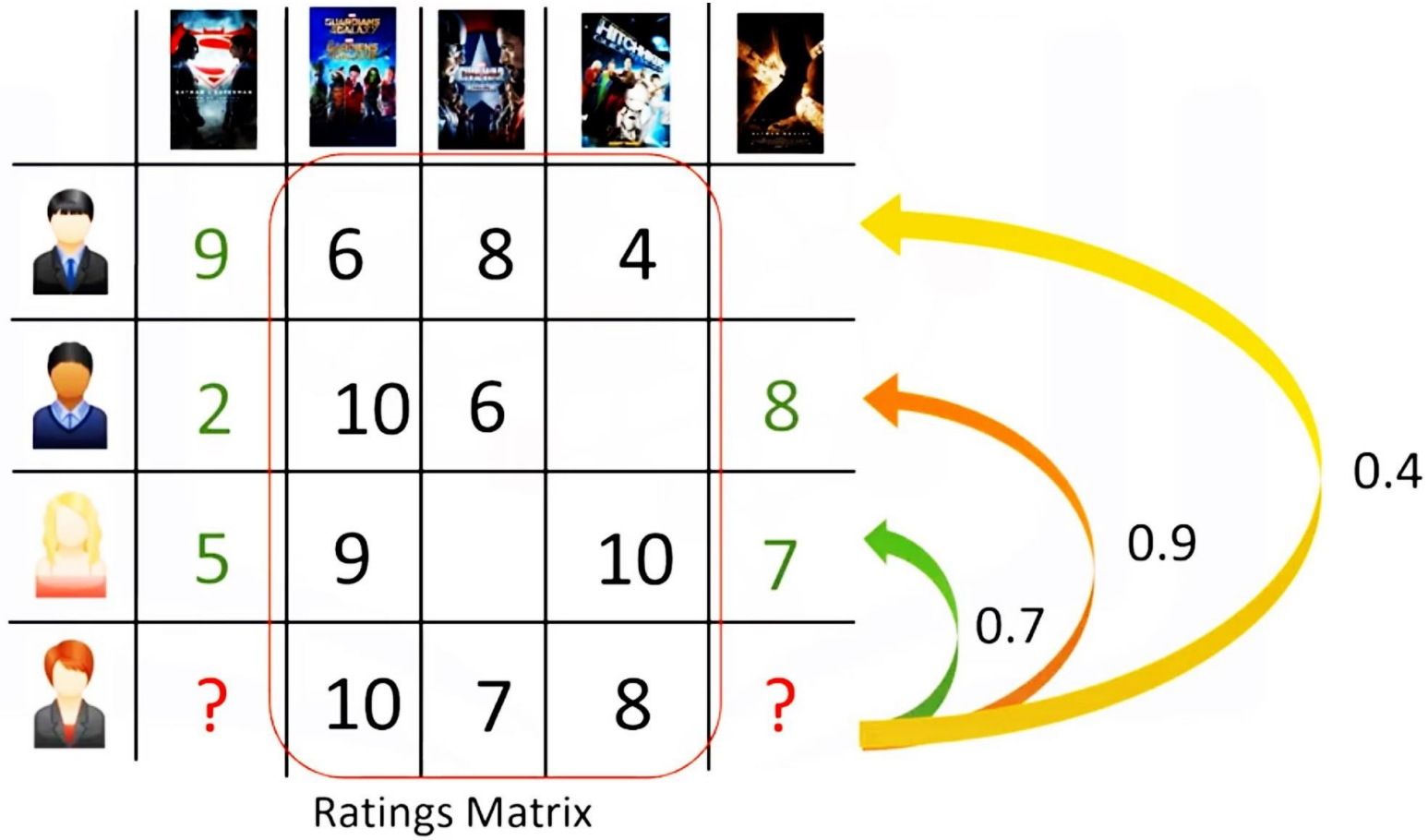
```
#Store the Pearson Correlation in a dictionary, where the key is the user Id and the value is the coefficient
pearsonCorrelationDict = {}

#For every user group in our subset
for name, group in userSubsetGroup:
    #Let's start by sorting the input and current user group so the values aren't mixed up later on
    group = group.sort_values(by='movieId')
    inputMovies = inputMovies.sort_values(by='movieId')
    #Get the N for the formula
    nRatings = len(group)
    #Get the review scores for the movies that they both have in common
    temp_df = inputMovies[inputMovies['movieId'].isin(group['movieId'].tolist())]
    #And then store them in a temporary buffer variable in a list format to facilitate future calculations
    tempRatingList = temp_df['rating'].tolist()
    #Let's also put the current user group reviews in a list format
    tempGroupList = group['rating'].tolist()
    #Now let's calculate the pearson correlation between two users, so called, x and y
    Sxx = sum([i**2 for i in tempRatingList]) - pow(sum(tempRatingList),2)/float(nRatings)
    Syy = sum([i**2 for i in tempGroupList]) - pow(sum(tempGroupList),2)/float(nRatings)
    Sxy = sum( i*j for i, j in zip(tempRatingList, tempGroupList)) - sum(tempRatingList)*sum(tempGroupList)/float(nRatings)

    #If the denominator is different than zero, then divide, else, 0 correlation.
    if Sxx != 0 and Syy != 0:
        pearsonCorrelationDict[name] = Sxy/sqrt(Sxx*Syy)
    else:
        pearsonCorrelationDict[name] = 0
```

Week 12 :CF RS - / Algorithm / Weighting Rate

User based

```
# Plot the correlation
import matplotlib.pylab as plt
myList = pearsonCorrelationDict.items()
myList = sorted(myList)
x, y = zip(*myList)
plt.scatter(x, y)
plt.show()
```





Week 12 :CF RS - / Algorithm / Weighting Rate User based


```
pearsonDF = pd.DataFrame.from_dict(pearsonCorrelationDict, orient='index')
pearsonDF.columns = ['similarityIndex']
pearsonDF['userId'] = pearsonDF.index
pearsonDF.index = range(len(pearsonDF))
pearsonDF.head()
```

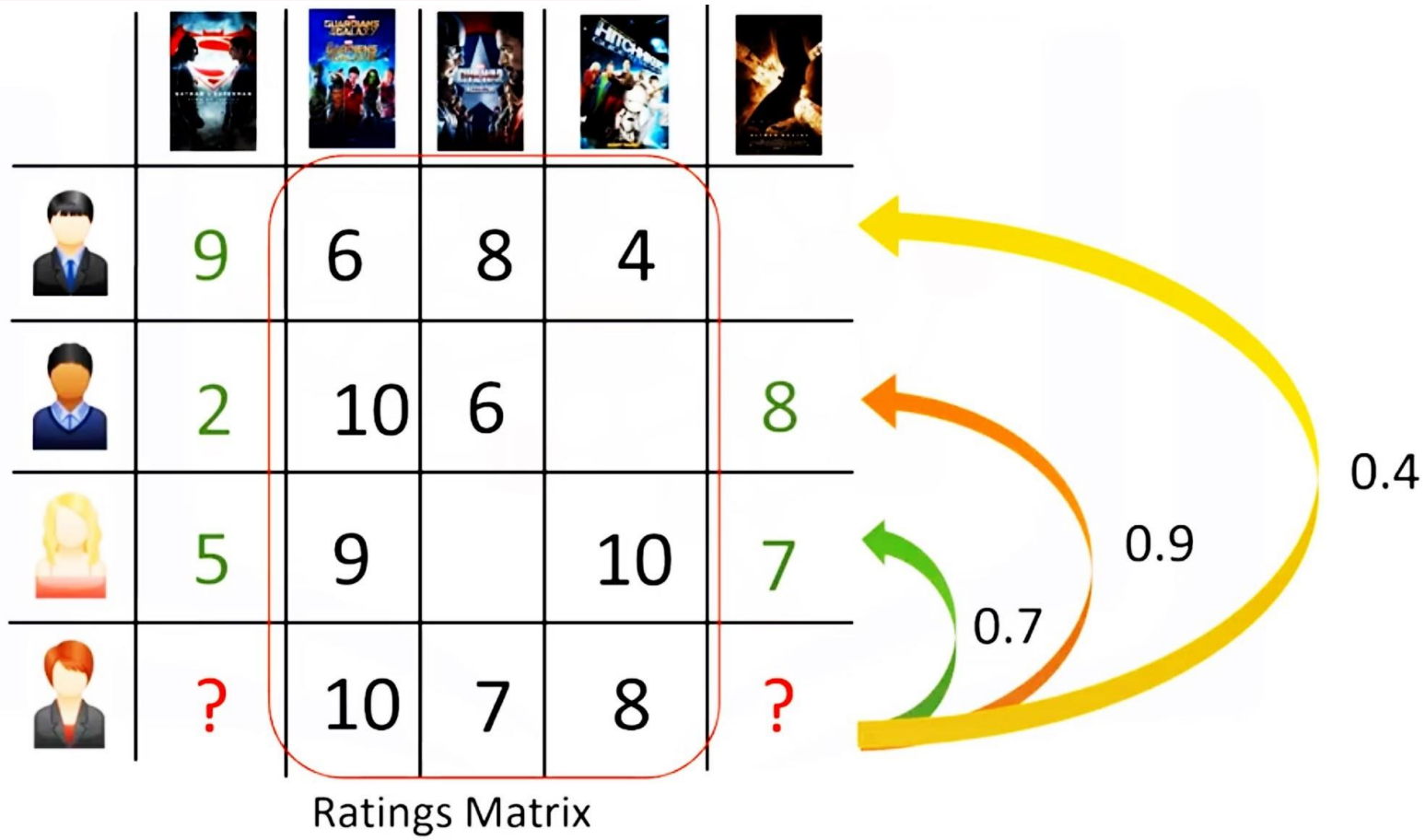
	similarityIndex	userId
0	0.827278	75
1	0.586009	106
2	0.832050	686
3	0.576557	815
4	0.943456	1040

The top x similar users to input user

Now let's get the top 50 users that are most similar to the input.

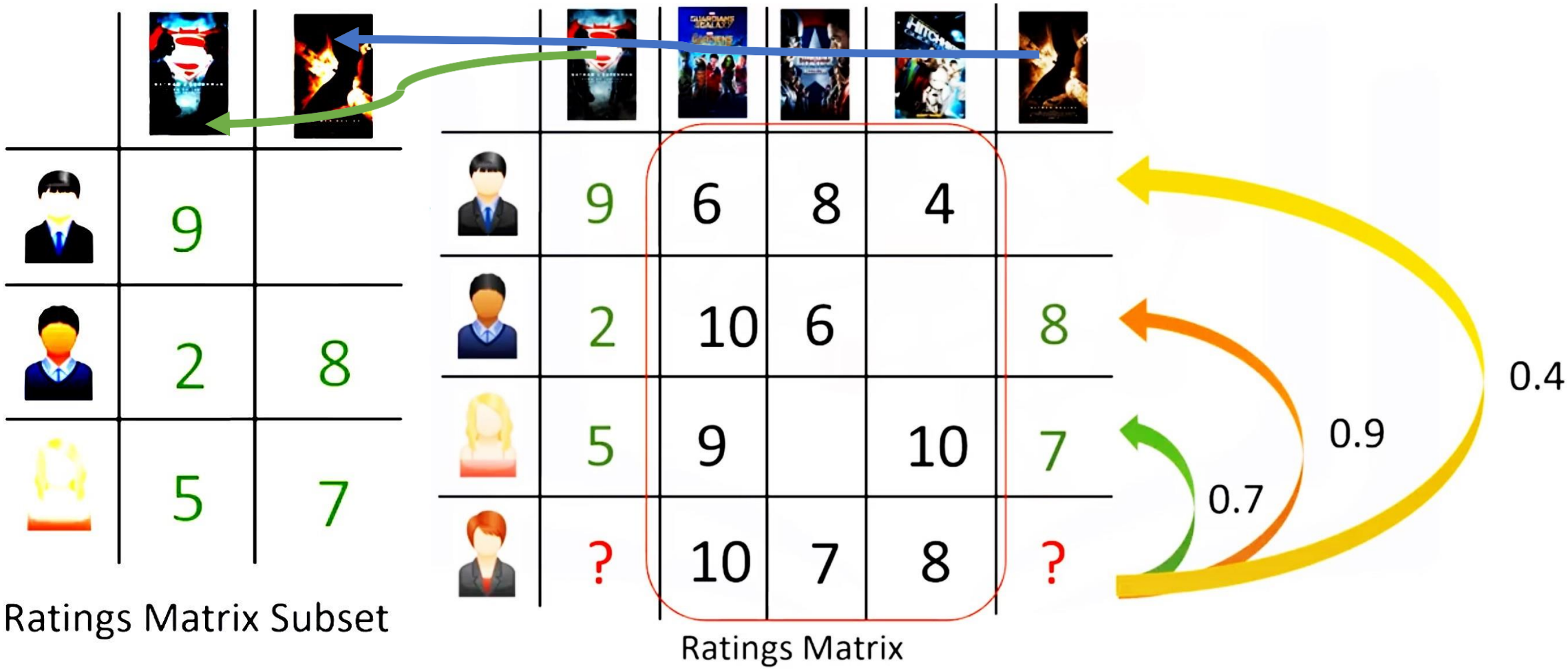
```
topUsers=pearsonDF.sort_values(by='similarityIndex', ascending=False)[0:50]
topUsers.head()
```

	similarityIndex	userId
64	0.961678	12325
34	0.961538	6207
55	0.961538	10707
67	0.960769	13053
4	0.943456	1040



Week 12 :CF RS - / Algorithm / Weighting Rate

User based

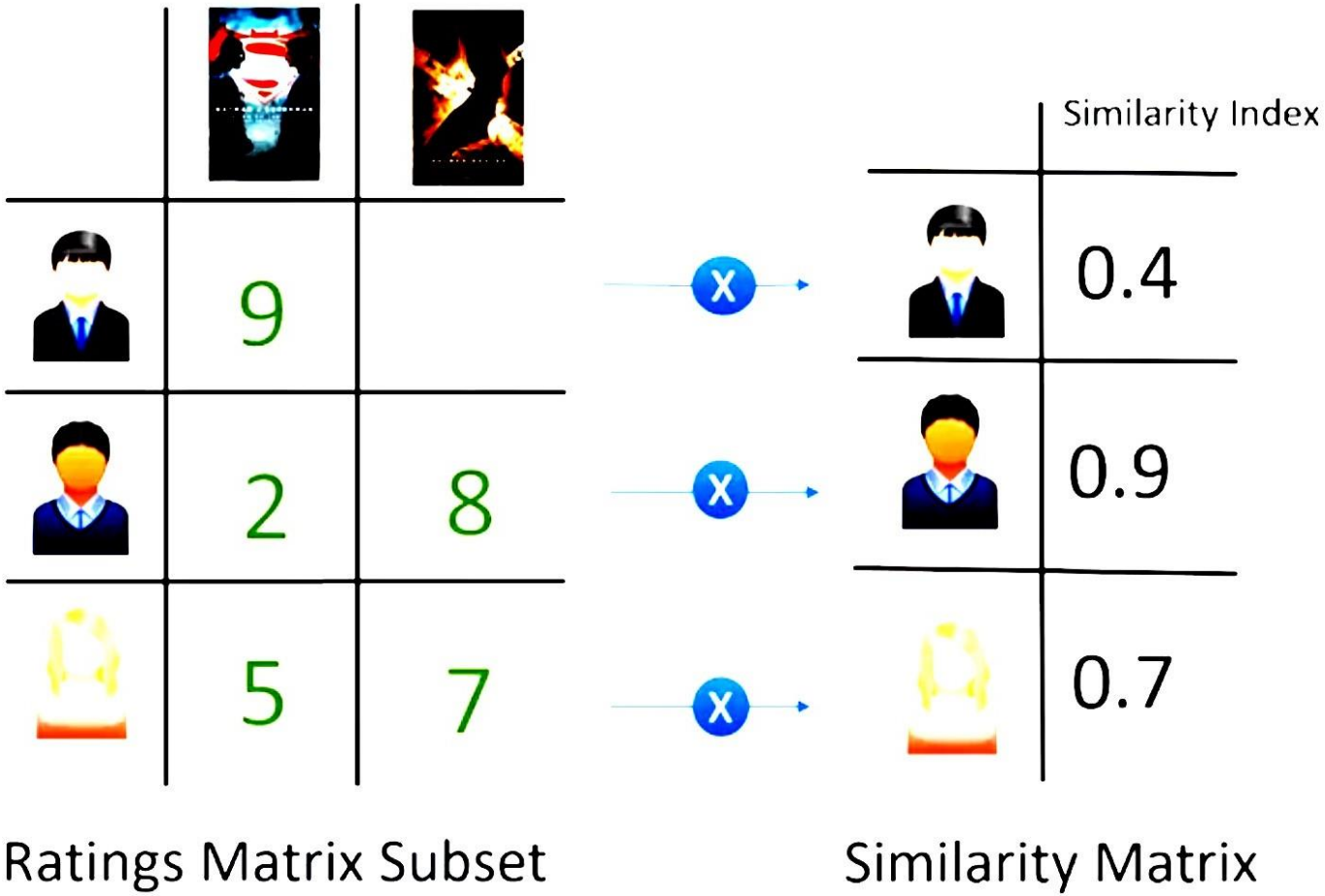


Week 12 :CF RS - User based / Algorithm / Weighting Rate

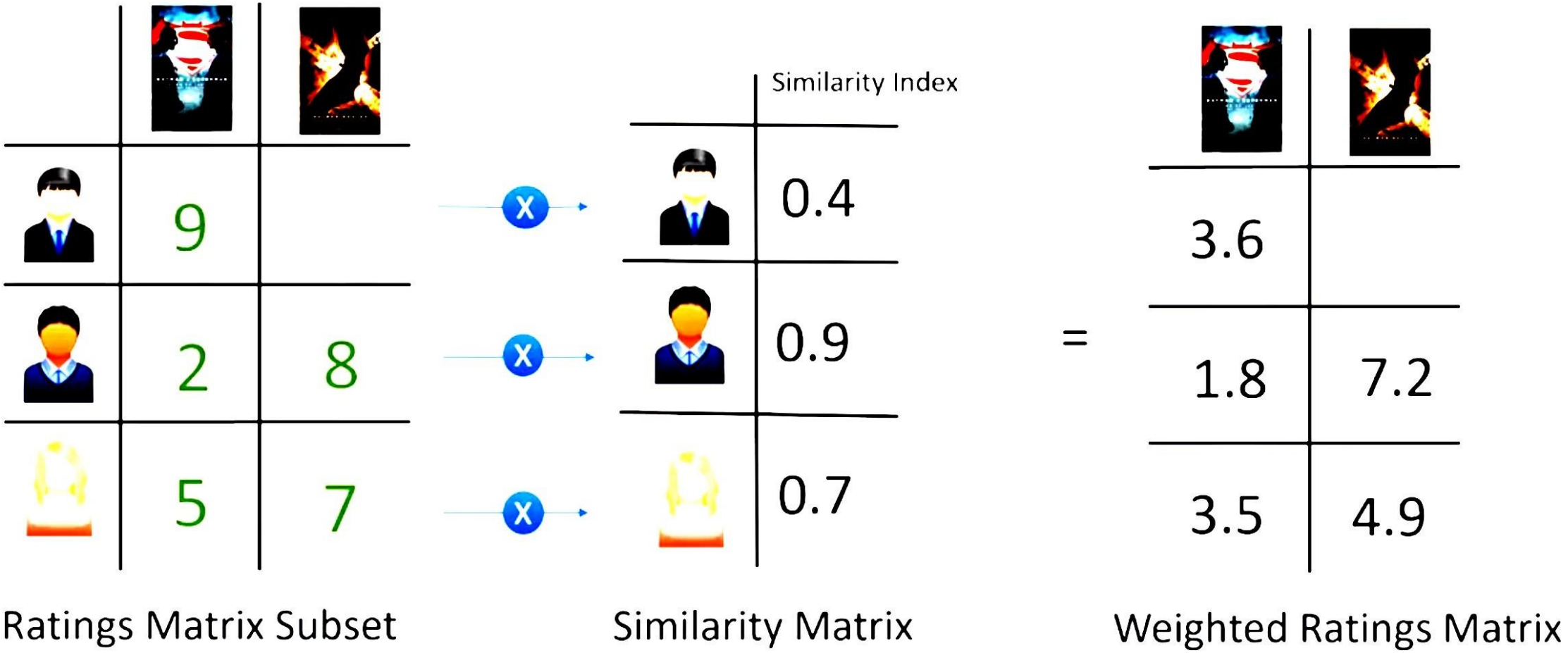
		
	9	
	2	8
	5	7

Ratings Matrix Subset

Week 12 :CF RS - User based / Algorithm / Weighting Rate



Week 12 :CF RS - User based / Algorithm / Weighting Rate



Week 12 :CF RS - User based / Algorithm / Weighting Rate

University of Salford MANCHESTER

```
#Multiplies the similarity by the user's ratings
topUsersRating['weightedRating'] = topUsersRating['similarityIndex']*topUsersRating['rating']
topUsersRating.head()
```

	similarityIndex	userId	movieId	rating	weightedRating
0	0.961678	12325	1	3.5	3.365874
1	0.961678	12325	2	1.5	1.442517
2	0.961678	12325	3	3.0	2.885035
3	0.961678	12325	5	0.5	0.480839
4	0.961678	12325	6	2.5	2.404196

Week 12 :CF RS - User based / Algorithm / Weighting Rate

	Similarity Index
	0.4
	0.9
	0.7

Similarity Matrix

similarityIndex	userId	movieId	rating	weightedRating
0.961678	12325	1	3.5	3.365874
0.961678	12325	2	1.5	1.442517
0.961678	12325	3	3.0	2.885035
0.961678	12325	5	0.5	0.480839
0.961678	12325	6	2.5	2.404196

		
	3.6	
	1.8	7.2
	3.5	4.9

Weighted Ratings Matrix

Week 12 :CF RS - / Algorithm /

User based

	Similarity Index
	0.4
	0.9
	0.7

Similarity Matrix

		
	3.6	
	1.8	7.2
	3.5	4.9

Weighted Ratings Matrix

Week 12 :CF RS - / Algorithm /

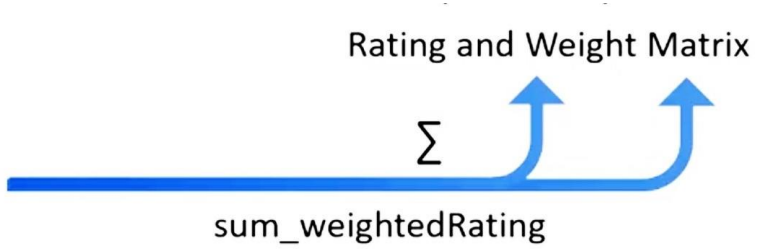
User based

	Similarity Index
	0.4
	0.9
	0.7

Similarity Matrix

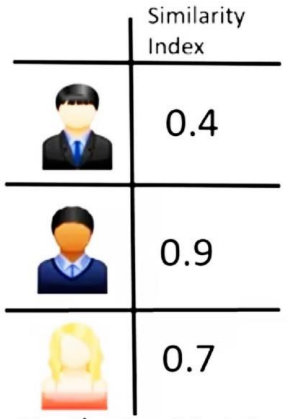
		
	3.6	
	1.8	7.2
	3.5	4.9

Weighted Ratings Matrix

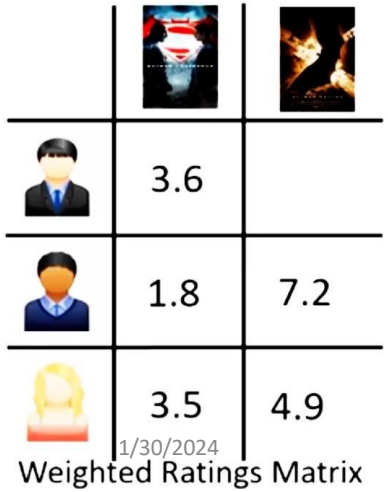


Week 12 :CF RS - / Algorithm /

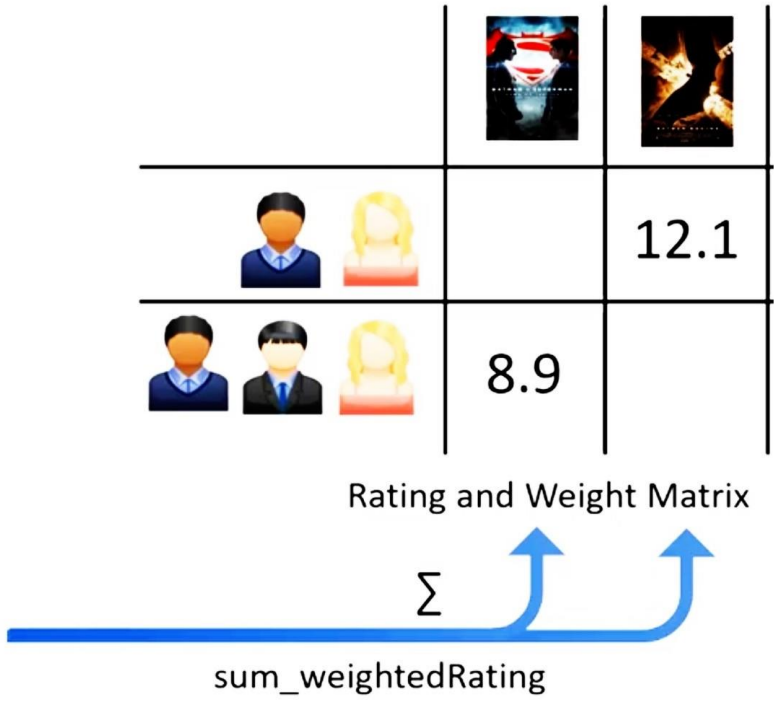
User based



Similarity Matrix



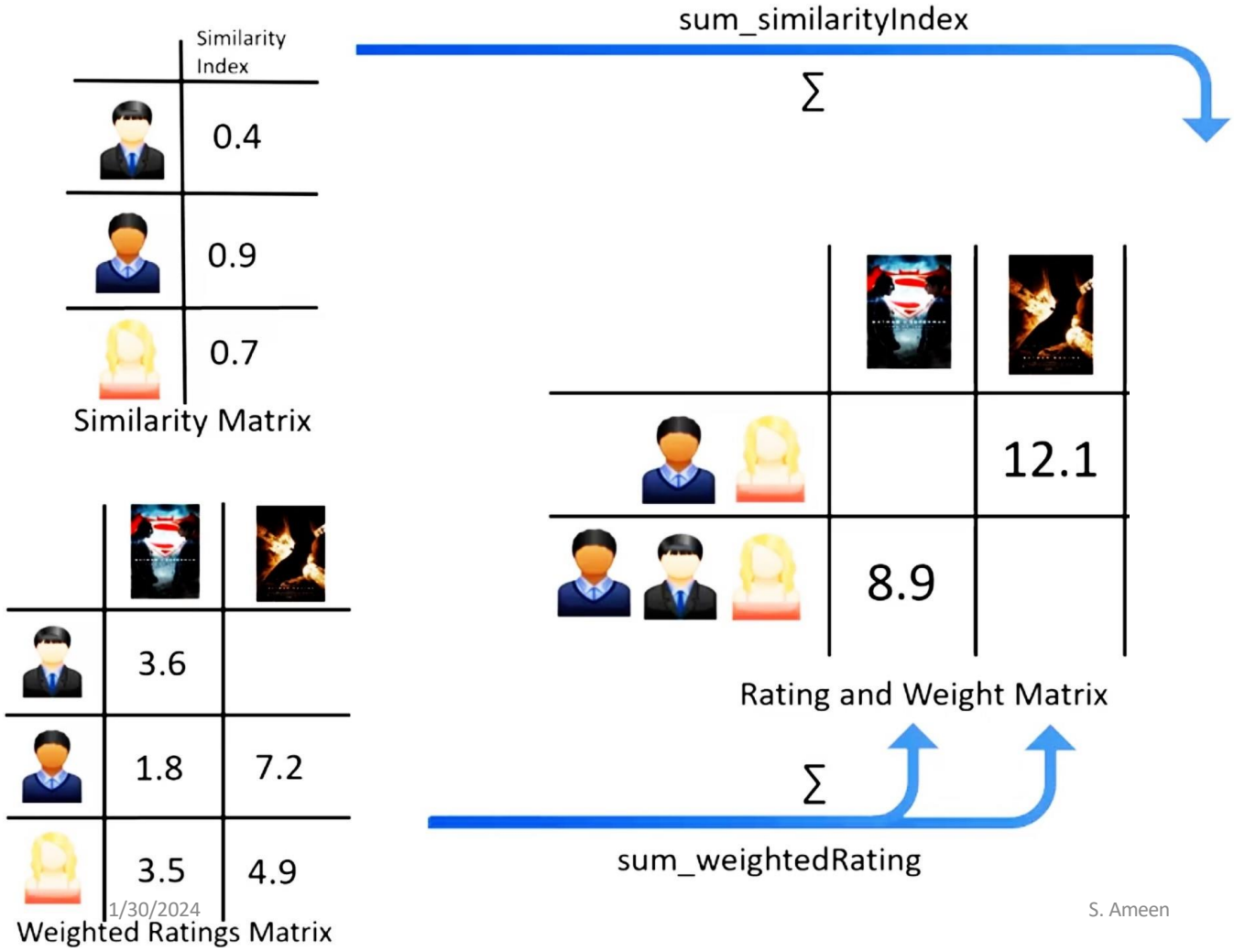
Weighted Ratings Matrix



Rating and Weight Matrix

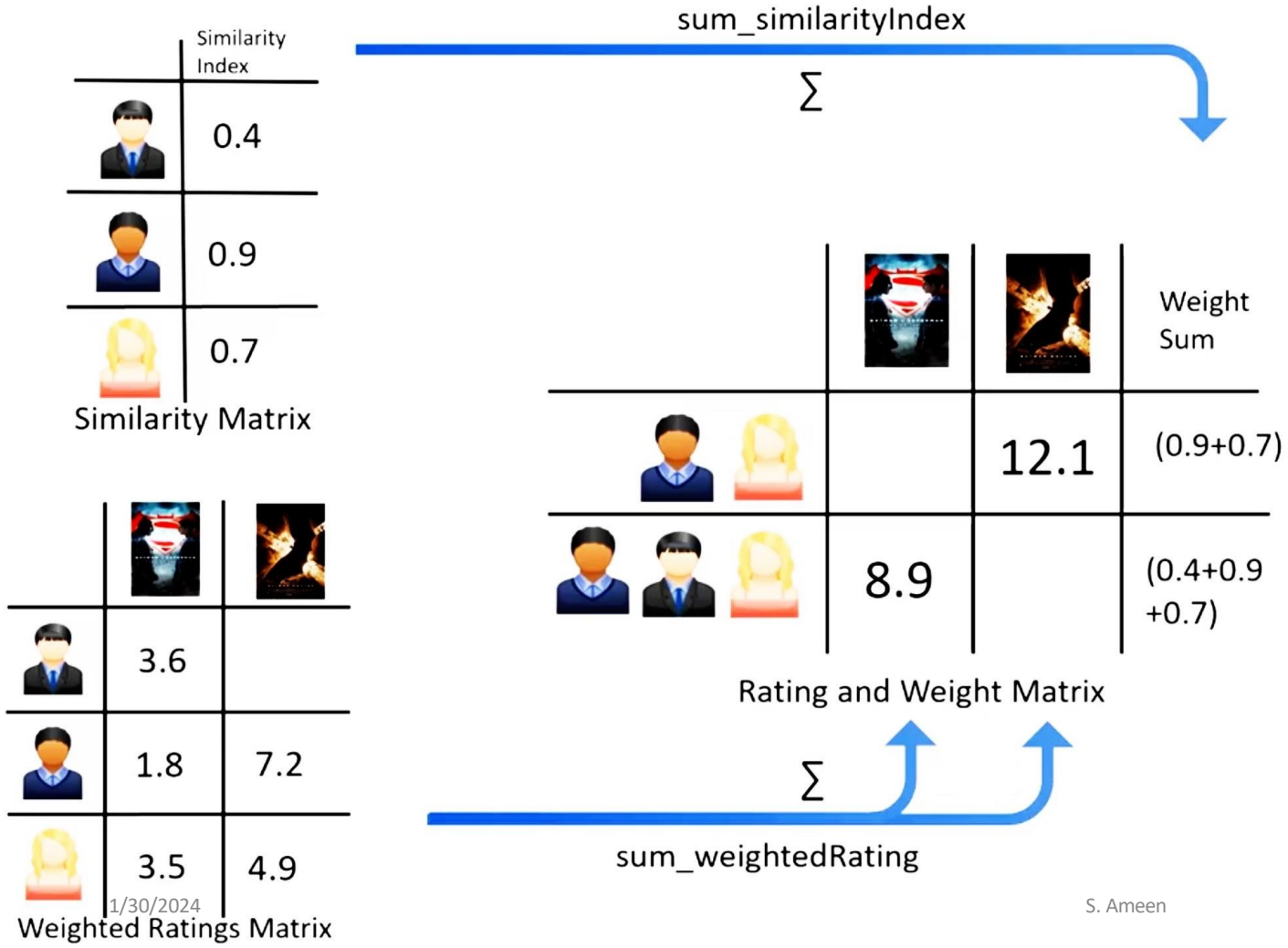
Week 12 :CF RS - / Algorithm /

User based



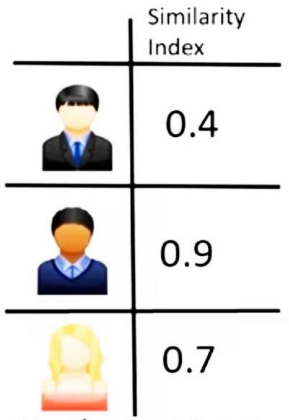
Week 12 :CF RS - / Algorithm /

User based



Week 12 :CF RS - / Algorithm /

User based



Similarity Matrix

sum_similarityIndex

∑

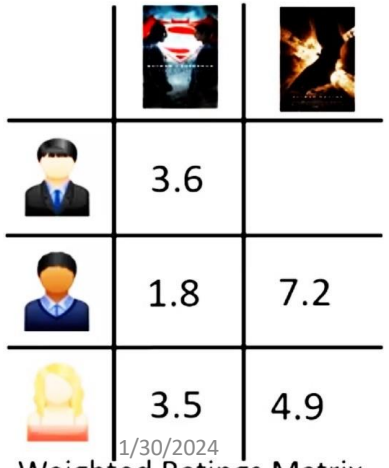
```
#Applies a sum to the topUsers after grouping it up by userId
tempTopUsersRating = topUsersRating.groupby('movieId').sum()[['similarityIndex','weightedRating']]
tempTopUsersRating.columns = ['sum_similarityIndex','sum_weightedRating']
tempTopUsersRating.head()
```

movieId	sum_similarityIndex	sum_weightedRating
1	38.376281	140.800834
2	38.376281	96.656745
3	10.253981	27.254477
4	0.929294	2.787882
5	11.723262	27.151751

Rating and Weight Matrix

∑

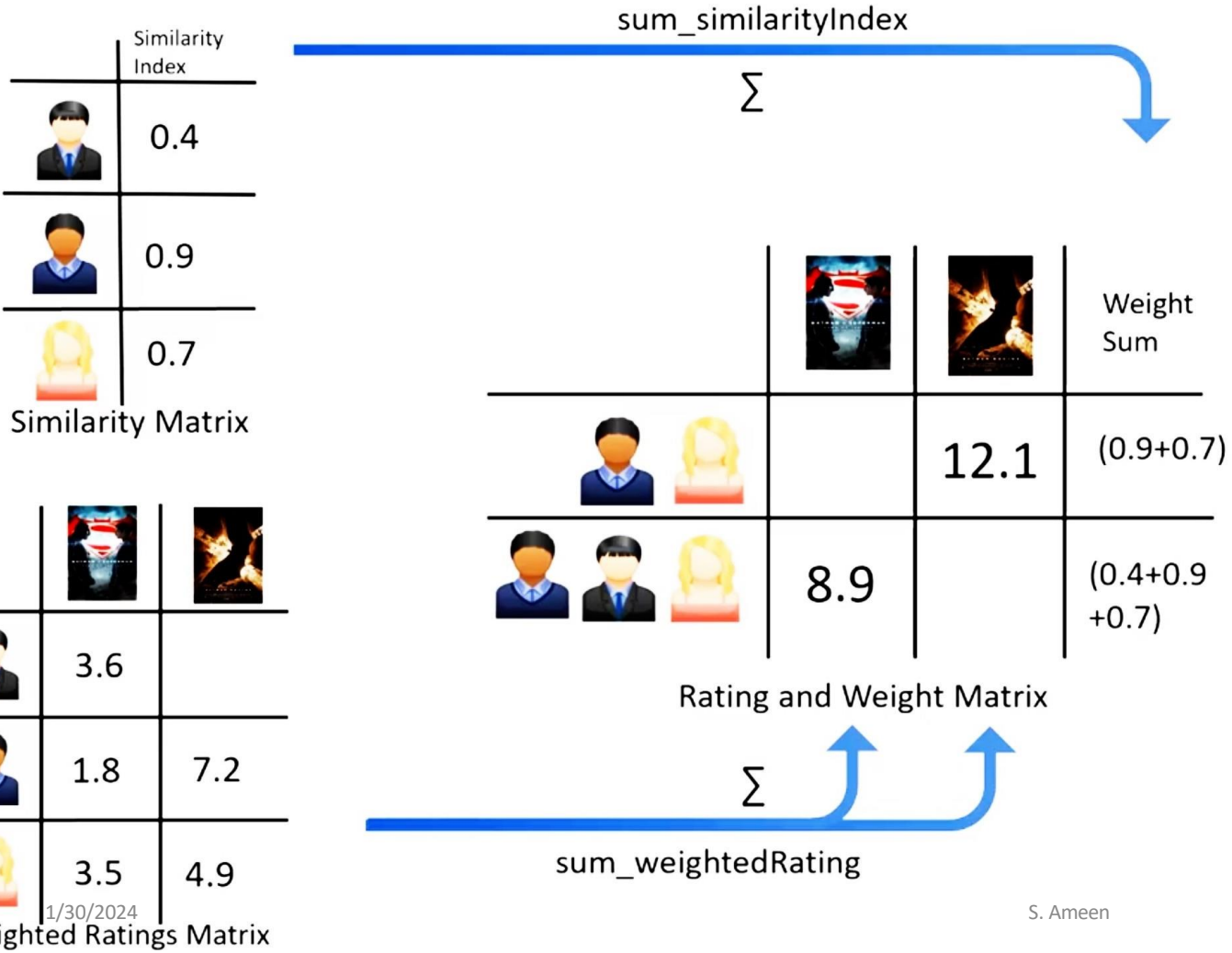
sum_weightedRating



Weighted Ratings Matrix

Week 12 :CF RS - / Algorithm /

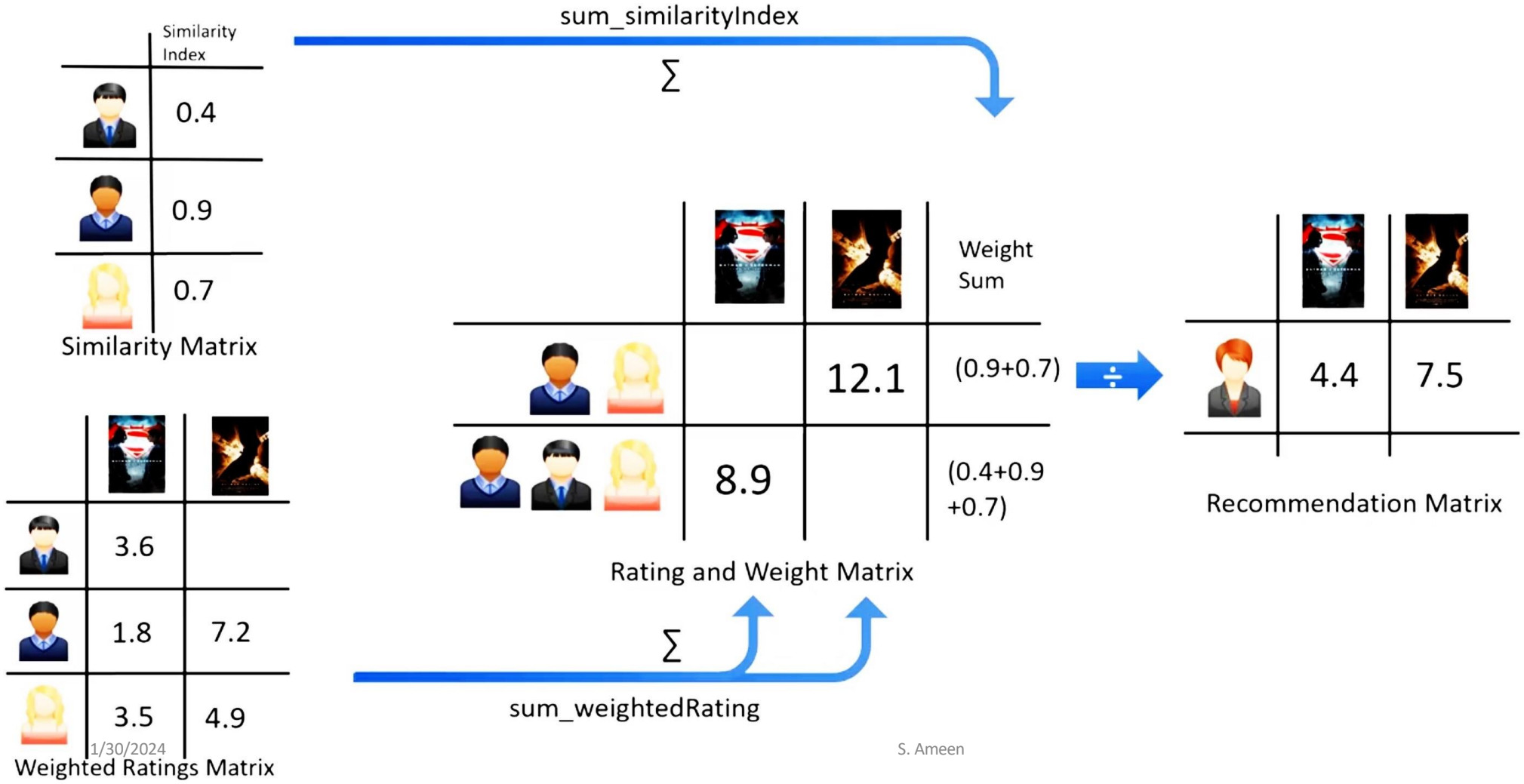
User based



movieId	sum_similarityIndex	sum_weightedRating
1	38.376281	140.800834
2	38.376281	96.656745
3	10.253981	27.254477
4	0.929294	2.787882
5	11.723262	27.151751

Week 12 :CF RS - / Algorithm /

User based



Week 12 :CF RS - / Algorithm /

User based

University of Salford MANCHESTER

```
#Creates an empty dataframe
recommendation_df = pd.DataFrame()
#Now we take the weighted average
recommendation_df['weighted average recommendation score'] = tempTopUsersRating['sum_weightedRating']/tempTopUsersRating['sum_similarityIndex']
recommendation_df['movieId'] = tempTopUsersRating.index
recommendation_df.head()
```

movieId	weighted average recommendation score	movieId
1	3.668955	1
2	2.518658	2
3	2.657941	3
4	3.000000	4
5	2.316058	5

Week 12 :CF RS - / Algorithm /

User based

Now let's sort it and see the top 20 movies that the algorithm recommended!

```
recommendation_df = recommendation_df.sort_values(by='weighted average recommendation score', ascending=False)
recommendation_df.head(10)
```

movieId	weighted average recommendation score	movieId
5073	5.0	5073
3329	5.0	3329
2284	5.0	2284
26801	5.0	26801
6776	5.0	6776
6672	5.0	6672
3759	5.0	3759
3769	5.0	3769
3775	5.0	3775
90531	5.0	90531

Recommendation Matrix

Week 12 :CF RS - / Algorithm /

User based

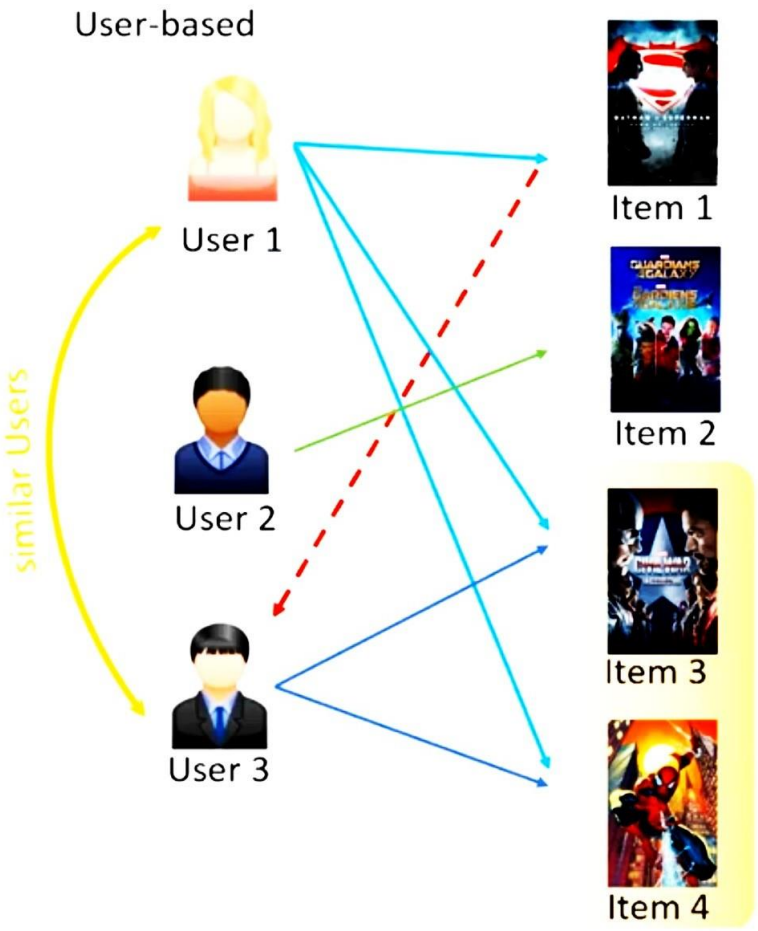

```
movies_df.loc[movies_df['movieId'].isin(recommendation_df.head(5)['movieId'].tolist())]
```

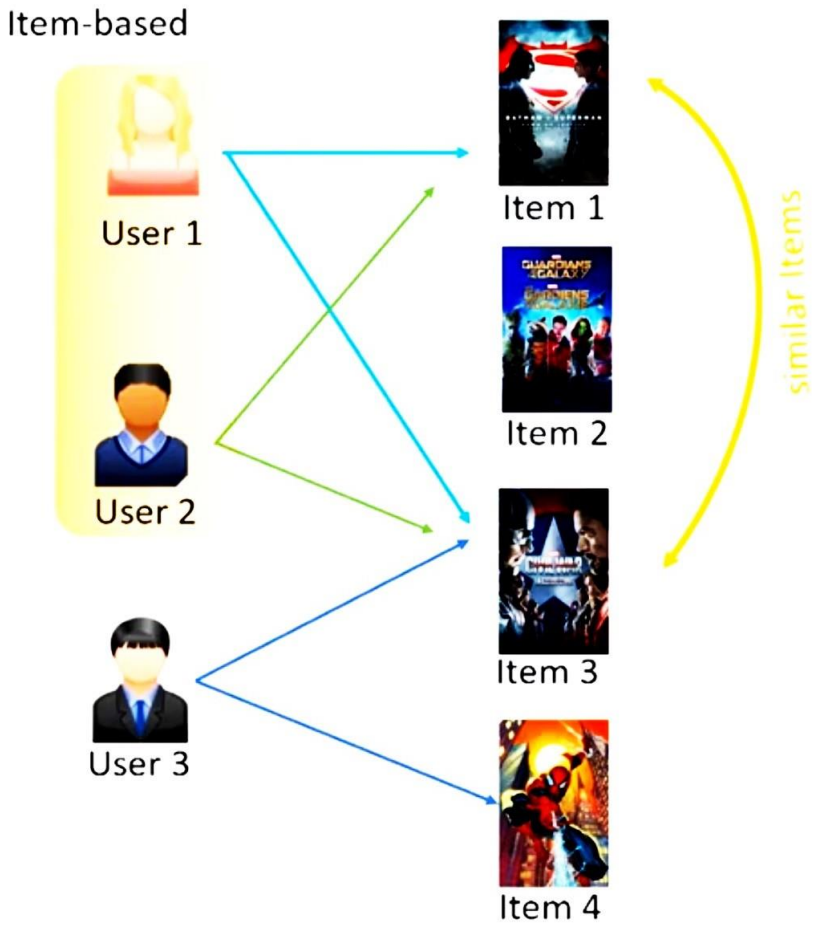
	movieId	title	year
2200	2284	Bandit Queen	1994
3243	3329	Year My Voice Broke, The	1987
4978	5073	Son's Room, The (Stanza del figlio, La)	2001
6667	6776	Lagaan: Once Upon a Time in India	2001
9064	26801	Dragon Inn (Sun lung moon hak chan)	1992

Recommendation Matrix

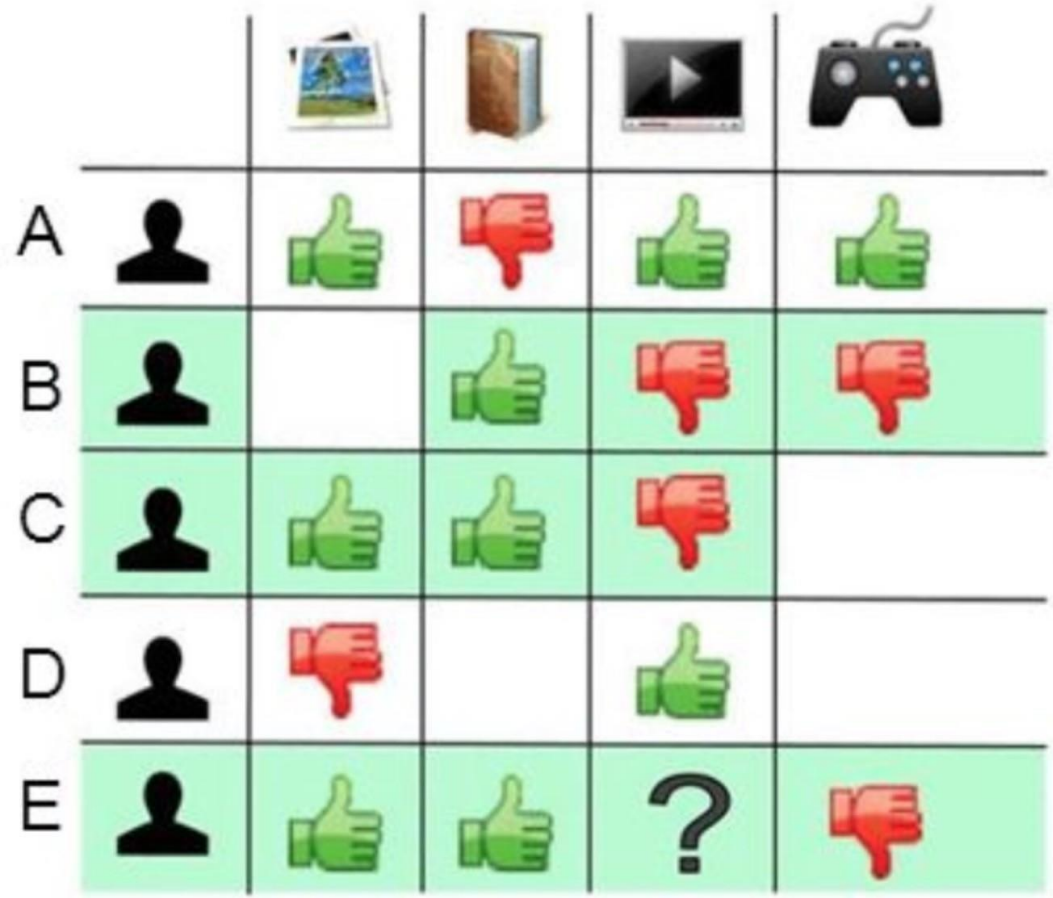
Week 12 :CF RS - / Algorithm / Item - based

User based





Week 12 :CF RS

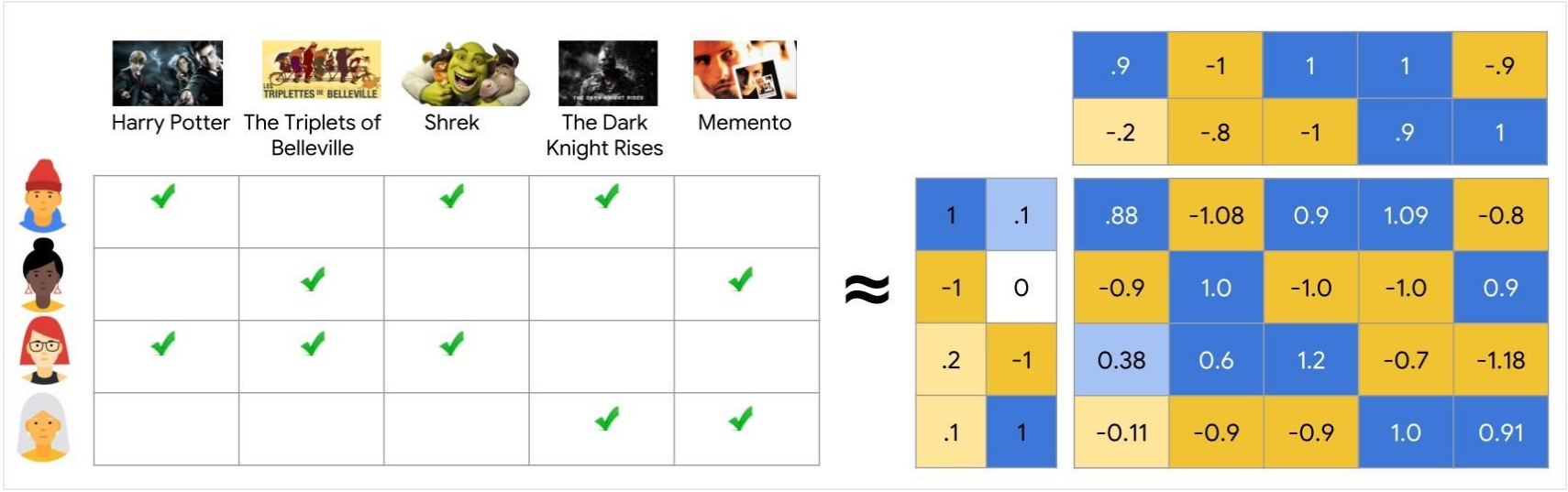


Week 12 :Recommendation as Association Rules

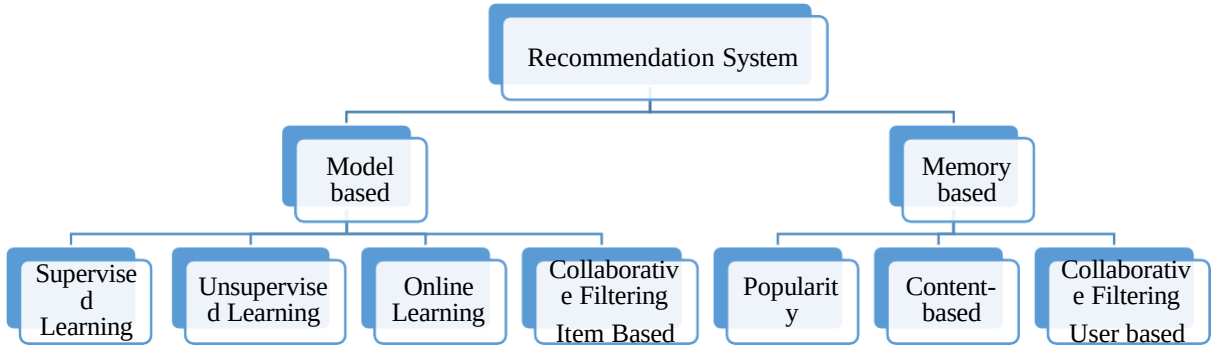
- **Ideas come from the market basket analysis (MBA)**
  - Let's go shopping!

  - What do my customer buy? Which product are bought together?
  - **Aim:** Find associations and correlations between the different items that customers place in their shopping basket

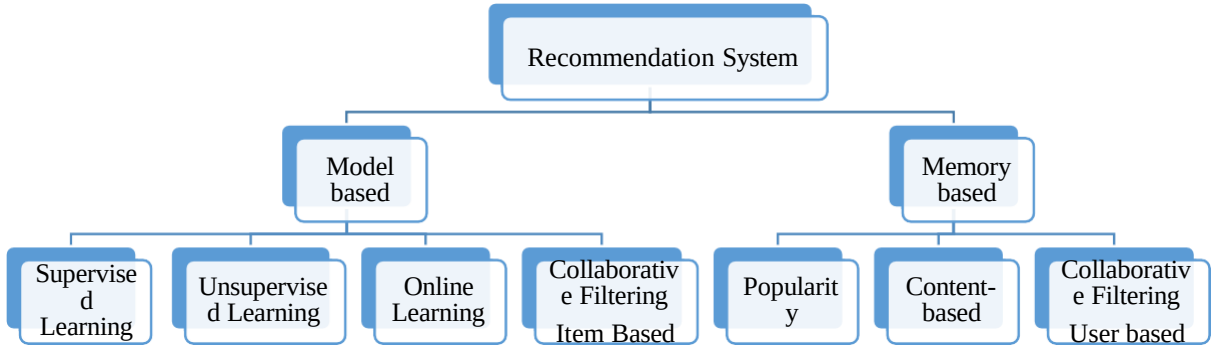
Week 12 :Recommendation as Matrix Factorization

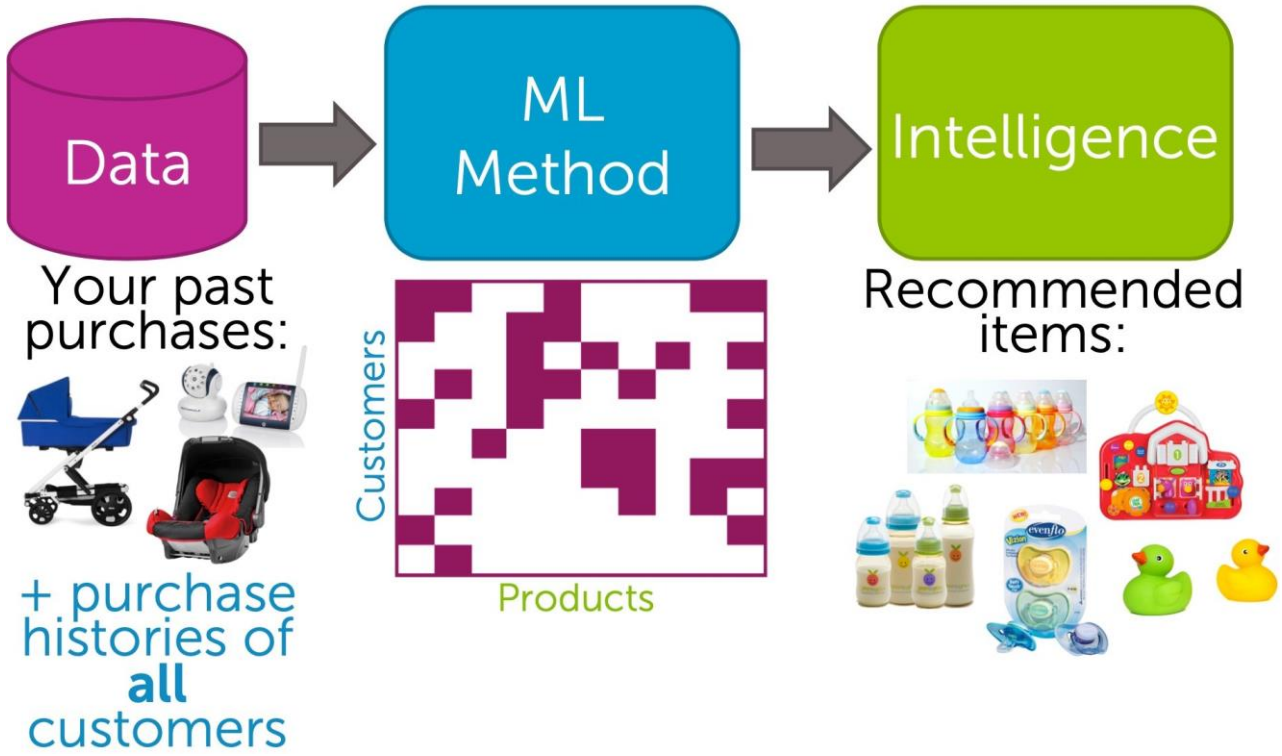


Week 12 - Summary

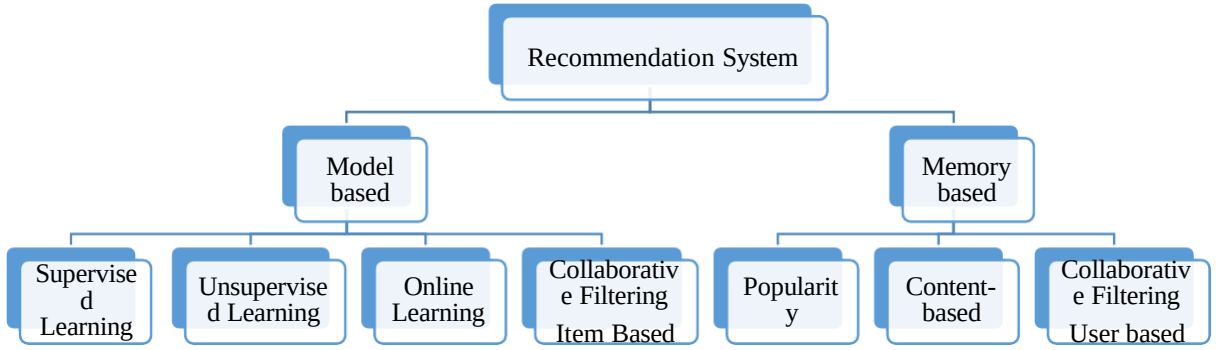


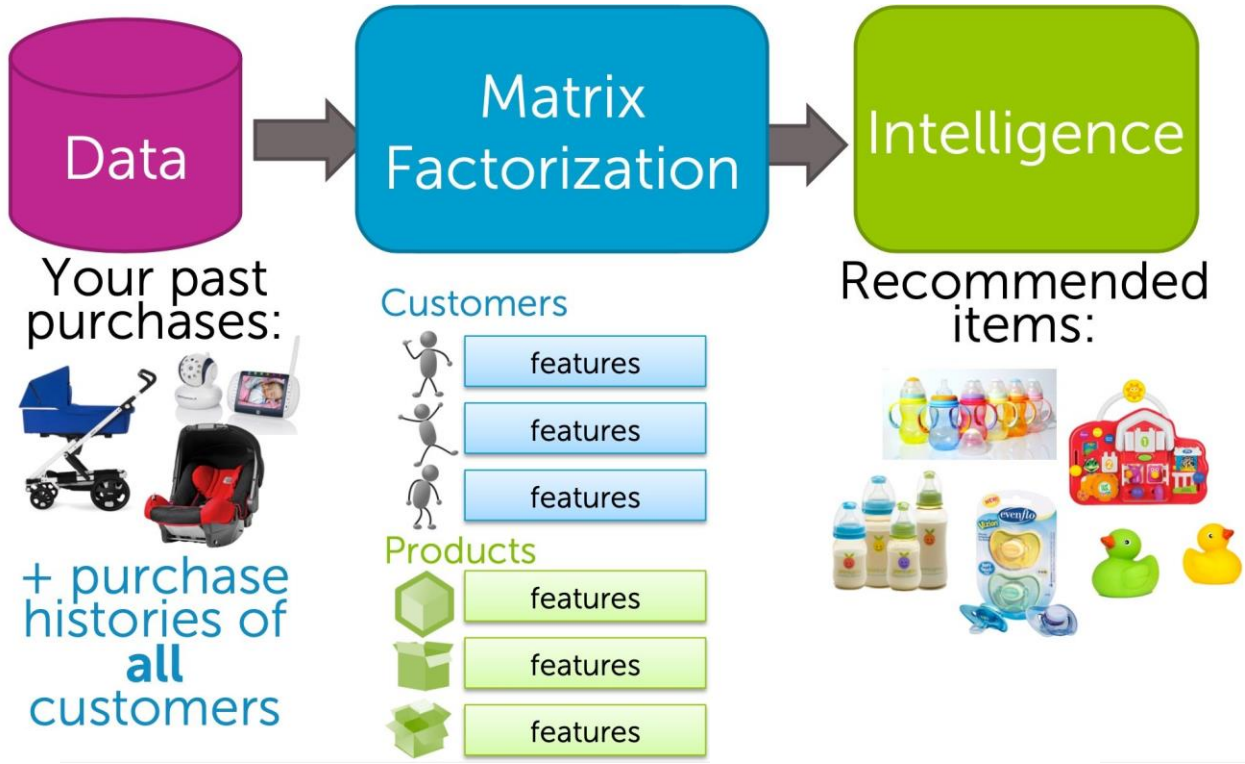
Week 12 - Summary



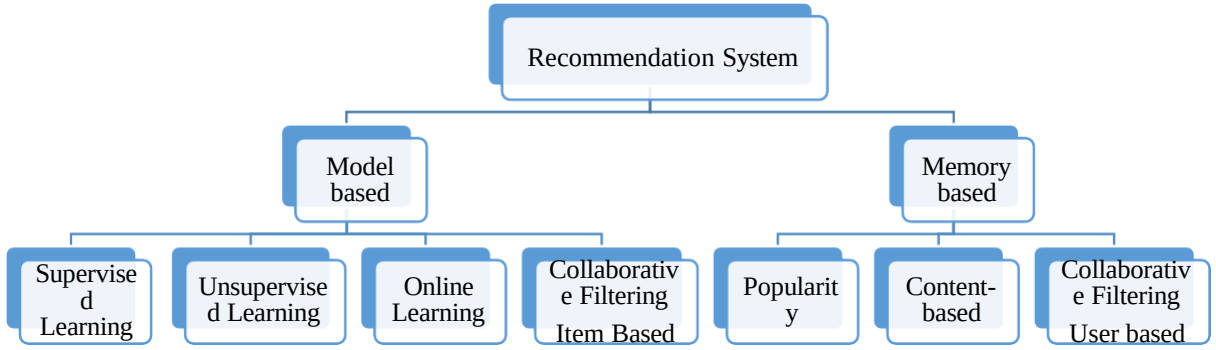


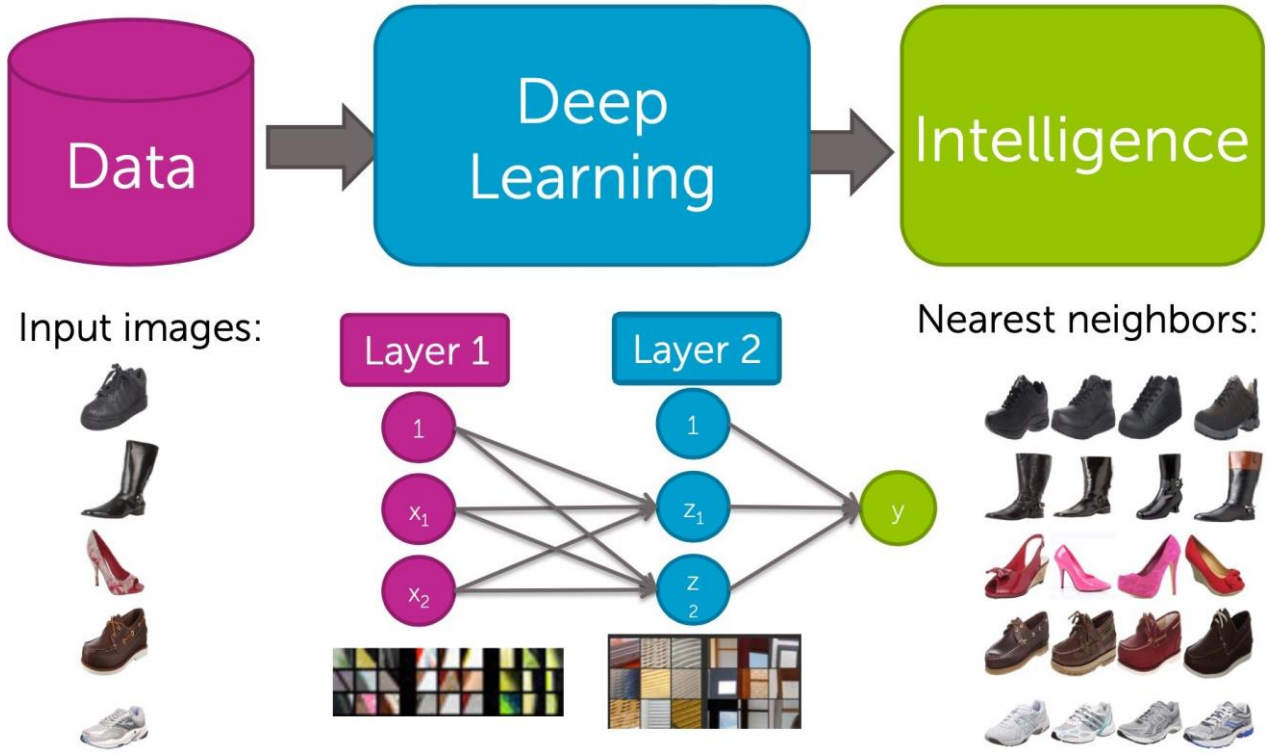
Week 12 - Summary





Week 12 - Summary





Week 12 Next Lecture

University of Salford MANCHESTER

More Deep Learning

Week 12

University of Salford MANCHESTER

Thanks for the Attention! ☺