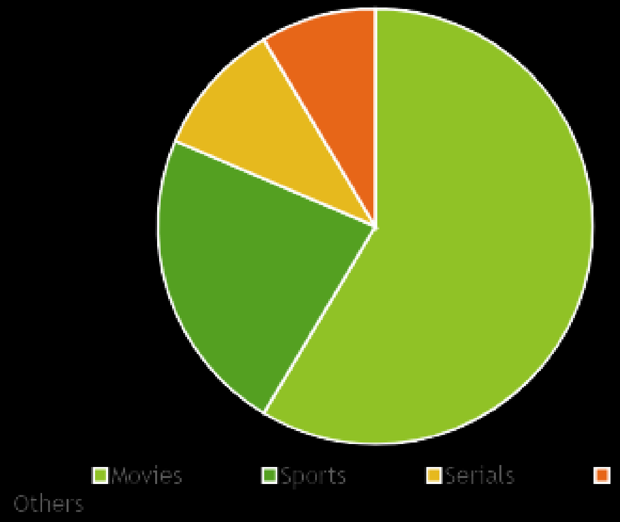
Recommendation system

SHASHANK.L.H PES1UG21CS557
SHREYAS.B PES1UG21CS579
SOURABHA.G PES1UG21CS608
SHRI GANESH PES1UG21CS612

What is it?

- Recommendation systems are a way of suggesting like or similar items and ideas to a users specific way

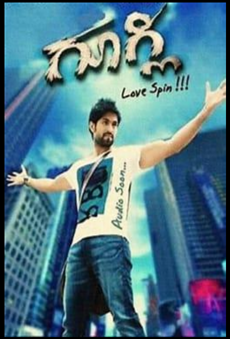


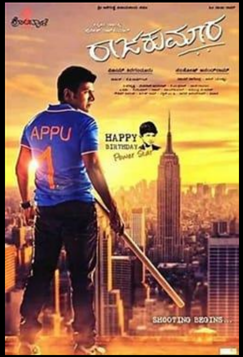
Where it is used ?

Example:

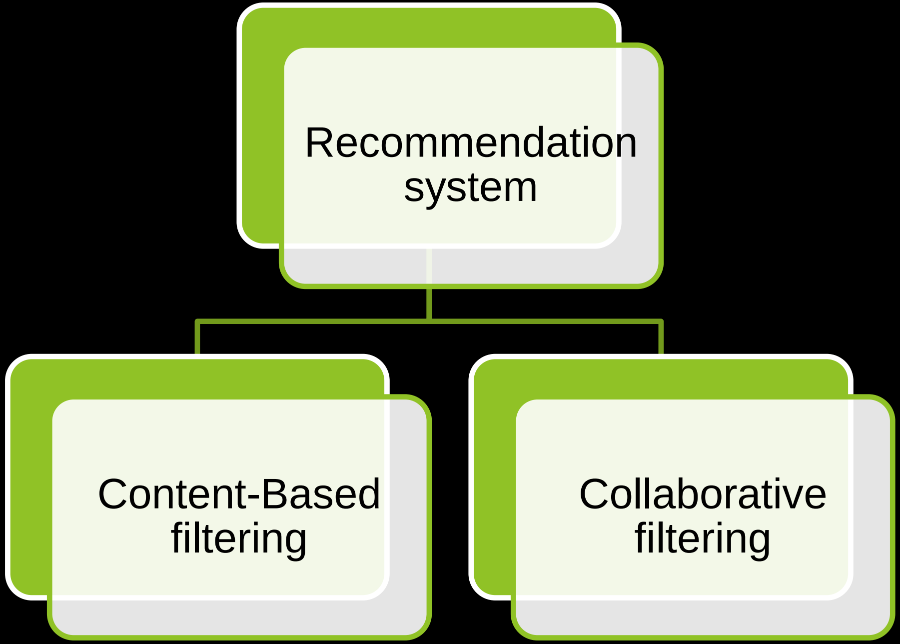
Shubman

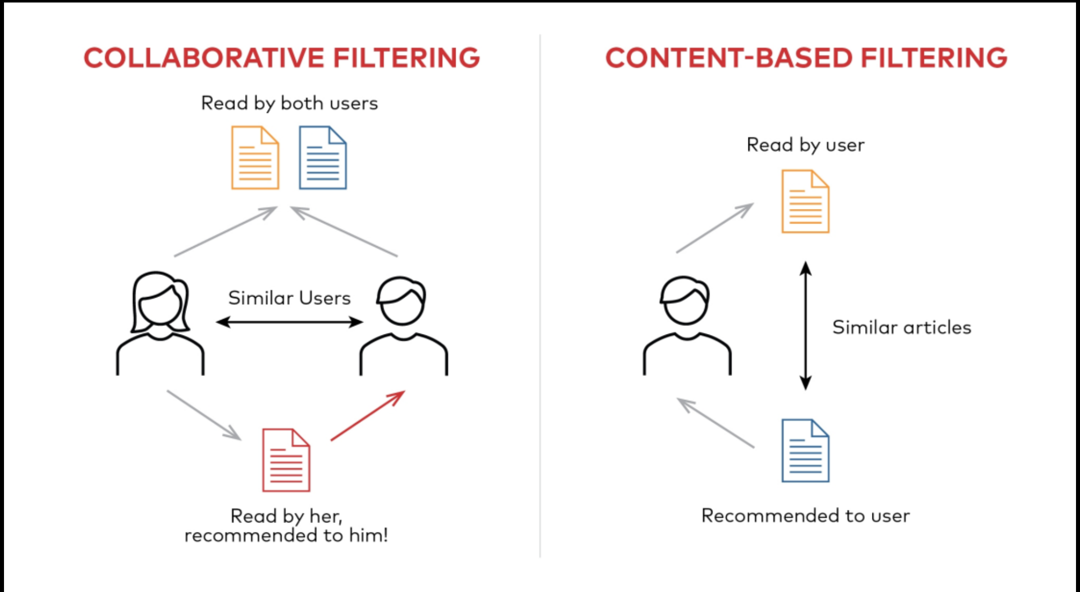
Sara





Types of Recommendation System



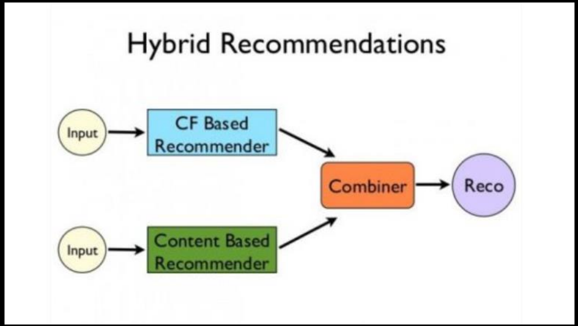

COLLABORATIVE FILTERING
Read by both users
Similar Users
Read by her,
recommended to him!
CONTENT-BASED FILTERING
Read by user
Similar articles
Recommended to user

Disadvantages

- The ‘cold start’ problem
- Inability to capture changes in user behavior
- Lack of Data
- Inactive participation of the users in surveys
- Changing Data

Solution to these problems...

Hybrid RS



- The hybrid recommendation system is a special type of system that used data of both collaborative data and content-based data simultaneously which helps to suggest a similar or close item to the users. Combining the two above approaches helps to resolve the big problems in more effective cases sometimes. In this, the system suggests similar items which are already used by the user or suggests the items which are likely to be used by another user with some similarities.

Lets Understand How it works…

Netflix ratings

	M1	M2	M3	M4	M5
	1	3	2	5	4
	2	1	1	1	5
	3	2	3	1	5
	2	4	1	5	2

Types of Peoples:

Group 1

Group 2 

Group 3

How do humans behave ?

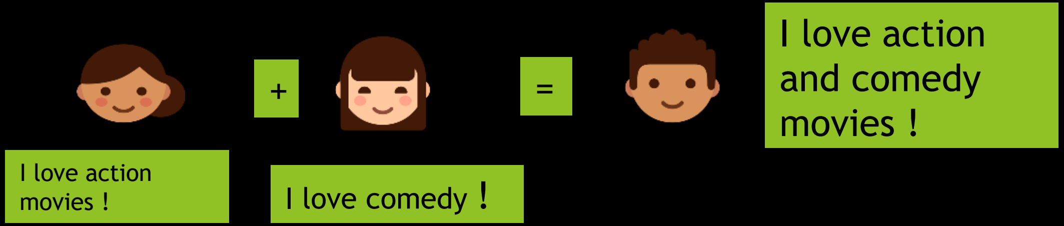
	M1	M2	M3	M4	M5
	3	3	3	3	3
	3	3	3	3	3
	3	3	3	3	3
	3	3	3	3	3

	M1	M2	M3	M4	M5
	1	3	2	5	4
	2	1	1	1	5
	3	2	3	1	5
	2	4	1	5	2

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Dependent Rows and Columns

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4



How machine treats user 1 & 3

How machine treats user 4

Q: How do we figure out all these depedencencies ?

Concept of matrix factorization

Matrix Factorization

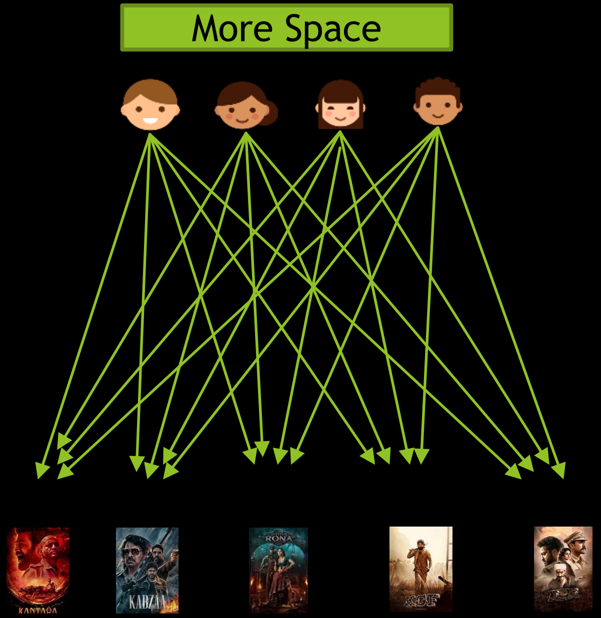
	comedy	action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

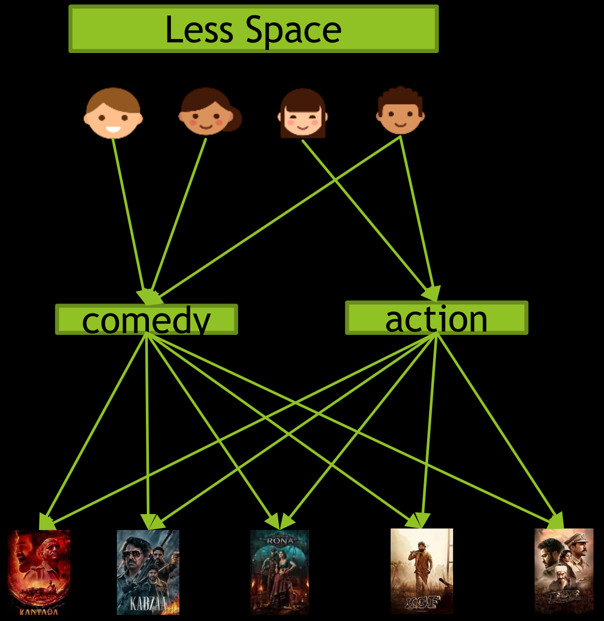
	comedy	action
	1	0
	0	1
	1	0
	1	1

=

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

Uses: Storage





Let's take example:

Without Factorization

720 CSE students

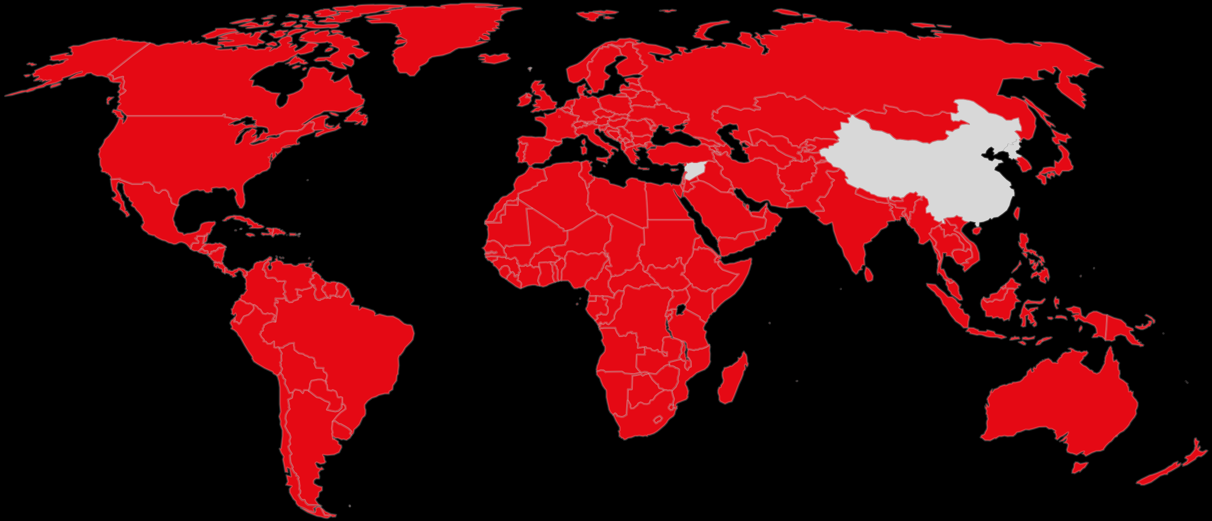
- 720 X 720 =5,18,400

With Factorization

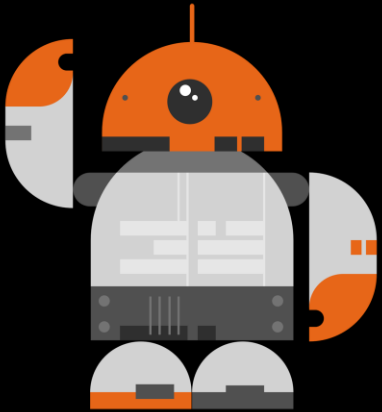
720 CSE students with 100 features

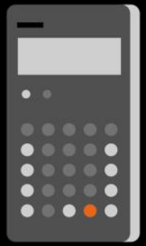
- 720 X 100 =72000
- (with 100 features)
- 100X100=10,000
- 72,000 +10,000=82,000

Can you think of it !
4.9Billion X 4.9Billion



Let’s Understand Error function





F1	1.2	3.1	0.3	2.5	0.2
F2	2.4	1.5	4.4	0.4	1.1

	comedy	action
	0.2	0.5
	0.3	0.4
	0.7	0.8
	0.4	0.5

	M1	M2	M3	M4	M5
	1.44	1.37	2.26	0.7	0.59
	1.32	1.53	1.85	0.91	0.5
	2.76	3.37	3.37	2.07	1.02
	1.68	1.99	2.32	1.2	0.63

Let’s Compare

	M1	M2	M3	M4	M5
	1.44	1.37	2.26	0.7	0.59
	1.32	1.53	1.85	0.91	0.5
	2.76	3.37	3.37	2.07	1.02
	1.68	1.99	2.32	1.2	0.63

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	3	4	4

This leads to error function…

Error Function

	M1	M2	M3	M4	M5
	1.44	1.37	2.26	0.7	0.59
	1.32	1.53	1.85	0.91	0.5
	2.76	3.37	3.37	2.07	1.02
	1.68	1.99	2.32	1.2	0.63

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	3	4	4

The Sum of Square of difference between the actual value and obtained value is called as Error function.
The Derivative of the error function tells us how much to increase or decrease the features or other parameters .

Equations used...

$$r_{ui} = x_i^T . y_u$$

$$Min(x, y) \sum_{(u, i) \in K} (r_{ui} - x_i^T . y_u)^2$$

$$Min(x, y) \sum_{(u, i) \in K} (r_{ui} - x_i^T . y_u)^2 + \lambda(||x_i||^2 + ||y_u||^2)$$

$$Min(x, y, b_i, b_u) \sum_{(u, i) \in K} (r_{ui} - x_i^T . y_u - \mu - b_i - b_u)^2 + \lambda(||x_i||^2 + ||y_u||^2 + b_i^2 + b_u^2)$$

- each item be represented by a vector *xi* & each user is represented by a vector *yu*.
- dot product gives the expected rating
- The *xi* and *yu* can be obtained in a manner that the square error difference between their dot product and the expected rating in the user-item matrix is minimum

Continued…

- In order to let the model generalise well and not overfit the training data, a regularisation term is added as a penalty to the above formula.
- In order to reduce the error between the value predicted by the model and the actual value, the algorithm uses a   bias term.
- This equation is the main component of the algorithm which works for singular value decomposition based recommendation system.

Concept of SVD

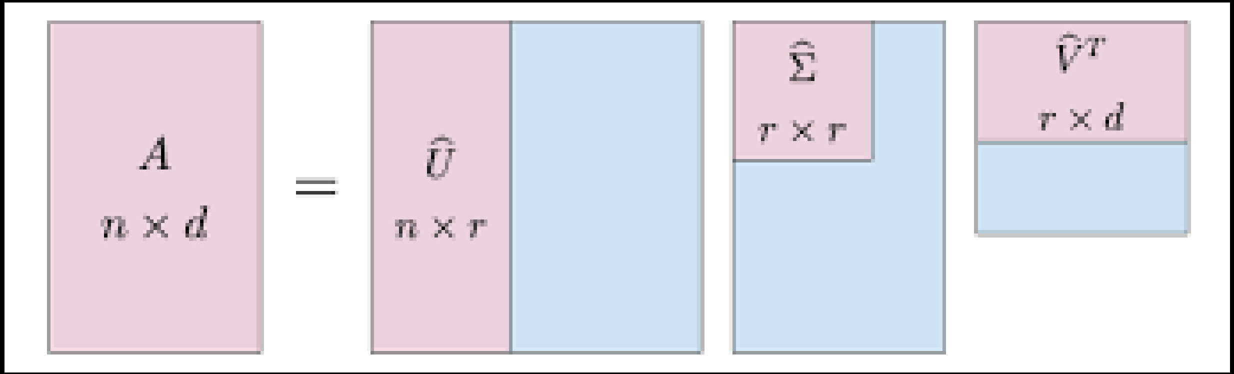
- **Singular Value Decomposition:-** $A = USV^T$
- The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning
- In the context of the recommender system, the SVD is used as a **collaborative filtering technique.**
- It also a type of **MATRIX FACTORIZATION TECHNIQUE**
- It uses a matrix structure where each row represents a user, and each column represents an item. The elements of this matrix are the ratings that are given to items by users.
- The singular value decomposition is a method of decomposing a matrix into three other matrices .
- Where A is a $m \ x \ n$ utility matrix.
- U is a $m \ x \ r$ orthogonal left singular matrix, which represents the relationship between users and latent factors. It is a user space.
- S is a $r \ x \ r$ diagonal matrix, which describes the strength of each latent factor

Singular Value Decomposition(Continued)

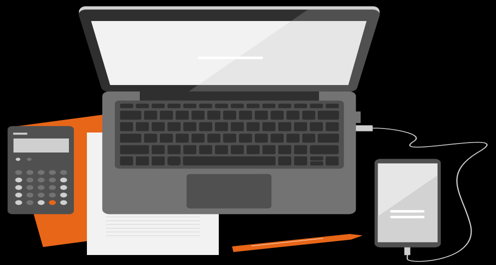
- V is a r x n diagonal right singular matrix, which indicates the similarity between items and latent factors. It is a item space
- The latent factors here are the characteristics of the items, for example, the genre of the music.
- The SVD decreases the dimension of the utility matrix A by extracting its latent factors.
- It maps each user and each item into a r-dimensional latent space. This mapping facilitates a clear representation of relationships between users and items .

$$A = USV^T$$

GRAPHICAL REPRESENTATION OF SVD DIMENSIONALITY REDUCTION



Code:



```
import numpy as np
from scipy.linalg import svd

# Generate temporary data
matrix = np.array([[5, 3, 0, 1,], [4, 0, 0, 1], [ 1, 1, 0, 5], [1, 0, 0, 4],
[0, 1, 5, 4]])
items = ['item1', 'item2', 'item3', 'item4']
print("Original matrix:\n" ,matrix)
U, S, V = svd(matrix, full_matrices=False)
k = 3 #key features to keep/ reduced dimensions
print("\nU:-",U)
print("\nS:-",S)
print("\nV:-",V)
```

```
$ python rec1.py
Original matrix:
 [[5 3 0 1]
 [4 0 0 1]
 [1 1 0 5]
 [1 0 0 4]
 [0 1 5 4]]

U:- [[-0.43689593 -0.66924125  0.29627751 -0.48637475]
 [-0.29717498 -0.44308727  0.05015708  0.79591123]
 [-0.51589728  0.13631518 -0.54893193 -0.28612203]
 [-0.39999635  0.11077382 -0.48349385  0.20569271]
 [-0.54282768  0.5700326   0.61205501  0.0760895 ]]

S:- [9.03171974 6.22925557 3.77397038 1.83890217]

V:- [[-0.47488998 -0.26234348 -0.3005118  -0.78444124]
 [-0.78203025 -0.20891356  0.45754472  0.36801718]
 [ 0.17212379  0.25224247  0.81089006 -0.49920382]
 [ 0.36507752 -0.907692    0.20688838  0.00329281]]
```

```python
matrix_svd = U[:, :k] @ np.diag(S[:k]) @ V[:k, :]
print("SVD Matrix:\n", matrix_svd)
```

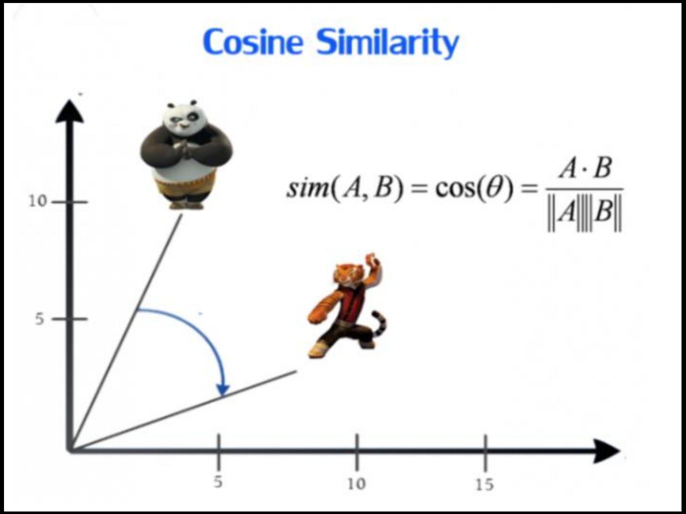
```
SVD Matrix:
 [[ 5.32652372  2.18816428  0.18504005  1.00294508]
 [ 3.46567149  1.32850063 -0.30280243  0.99518063]
 [ 1.19208569  0.52241748  0.10885441  5.00173251]
 [ 0.86190988  0.34333338 -0.07825527  3.9987545 ]
 [-0.05108206  1.1270053   4.97105194  3.99953927]]
```

```python
def item_similarity(matrix):
    similarity = np.dot(matrix.T,matrix)/(norm(matrix.T)*norm(matrix))
    np.fill_diagonal(similarity, 0)
    return similarity
similarity_svd = item_similarity(matrix_svd)
print("similarity :\n",similarity_svd)
```

```
similarity :
 [[ 0.          0.12717853 -0.00189729  0.13368105]
 [ 0.12717853  0.          0.04185925  0.08921591]
 [-0.00189729  0.04185925  0.          0.14855095]
 [ 0.13368105  0.08921591  0.14855095  0.        ]]
```

Here we are using cosine similarity to find the similarity between the items.

Cosine Similarity



Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

The cosine similarity between two vectors A and B is calculated as the dot product of A and B divided by the product of the magnitudes of A and B.

The resulting value ranges from -1 to 1, with 1 indicating that the two vectors are identical and -1 indicating that they are completely dissimilar.

```python
def recommend(user, matrix, similarity, items):
    user_index = user - 1
    user_ratings = matrix[user_index, :]
    item_scores = np.zeros((matrix.shape[1],))
    for item in range(matrix.shape[1]):
        item_sum = 0
        for other_item in range(matrix.shape[1]):
            if user_ratings[other_item] == 0 or similarity[item][other_item] == 0:
                continue
            item_sum = user_ratings[other_item] * similarity[item][other_item]
        item_scores[item] = item_sum / np.abs(similarity[item]).sum()
    recommendations = [items[i] for i in np.argsort(item_scores)[::-1]]
    print(item_scores)
    return recommendations

n=int(input("Enter the user: "))
user =n
recommendations = recommend(user, matrix, similarity_svd, items)
print(f"Recommendations for user {user}: {recommendations}")
```

```
Enter the user: 3
[1.39993303 1.46994214 2.82446309 0.38233002]
Recommendations for user 3: ['item3', 'item2', 'item1', 'item4']
```

Resources

- https://analyticsindiamag.com/singular-value-decomposition-svd-application-recommender-system/
- https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00592-5
- https://jaketae.github.io/study/svd/


THANK YOU