

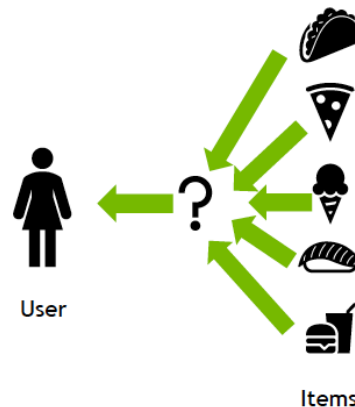
CS2012 – MACHINE LEARNING TECHNIQUES

RECOMMENDATION SYSTEMS

Muthu Palaniappan M

RECOMMENDATION SYSTEM

- A **recommendation system** (or recommender system) is a class of machine learning that uses data to help predict, narrow down, and find what people are looking for among an exponentially growing number of options.
- These can be based on various criteria including past purchases, search history, demographic information, and other factors.
- Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.



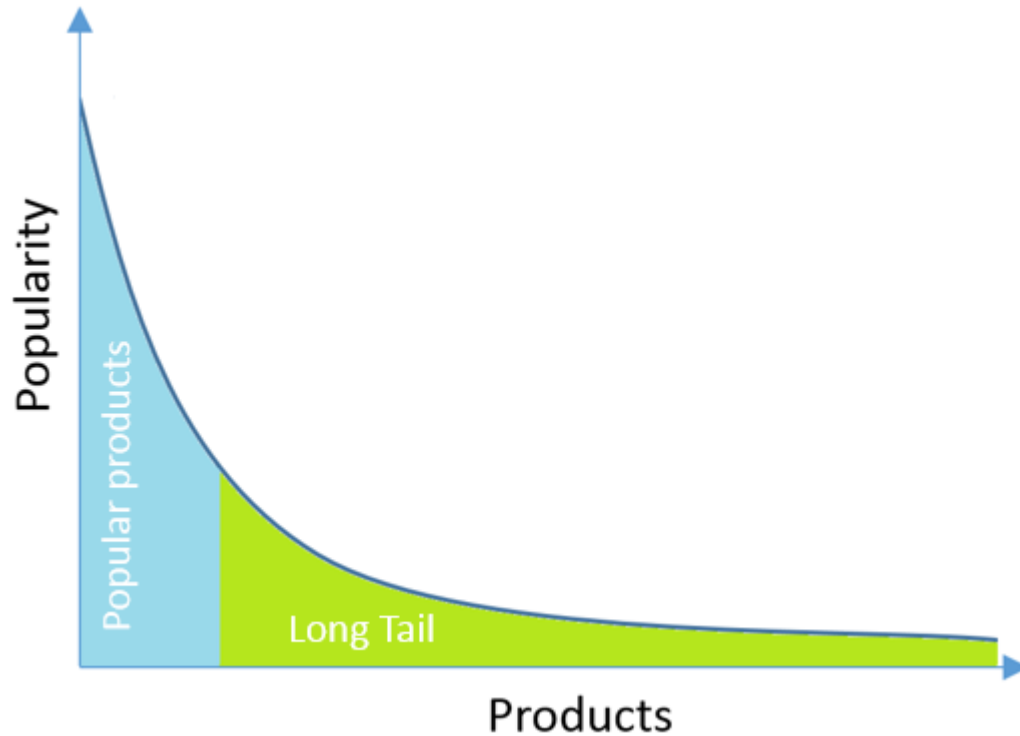
EXAMPLES OF RECOMMENDATION ENGINE

Some of the great examples of recommendation engine usages are:

- **Facebook** – It recommends you under the section – “People You May Know”.
- **Netflix** – It recommends you under the section – “Other Movies You May Enjoy”.
- **LinkedIn** – It helps you find right jobs under the section – “Jobs You May Be Interested In”.
- **Amazon** – it helps suggests products under the section – “Inspired by your shopping trends”.
- **YouTube** – It helps suggests videos under the section – “Recommended Videos”.



THE LONG-TAIL PROBLEM



Long-Tail graph shows the distributions of popularity among the products in the market.

- Products on the left side (blue area) are called as the popular products.
- Products on the Long Tail (green area) are thought to be unpopular or the new products in the market.
- The threshold which discriminates the popular and unpopular items in the market is a hyper-parameter for a merchant.
- Same and known items can make the customers bored. Therefore, a good recommendation systems should provide diversity.
- If you recommend items in the long tail and the people like them then you are effectively doing “discovery”

TF-IDF VECTORIZER

- The problem with natural language processing is that the data is in the form of raw text, so that the text need to be transformed into the vector.
- The process of transforming text into a vector is called as text vectorization.
- Text vectorization algorithm namely TF-IDF vectorizer, which is a very popular approach for traditional machine learning algorithms can help in transforming text into vectors.

TF*IDF

TF*IDF=TERM FREQUENCY * INVERSE
DOCUMENT FREQUENCY

TF-IDF

- Term frequency-inverse document frequency is a text vectorizer that transforms the text into a usable vector. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF).
- **Term frequency:**
 - The term frequency is the number of occurrences of a specific term in a document.
 - Term frequency represents every text from the data as a matrix whose rows are the number of documents and columns are the number of distinct terms throughout all documents.

Text 1	i love natural language processing but i hate python
Text 2	i like image processing
Text 3	i like signal processing and image processing

	and	but	hate	i	image	language	like	love	natural	processing	python	signal
Text 1	0	1	1	2	0	1	0	1	1	1	1	0
Text 2	0	0	0	1	1	0	1	0	0	1	0	0
Text 3	1	0	0	1	1	0	1	0	0	2	0	1

■ Inverse document frequency (IDF):

- It is the weight of the term.
- idf_i - IDF score for term I
- df_i - The number of documents containing term I
- n - The total number of documents.
- Higher the df of the term lower the IDF
- When the number of DF is equal to n which means that the term appears in all documents, the IDF will be zero, since $\log(1)$ is zero.

$$idf_i = \log\left(\frac{n}{df_i}\right)$$

TF-IDF

The TF-IDF score as the name suggests is just a multiplication of the term frequency matrix with its IDF, it can be calculated as follow:

$$w_{i,j} = tf_{i,j} \times idf_i$$

w_{ij} is TF-IDF score for term i in document j , tf_{ij} is term frequency for term i in document j , and idf_i is IDF score for term i .

Text 1	i love natural language processing but i hate python
Text 2	i like image processing
Text 3	i like signal processing and image processing

TF-IDF

■ Step 1:

Create a term frequency matrix where rows are documents and columns are distinct terms throughout all documents. Count word occurrences in every text.

	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
<i>Text 1</i>	0	1	1	2	0	1	0	1	1	1	1	0
<i>Text 2</i>	0	0	0	1	1	0	1	0	0	1	0	0
<i>Text 3</i>	1	0	0	1	1	0	1	0	0	2	0	1

■ Step 2:

Compute inverse document frequency (IDF) using the previously explained formula.

<i>Term</i>	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
<i>IDF</i>	0.47712	0.47712	0.4771	0	0.1760913	0.477121	0.1760913	0.477121	0.47712125	0	0.477121	0.477121

TF-IDF EXAMPLE

■ Step 3:

Multiply TF matrix with IDF respectively

	<i>and</i>	<i>but</i>	<i>hate</i>	<i>i</i>	<i>image</i>	<i>language</i>	<i>like</i>	<i>love</i>	<i>natural</i>	<i>processing</i>	<i>python</i>	<i>signal</i>
<i>Text 1</i>	0	0.47712	0.4771	0	0	0.477121	0	0.477121	0.47712125	0	0.477121	0
<i>Text 2</i>	0	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0
<i>Text 3</i>	0.47712	0	0	0	0.1760913	0	0.1760913	0	0	0	0	0.477121

It gives the rare term high weight and gives the common term low weight.

THE UTILITY MATRIX

- In a recommendation-system application there are two classes of entities, which we shall refer to as **users** and **items**.
- Utility matrix shows the degree of preference of that user for that item.
- Example: In Fig we see an example utility matrix, representing users' ratings of movies on a 1–5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for Harry Potter I, II, and III, TW for Twilight, and SW1, SW2, and SW3 for Star Wars episodes 1, 2, and 3. The users are represented by capital letters A through D.

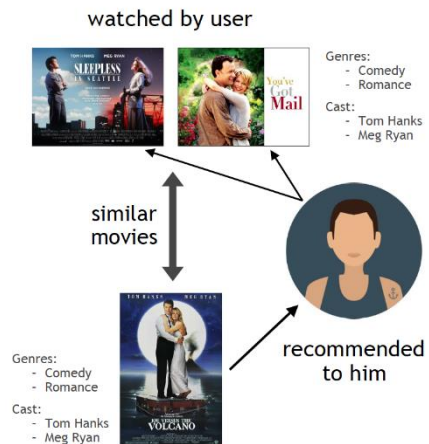
	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

The goal of a recommendation system is to predict the blanks in the utility matrix. For example, would user A like SW2?

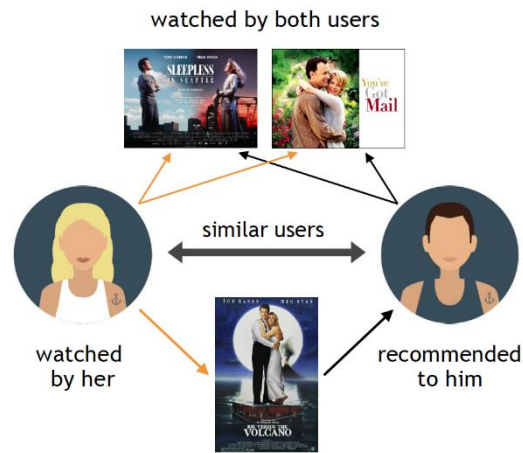
TYPES OF RECOMMENDATION SYSTEMS

- There are three types of recommendation systems.
 - Content-based recommendation systems.
 - Collaborative filtering systems.
 - Hybrid recommendation systems.

Content-based Filtering



Collaborative Filtering



Hybrid Recommendations

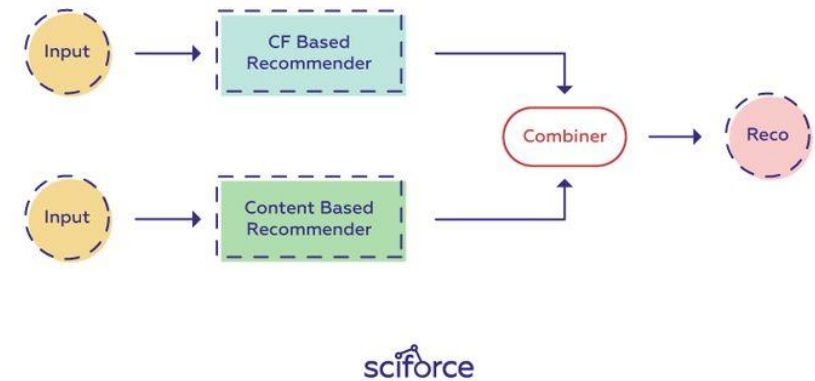
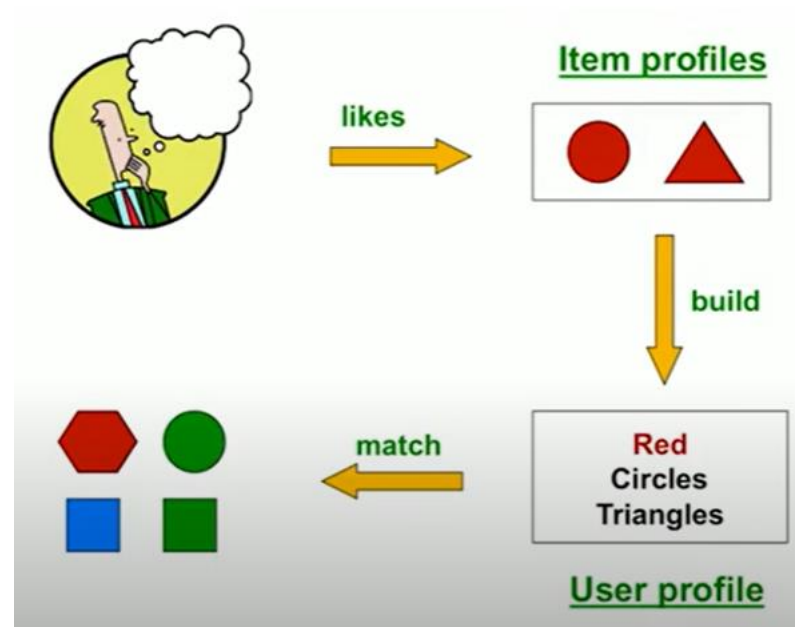


Image Source: NVidia and sciforce

CONTENT BASED RECOMMENDATIONS

- It uses the attributes or the features of the item to recommend other similar items to the user's preferences.
- This approach is based on similarity of item and user features, given information about a user and items they have interacted with (e.g. a user's age, the category of a restaurant's cuisine, the average review for a movie), model the likelihood of a new interaction.



CONTENT BASED RECOMMENDATIONS

- In a content-based recommendation system we must construct for each item a profile which is called **item-profile**.
- It is a record or collection of records representing important characteristics of that item.
- In simple cases, the profile consists of some characteristics of the item that are easily discovered.
- For an example consider the features of the movie that might be relevant to recommendation systems.
 - Set of actors.
 - The director.
 - The genre or the general type of the movie.

CONTENT BASED RECOMMENDATIONS

- **User profile** is the collection of information that describes the interest, preferences and behaviour of a particular user.
- It is created by analyzing the user's interactions with the system, such as items they have rated, purchased, or viewed.
- The user profile can be continually updated as the user interacts with the system, enabling the recommendation system to adapt and provide more accurate and relevant recommendations over time.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
<i>A</i>	4			5	1		
<i>B</i>	5	5	4				
<i>C</i>				2	4	5	
<i>D</i>		3					3

CASE STUDY – BINARY REPRESENTATION – STEP1

- First step: For each user, we will track user engagement (like / share / comments) with various articles.

Articles	Big Data	R	Python	Machine Learning	Learning Paths	Total attributes	User 1	User 2
Article 1	1	0	1	0	1	3	1	-1
Article 2	0	1	1	1	0	3	-1	1
Article 3	0	0	0	1	1	2		
Article 4	0	0	1	1	0	2		1
Article 5	0	1	0	0	0	1		
Article 6	1	0	0	1	0	2	1	
DF	2	2	3	4	2			

- Article 1 is about the Big Data, Python and Learning paths. Similarly, article 2 is about the R, Python and Machine Learning.
- User 1 has liked article 1, article 6 but not article 2. Similarly, user2 liked article 2, article 4 but not article 1.
- User 1 not engaged with article 3,4 and 5.

CASE STUDY – NORMALIZE – STEP2

- We can perform normalization by dividing the term by the square root of number of attributes in the article.

Articles	Big Data	R	Python	Machine Learning	Learning Paths	Total attributes
Article 1	0.577350269	0	0.577350269	0	0.577350269	3
Article 2	0	0.577350269	0.577350269	0.577350269	0	3
Article 3	0	0	0	0.707106781	0.707106781	2
Article 4	0	0	0.707106781	0.707106781	0	2
Article 5	0	1	0	0	0	1
Article 6	0.707106781	0	0	0.707106781	0	2

CASE STUDY –FINDING USER PROFILE– STEP3

- To find the user profile: Each attribute column is multiplied with each user attribute value.
- This is simply the dot product of vectors which will produce a user profile.

Articles	Big Data	R	Python	Machine Learning	Learning Paths	Total attributes	User 1	User 2
Article 1	0.577350269	0	0.577350269	0	0.577350269	3	1	-1
Article 2	0	0.577350269	0.577350269	0.577350269	0	3	-1	1
Article 3	0	0	0	0.707106781	0.707106781	2		
Article 4	0	0	0.707106781	0.707106781	0	2		1
Article 5	0	1	0	0	0	1		
Article 6	0.707106781	0	0	0.707106781	0	2	1	
User Profiles								
User1	1.28445705	-0.577350269	0	0.129756512	0.577350269			
User2	-0.577350269	0.577350269	0.707106781	1.28445705	=SUMPRODUCT(F16:F21, \$K\$10:\$K\$21)			
SUMPRODUCT(array1, [array2], [array3], [array4], ...)								

- Thus, user 1 likes articles on big data most(highest score of 1.28) followed by learning paths and then machine learning. Similarly, user 2 like articles on machine learning the most.

CASE STUDY –WEIGHTED SCORES– STEP4

- Now we have the user profile vectors and the article vectors, let's use these to predict which articles will be similar to the user's taste.

Articles	big data	R	python	machine learning	learning paths	User 1	User 2	Pred User1	PredUser2
Article 1	0.577350269	0	0.577350269	0	0.577350269	1	-1	0.751333312	
Article 2	0	0.577350269	0.577350269	0.577350269	0	-1	1	-0.20317834	
Article 3	0	0	0	0.707106781	0.707106781			0.321864985	
Article 4	0	0	0.707106781	0.707106781	0		1	0.036511676	
Article 5	0	1	0	0	0			-0.40355052	
Article 6	0.707106781	0	0	0.707106781	0	1	=SUMPRODUCT(B36:F36, \$B\$36:\$F\$36, \$B\$42:\$F\$42)		
							SUMPRODUCT(array1, [array2], [array3], [array4], [array5], ...)		
User Profiles									
User1	1.28445705	-0.577350269	0	0.129756512	0.577350269				
User2	-0.577350269	0.577350269	0.707106781	1.28445705	-0.577350269				
DF	2	2	3	4	2				
IDF	0.698970004	0.698970004	0.52287875	0.397940009	0.698970004				

- The dot product of article vectors and IDF vectors gives us the weighted scores of each article. These weighted scores are again used for a dot product with the user profile vector (user 1 here). This gives a probability that the user will like a particular article. For article 1, the probability is 75%.

PROS AND CONS OF CONTENT BASED SYSTEMS

PROS

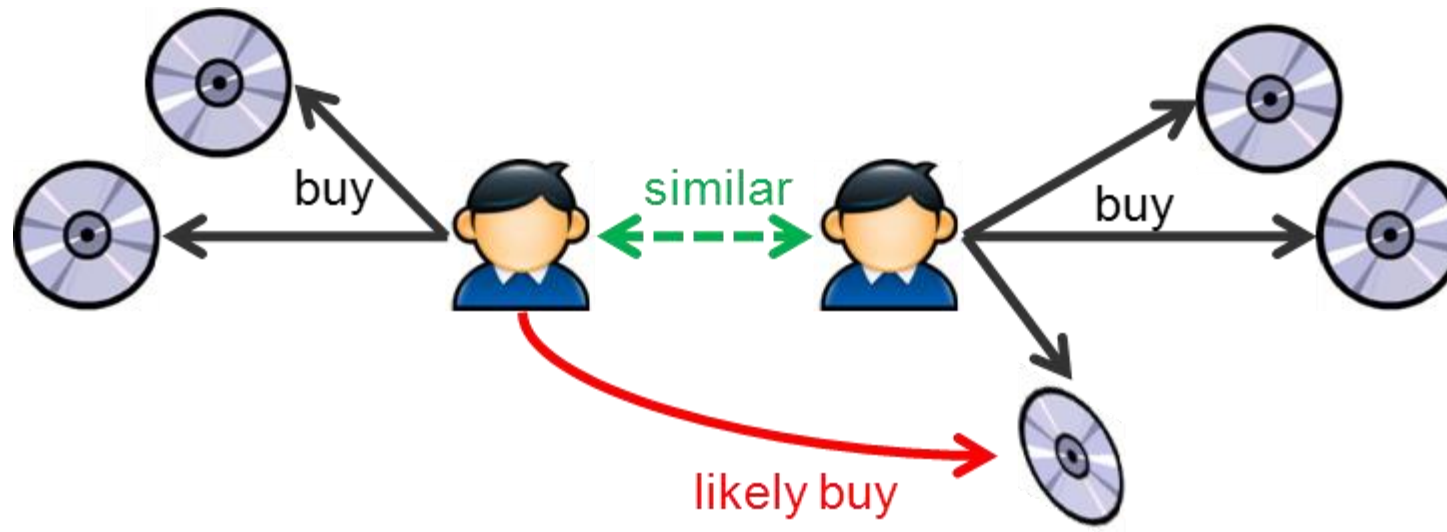
- User independence: collaborative filtering needs other users' rating to find the similarity between the users and then give the suggestion. Instead, content-based method only have to analyze the items and user profile for recommendation.
- No cold start: opposite to collaborative filtering, new items can be suggested before being rated by a substantial number of users.

CONS

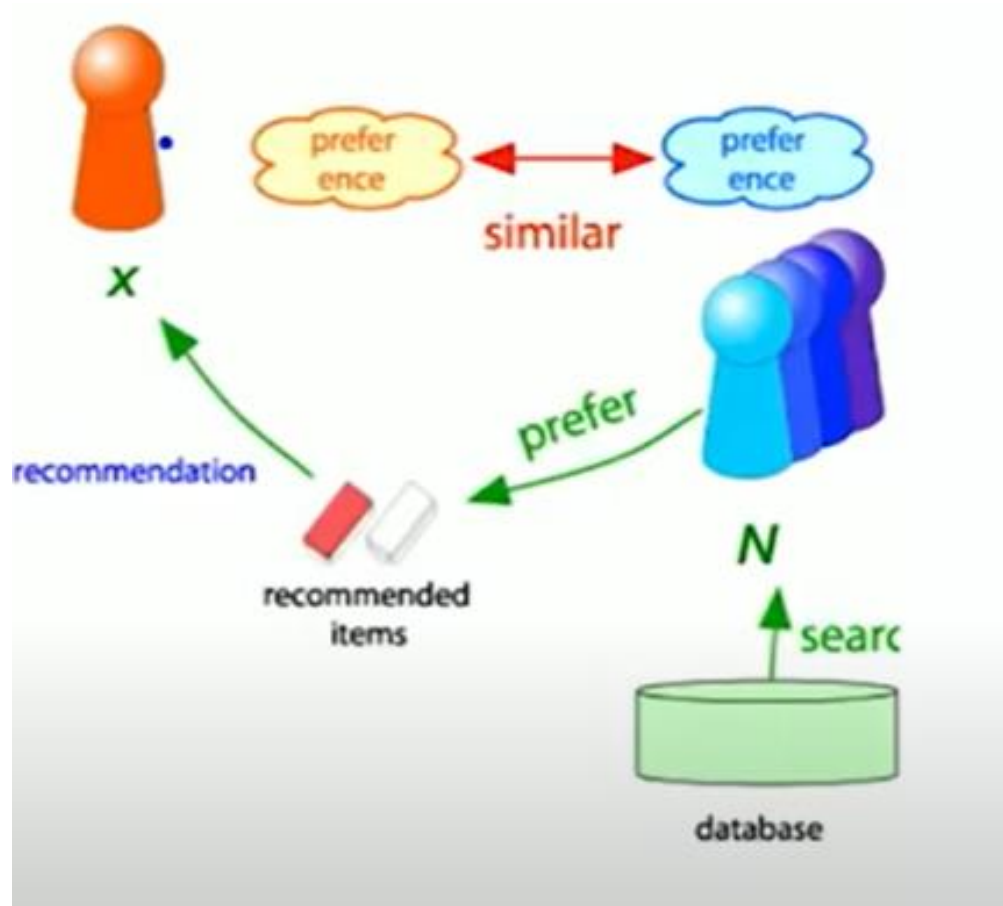
- They are not good at capturing inter-dependencies or complex behaviors. For example: I might like articles on Machine Learning, only when they include practical application along with the theory, and not just theory. This type of information cannot be captured by these recommenders.

COLLABORATIVE BASED RECOMMENDATIONS

- Collaborative filtering uses similarities between users and items simultaneously to provide recommendations.
- Collaborative filtering models can recommend an item to user A based on the interests of a similar user B.
- These recommender systems build a model from a user's past behavior, such as items purchased previously or ratings given to those items and similar decisions by other users.



COLLABORATIVE BASED RECOMMENDATIONS



The first question we must deal with is how to measure similarity of users or items from their rows or columns in the utility matrix.

SIMILAR USER

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Captured Intuition:
 $\text{sim}(A,B) > \text{sim}(A,C)$
.

- There are ways to find the similarity between the users or items.
 - ✓ Option 1: Jaccard Similarity.
 - ✓ Option 2: Cosine Similarity.
 - ✓ Option 3: Person based similarity.

JACCARD SIMILARITY

- It is defined as the fraction of number of common elements in the two sets to the total number of elements in the union of two sets.

$$\text{sim}(A,B) = |r_A \cap r_B| / |r_A \cup r_B|$$

- Ranges from 0-1 . Higher the similarity more the similar sets.
- $\text{Sim}(A,B) = 1/5$ and $\text{Sim}(A,C) = 2/4$ which means $\text{Sim}(A,C) > \text{Sim}(A,B)$.
- The problem with the Jaccard similarity is **Ignores the rating values!**.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

COSINE SIMILARITY

- Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space.

$$\text{sim}(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

The cosine of the angle between *A* and *B* is

$$\frac{4 \times 5}{\sqrt{4^2 + 5^2 + 1^2} \sqrt{5^2 + 5^2 + 4^2}} = 0.380$$

The cosine of the angle between *A* and *C* is

$$\frac{5 \times 2 + 1 \times 4}{\sqrt{4^2 + 5^2 + 1^2} \sqrt{2^2 + 4^2 + 5^2}} = 0.322$$

- A is slightly closer to B than C.
- The problem is it will treat the missing values as negative. This means User A didn't rate HP2. On a scale of 0-5 rating. 0 is the least rating.

CENTERED COSINE

- Normalize the ratings by subtracting the row mean.
- we turn low ratings into negative numbers and high ratings into positive numbers.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3



	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

The cosine of the angle between A and C is

$$\frac{(5/3) \times (-5/3) + (-7/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2} \sqrt{(-5/3)^2 + (1/3)^2 + (4/3)^2}} = -0.559$$


Let us compute the cosine of the angle between A and B :

$$\frac{(2/3) \times (1/3)}{\sqrt{(2/3)^2 + (5/3)^2 + (-7/3)^2} \sqrt{(1/3)^2 + (1/3)^2 + (-2/3)^2}} = 0.092$$


Both these observations make intuitive sense, given that A and C disagree on the two movies they rated in common, while A and B give similar scores to the one movie they rated in common

CASE STUDY

movies	users											
	1	2	3	4	5	6	7	8	9	10	11	12
	1		3			5			5		4	
	2			5	4			4		2	1	3
	3	2	4		1	2		3		4	3	5
	4		2	4		5			4		2	
	5			4	3	4	2				2	5
	6	1		3		3			2		4	

 - unknown rating  - rating between 1 to 5

movies	users											
	1	2	3	4	5	6	7	8	9	10	11	12
	1	1		3		5			5		4	
	2			5	4			4		2	1	3
	3	2	4		1	2		3		4	3	5
	4		2	4		5			4		2	
	5			4	3	4	2				2	5
	6	1		3		3			2		4	

 - estimate rating of movie 1 by user 5

CASE STUDY

- Calculate Pearson correlation as a similarity metrics.

		users												sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12	
movies	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

Here we use Pearson correlation as similarity:

CASE STUDY

- Neighbour selection: Chose $N = 2$.

	users												
	1	2	3	4	5	6	7	8	9	10	11	12	
1	1		3		?	5			5		4		sim(1,m) 1.00
2			5	4			4			2	1	3	-0.18
3	2	4		1	2		3		4	3	5		0.41
4		2	4		5			4			2		-0.10
5			4	3	4	2					2	5	-0.31
6	1		3		3			2			4		0.59

Compute similarity weights:
 $s_{13}=0.41, s_{16}=0.59$

Predict by taking weighted average:

$$r_{15} = (0.41*2 + 0.59*3) / (0.41+0.59) = 2.6$$

PROS AND CONS OF CONTENT BASED SYSTEMS

PROS

- **Personalization:** Collaborative filtering provides personalized recommendations to users based on their behavior and interactions with items. This makes the recommendations more relevant and useful to the user.
- **Diversity:** Collaborative filtering can recommend items that the user may not have discovered on their own. This can increase the diversity of recommended items and expose users to new and interesting content.

CONS

- **Cold start problem:** Collaborative filtering can have difficulty making recommendations for new users or items that have little to no user data. This is known as the "cold start" problem.