

AN IN-DEPTH GUIDE TO HOW RECOMMENDER SYSTEMS WORK

You might also like ... to know how programs know what you want before you do

Badreesh Shetty

July 24, 2019

Updated: September 4, 2019

Recommender systems are machine learning systems that help users discover new product and services. Every time you shop online, a recommendation system is guiding you towards the most likely product you might purchase.

Recommender systems are an essential feature in our digital world, as users are often overwhelmed by choice and need help finding what they're looking for. This leads to happier customers and, of course, more sales. Recommender systems are like salesmen who know, based on your history and preferences, what you like.

WHAT ARE RECOMMENDER SYSTEMS?

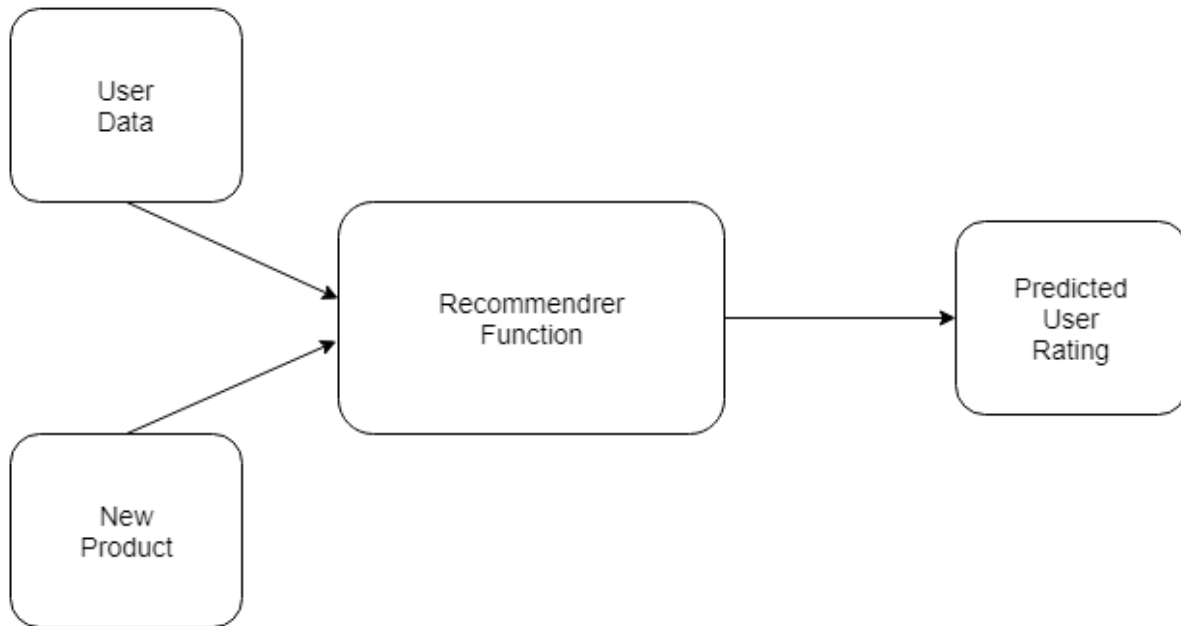
Recommender systems are so commonplace now that many of us use them without even knowing it. Because we can't possibly look through all the products or content on a website, a recommendation system plays an important role in helping us have a better user experience while also exposing us to more

[View jobs at top tech companies](#)

TECH JOBS

your feed, recommended videos on YouTube, music on Spotify, the Facebook newsfeed and Google Ads.

An important component of any of these systems is the recommender function, which takes information about the user and predicts the rating that user might assign to a product, for example. Predicting user ratings, even before the user has



Recommender Function

actually provided one, makes recommender systems a powerful tool.

HOW DO RECOMMENDER SYSTEMS WORK?

UNDERSTANDING RELATIONSHIPS

Relationships provide recommender systems with tremendous insight, as well as an understanding of customers. There are three main types that occur:

User-Product Relationship

The user-product relationship occurs when some users have an affinity or preference towards specific products that they need. For example, a cricket

[View jobs at top tech companies](#)

TECH JOBS

▲

Product-product relationships occur when items are similar in nature, either by appearance or description. Some examples include books or music of the same genre, dishes from the same cuisine, or news articles from a particular event.

User-User Relationship

User-user relationships occur when some customers have similar taste with respect to a particular product or service. Examples include mutual friends, similar backgrounds, similar age, etc.

DATA & RECOMMENDER SYSTEMS

In addition to relationships, recommender systems utilize the following kinds of data:

User Behavior Data

Users behavior data is useful information about the engagement of the user on the product. It can be collected from ratings, clicks and purchase history.

User Demographic Data

User demographic information is related to the user's personal information such as age, education, income and location.

Product Attribute Data

Product attribute data is information related to the product itself such as genre in case of books, cast in case of movies, cuisine in case of food.

HOW DO WE PROVIDE DATA FOR RECOMMENDER SYSTEMS?

[View jobs at top tech companies](#)

TECH JOBS

Explicit Ratings

Explicit ratings are provided by the user. They infer the user's preference.

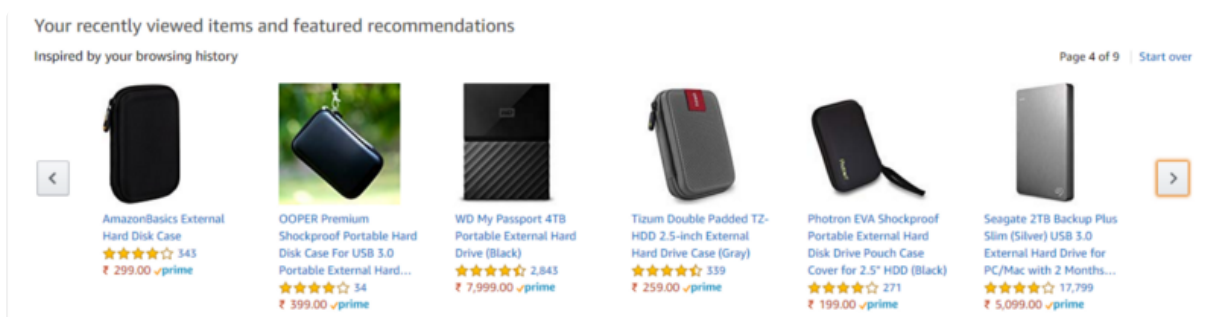
Examples include star ratings, reviews, feedback, likes and following. Since users don't always rate products, explicit ratings can be hard to get.

Implicit Ratings

Implicit ratings are provided when users interact with the item. They infer a user's behavior and are easy to get as users are subconsciously clicking. Examples include clicks, views and purchases. (Note: Views and purchases can be a better entity to recommend as users will have spent time and money on what is most crucial for them.)

Product Similarity (Item-Item Filtering)

Product similarity is the most useful system for suggesting products based on how much the user would like the product. If the user is browsing or searching for a particular product, they can be shown similar products. Users often expect to find products they want quickly and move on if they have a hard time finding the relevant product. When the user clicks on one product we can show another similar product, or if the user buys the product we can email the user advertisements or coupons based on a similar product. Product similarity is particularly useful when we don't know much about the user yet, but we do know what products they're viewing.



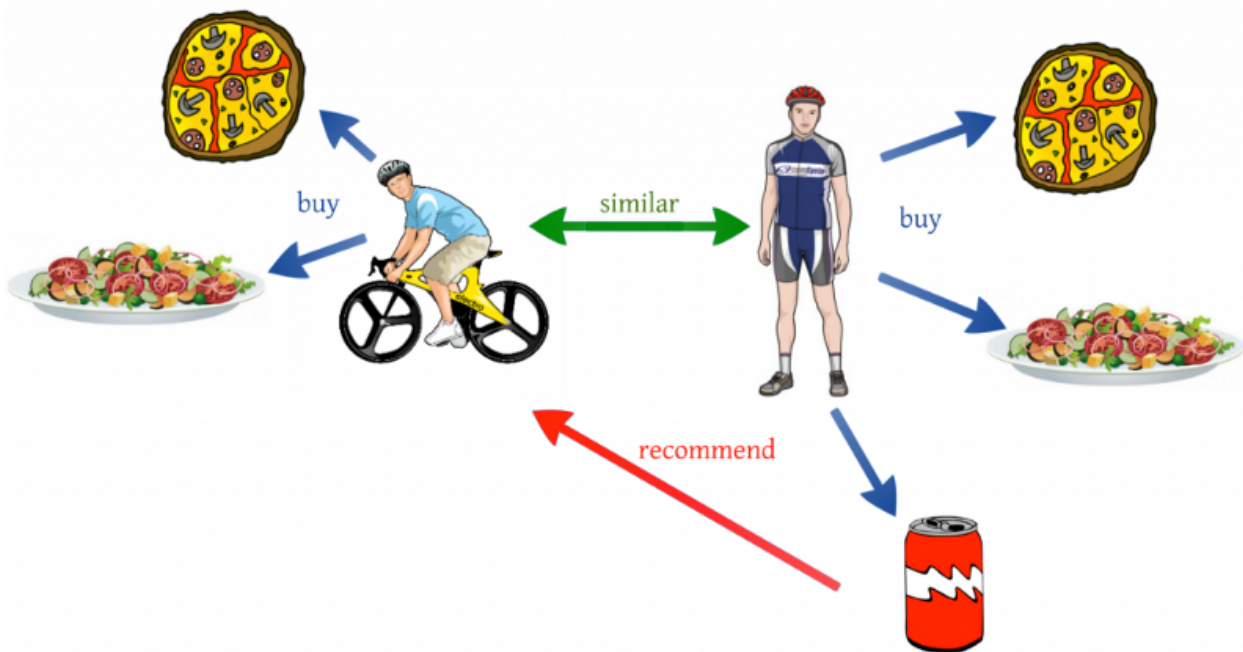
Amazon suggesting similar products.

User Similarity (User-User Filtering)

View jobs at top tech companies

TECH JOBS

One shortcoming of user similarity, however, is that it requires all the user data to suggest products. It's called a **cold start problem** because beginning the recommendation process requires previous data from users. A newly launched e-commerce website, for example, suffers from the cold start problem because it doesn't have a large number of users.



User Similarity

Customers who viewed this item also viewed

Page 1 of 7



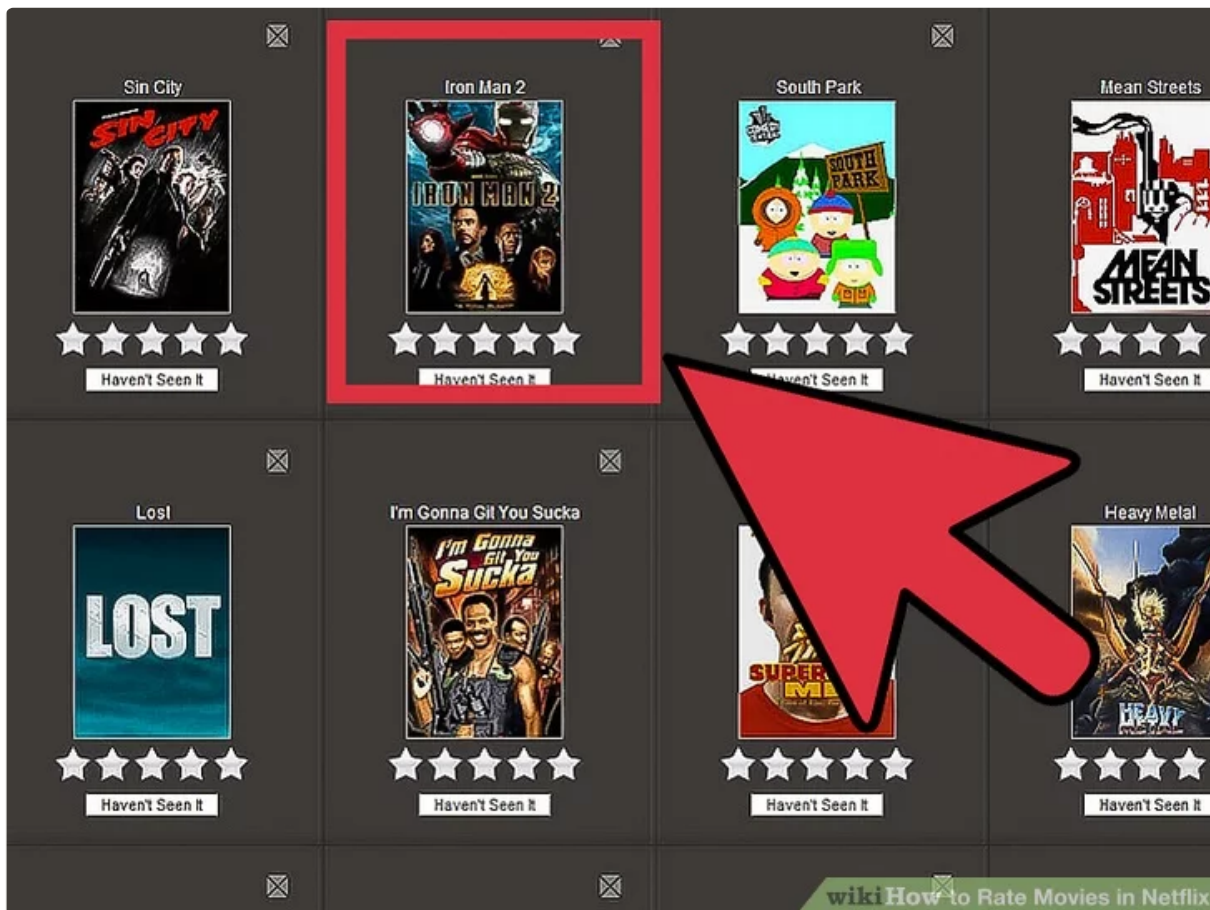
Product Association

Amazon Customer Similarity

Product similarity doesn't have this problem because it just requires product information and the user's preference. Netflix, for example, avoids this issue by asking users their likes when starting a new subscription.

View jobs at top tech companies

TECH JOBS

*Netflix It*

Similarity Measures

Similarity is measured using the distance metric. Nearest points are the most similar and farthest points are the least relevant. The similarity is subjective and is highly dependent on the domain and application. For example, two movies are similar because of genre or length or cast. Care should be taken when calculating distance across dimensions/features that are unrelated. The relative values of each element must be normalized, or one feature could end up dominating the distance calculation.

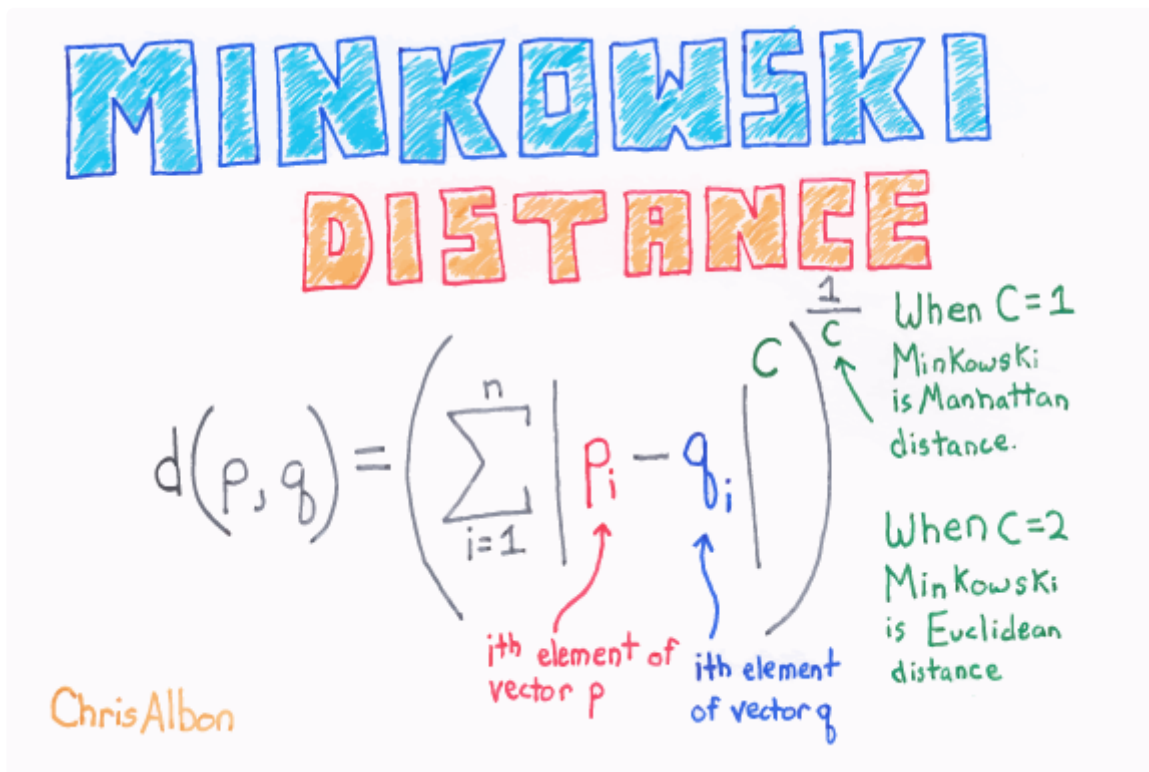
View jobs at top tech companies

TECH JOBS

$$d(x, y) = \left(\sum_i^n (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

It is
a

Minkowski Distance



Minkowski Distance

generic distance metric where Manhattan($r=1$) or Euclidean($r=2$) distance measures are generalizations of it.

Manhattan Distance: The distance between two points measured along axes at right angles.

$$d(x, y) = \sum_i^n |x_i - y_i|$$

Manhattan Distance

Euclidean Distance: The square root of the sum of squares of the difference between the coordinates and is given by Pythagorean theorem.

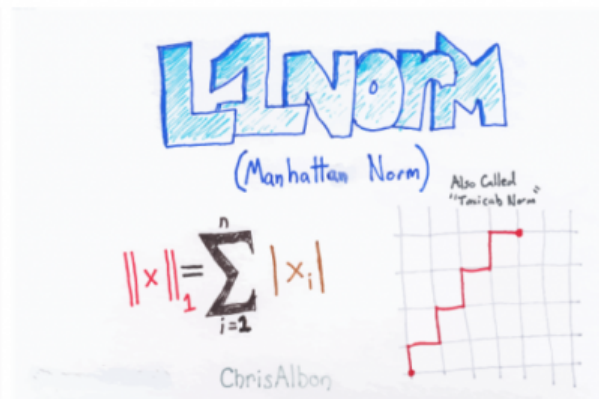
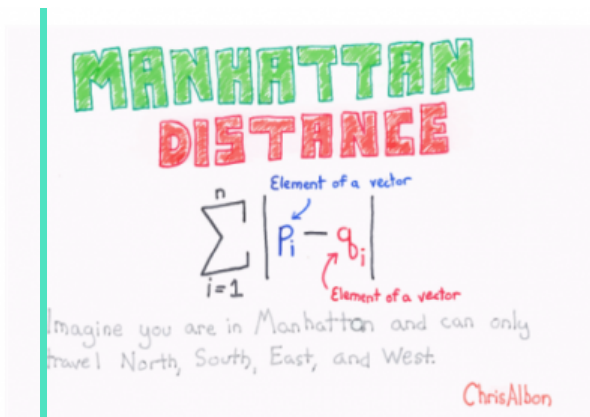
Cosine Similarity: Measures the cosine of the angle between two vectors. It is a judgment of orientation

View jobs at top tech companies

TECH JOBS

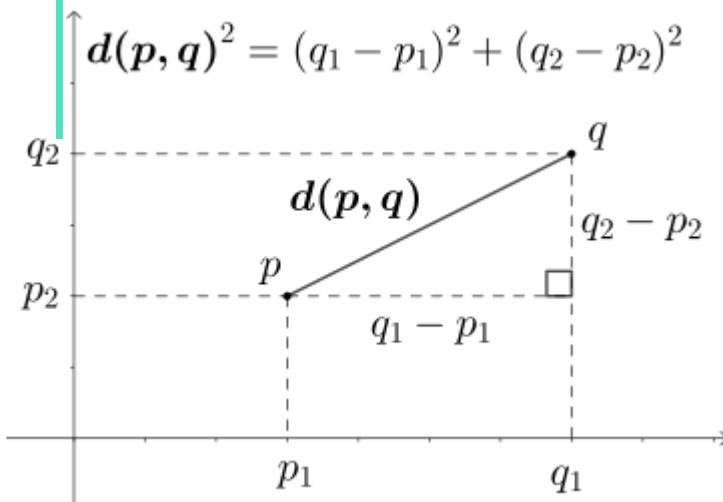
Pearson Coefficient: It is a measure of correlation between two random variables and ranges between $[-1, 1]$.

If the value is 1, it is a positive correlation, and if -1 then there is a negative correlation among variables.



Manhattan Distance

It is also called rectilinear distance, L1-distance/L1-norm, Minkowski's L1-distance, city block distance and taxi cab distance.



Euclidean Distance (General)

Correlation does not imply causation

Jaccard Similarity: In the other similarity metrics, we discussed some ways to find the similarity between objects, where the objects are points or vectors. We use Jaccard similarity to find similarities

between finite sets. It is defined as the cardinality of the intersection of sets divided by the

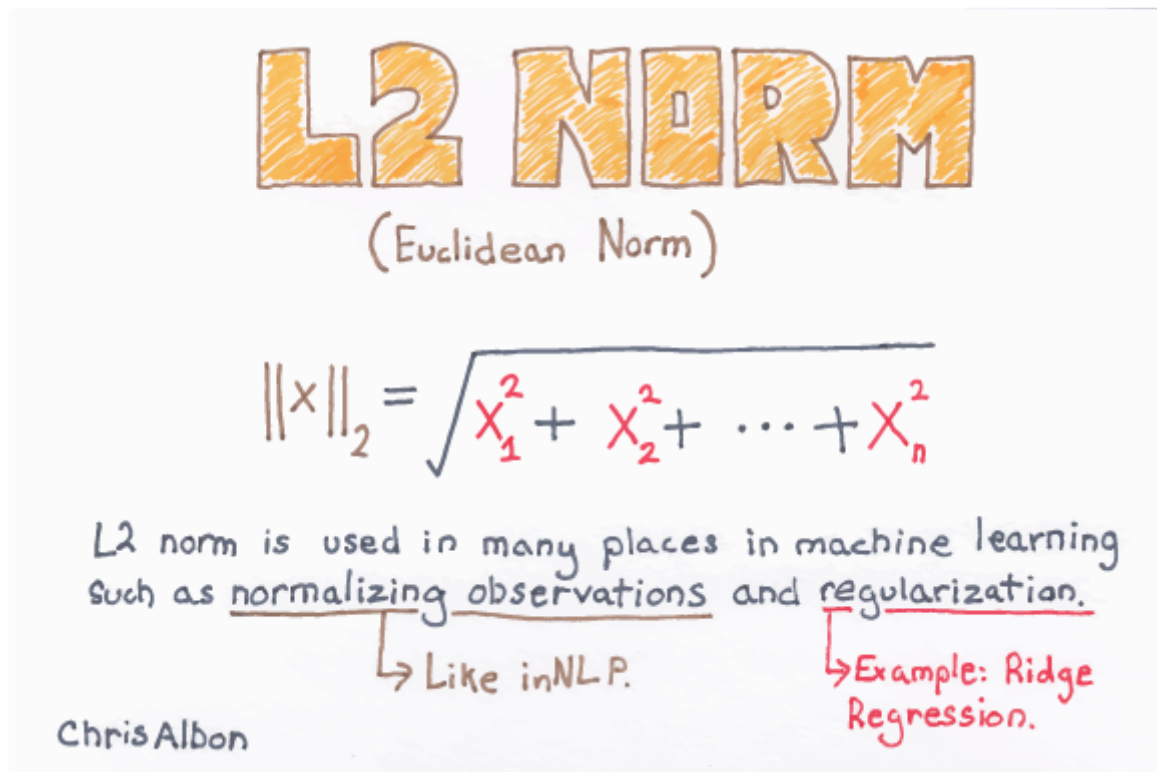
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

View jobs at top tech companies

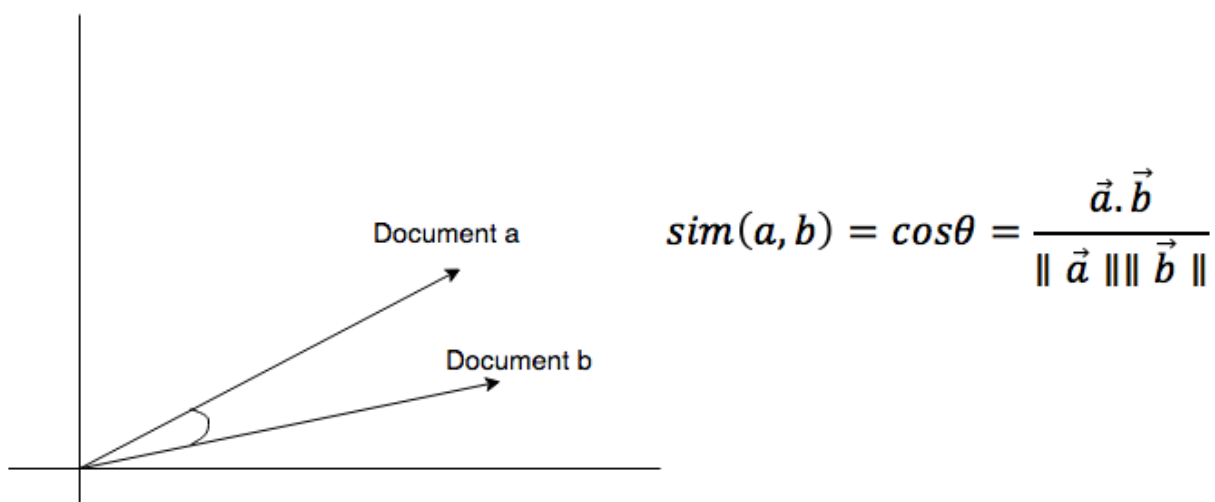
TECH JOBS

variables. In the case of categorical variables, Hamming distance must be used.

If the value (x) and the value (y) are the same, the distance D will be equal to 0, otherwise D=1. If we have data that is binary (i.e., classification), one would go for Hamming distance. The lower value means high similarity and higher value means



It is also called as L2 norm or ruler distance. In general, default distance is considered Euclidean distance.



View jobs at top tech companies

TECH JOBS

as similar with cosine similarity, whereas the Euclidean distance measure shows them as being far away from each other (i.e., they are dissimilar).

less similarity between variables. For example, the Hamming distance between 1101111 and 1001001 is 3, while the Hamming distance between 'batman' and 'antman' is 2

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

Pearson Correlation Coefficient

CORRELATION
(Pearson's R)

value of x_i mean of x value of y_i mean of y

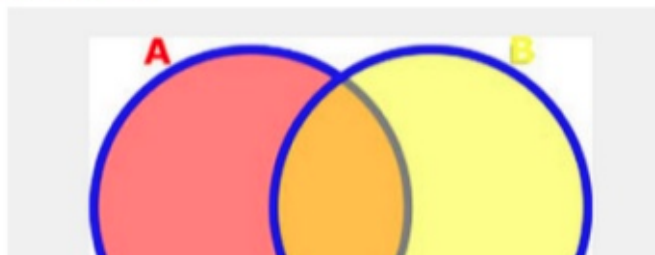
$$\text{Cor}(X, Y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2} \sqrt{\sum (y_i - \bar{y})^2}}$$

Ranges between -1.0 and +1.0. The closer to 0.0 the less linear dependence between variables.

BY CHRIS ALBON

Pearson Correlation Coefficient

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



View jobs at top tech companies

TECH JOBS

Jaccard Similarity

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

$$L_{\text{Hamming}} = \frac{\sum_{i=1}^n I(\hat{y}_i \neq y_i)}{n}$$

Chris Albon

Hamming Distance

APPROACHES TO CONTENT-BASED RECOMMENDER SYSTEMS

Content-based recommendation systems use their knowledge about each product to recommend new ones. Recommendations are based on attributes of the item. Content-based recommender systems work well when descriptive data on the content is provided beforehand. “Similarity” is measured against product attributes.

Suppose I watch a movie in a particular genre, then I will be recommended movies within that specific genre. The movie's attributes, like title, year of release, director and cast, are also helpful in identifying similar movie content.

APPROACH 1: USING RATED CONTENT TO RECOMMEND

In this approach contents of the product are already rated and based on the user's preference, then a rating is predicted for a similar product. We'll use movie

View jobs at top tech companies

TECH JOBS

User\Movies	Interstellar	Inception	The Shining	Alien
Clark	5 stars			
Bruce			5 stars	
Tony				
Steve		2 stars		1 star

In the example above, Clark and Bruce have given five-star ratings to the movies "Interstellar" and "The Shining," clearly indicating a preference for these films. For Tony, who has rated nothing, and Steve who has provided only low ratings, it's more difficult to discern their preferences.

Movie Attributes

Movie\Attributes	Science	Adventure	Horror	Mystery
Interstellar	5	4	1	2
Inception	5	3	1	2
The Shining	1	2	5	4
Alien	2	3	5	3

In the table above, note that "Interstellar" and "Inception" received 5s in the science category, whereas "The Shining" and "Alien" get the highest marks under the horror genre.

Predicted User Rating

[View jobs at top tech companies](#)

TECH JOBS

Bruce			5 stars	5 stars
Tony				
Steve		2 stars		1 star

"Inception" is suggested for Clark because he liked "Interstellar" and the movies share similar attributes. "Alien" is suggested to Bruce because he liked "The Shining," which is in the horror genre.

Advantages: Works even when a product has no user reviews.

Disadvantages: Requires descriptive data of all content to recommend, which is time consuming. It's also difficult to implement on large product databases as user's have different opinions about each item.

APPROACH 2: RECOMMENDATION THROUGH DESCRIPTION OF THE CONTENT

This approach uses the description of the item to make recommendations. The description goes deeper into the product details, like title, summary, tag lines, genre, etc., and it provides much more information about the item. The format of these details are in text format(string) and it's important to convert this into numbers to easily calculate for similarity.

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is used in information retrieval for feature extraction purposes and it is a sub-area of natural language processing (NLP).

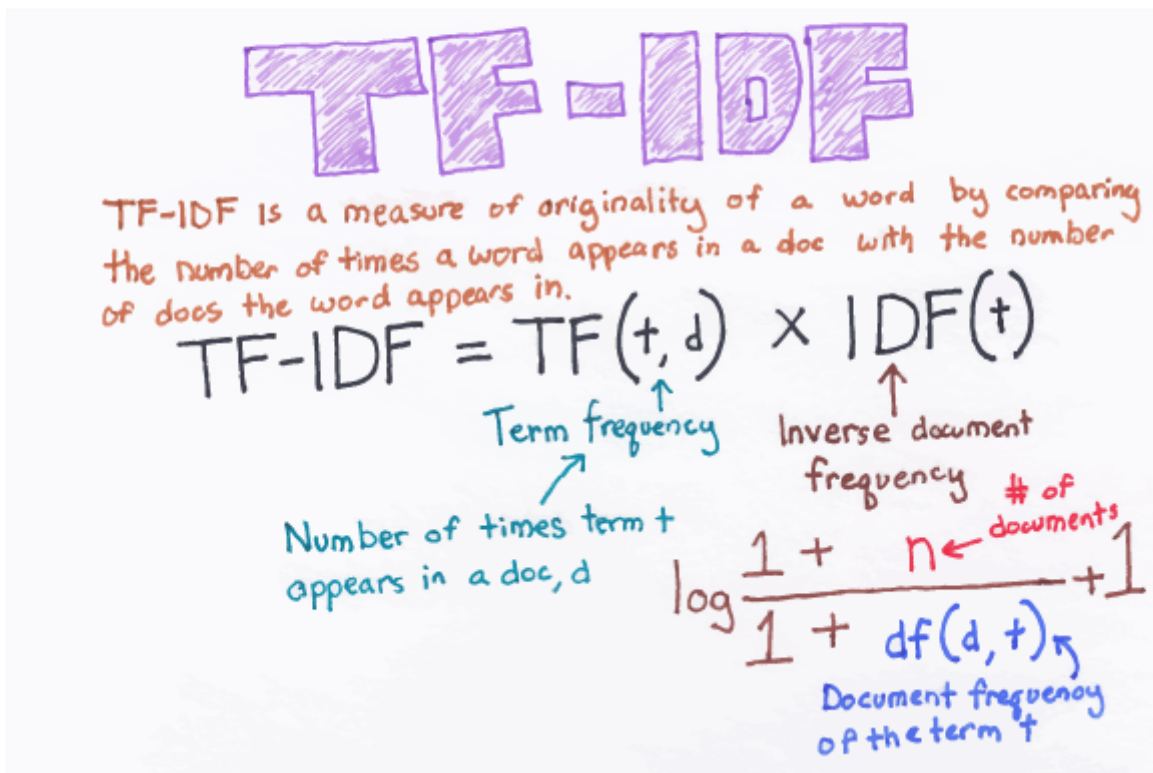
Term Frequency: Frequency of the word in the current document to the total number of words in the document. It signifies the occurrence of the word in a document and gives higher weight when the frequency is more, so it is divided by document length to normalize.

[View jobs at top tech companies](#)

TECH JOBS

Frequency: Total number of documents to the frequency occurrence of documents containing the word. It signifies the rarity of the word – the less the word occurs in the document, the IDF increases. It helps in giving a higher score to rare terms in the documents.

$$\text{Idf}(t) = \log_{10}\left(\frac{\text{Total Number of documents}}{\text{Number of documents containing term } t}\right)$$



TF-IDF

In the end, TF-IDF is a measure used to evaluate how important a word is to a document in a document corpus. The importance of the word increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Example for TF-IDF

[View jobs at top tech companies](#)

TECH JOBS

inverse document frequency is calculated as $\log(10,000,000 / 1,000) = 3$. Thus, the TF-IDF weight for Odyssey is $0.03 * 3 = 0.09$.

COLLABORATIVE FILTERING RECOMMENDER

Collaborative filtering recommender makes suggestions based on how users rated in the past and not based on the product themselves. It only knows how other customers rated the product. “Similarity” is measured against the similarity of users.



Collaborative Filtering Recommender

Going back to our movie example earlier, we can illustrate this technique.

USER RATING

User\Movies	Interstellar	Inception	The Shining	Alien
Clark	5 stars	2 stars	1 star	
Bruce	1 star	5 stars		1 star
Tony	5 star	3 stars	1 star	5 stars

View jobs at top tech companies

TECH JOBS

movies similarly.

PREDICTED USER RATING

Clark is recommended "Alien" from Tony because of their similarity in rating. Bruce was suggested "The Shining" because Steve rated it highly.

User\Movies	Interstellar	Inception	The Shining	Alien
Clark	5 stars	2 stars	1 star	5 stars
Bruce	1 star	5 stars	5 stars	1 star
Tony	5 stars	3 stars	1 star	5 stars
Steve	1 star	5 stars	5 stars	1 star

Advantages:

No requirement for product descriptions.

Disadvantages:

-
- Can't recommend item if no user reviews exist (suffers from the cold start problem).
 - Difficult to recommend new users and is inclined to favor popular products with lots of reviews.
 - Suffers from a sparsity problem as user will review only selected items.
 - Faces the "gray sheep problem" (i.e., useful predictions cannot be made due to sparsity).
 - Difficult to recommend new releases since they have less reviews.

SINGULAR VALUE DECOMPOSITION (SVD)

Most collaborative recommender systems perform poorly when dimensions in the user-item matrix are too high. This is because the matrix is too sparse to be useful.

[View jobs at top tech companies](#)

TECH JOBS

(latent factors) and remove redundant parts. There are many dimensionality reduction algorithms such as principal component analysis (PCA) and linear discriminant analysis (LDA), but SVD is used mostly in the case of recommender systems. SVD uses matrix factorization to decompose matrix.

$$A = U \Sigma V^T$$

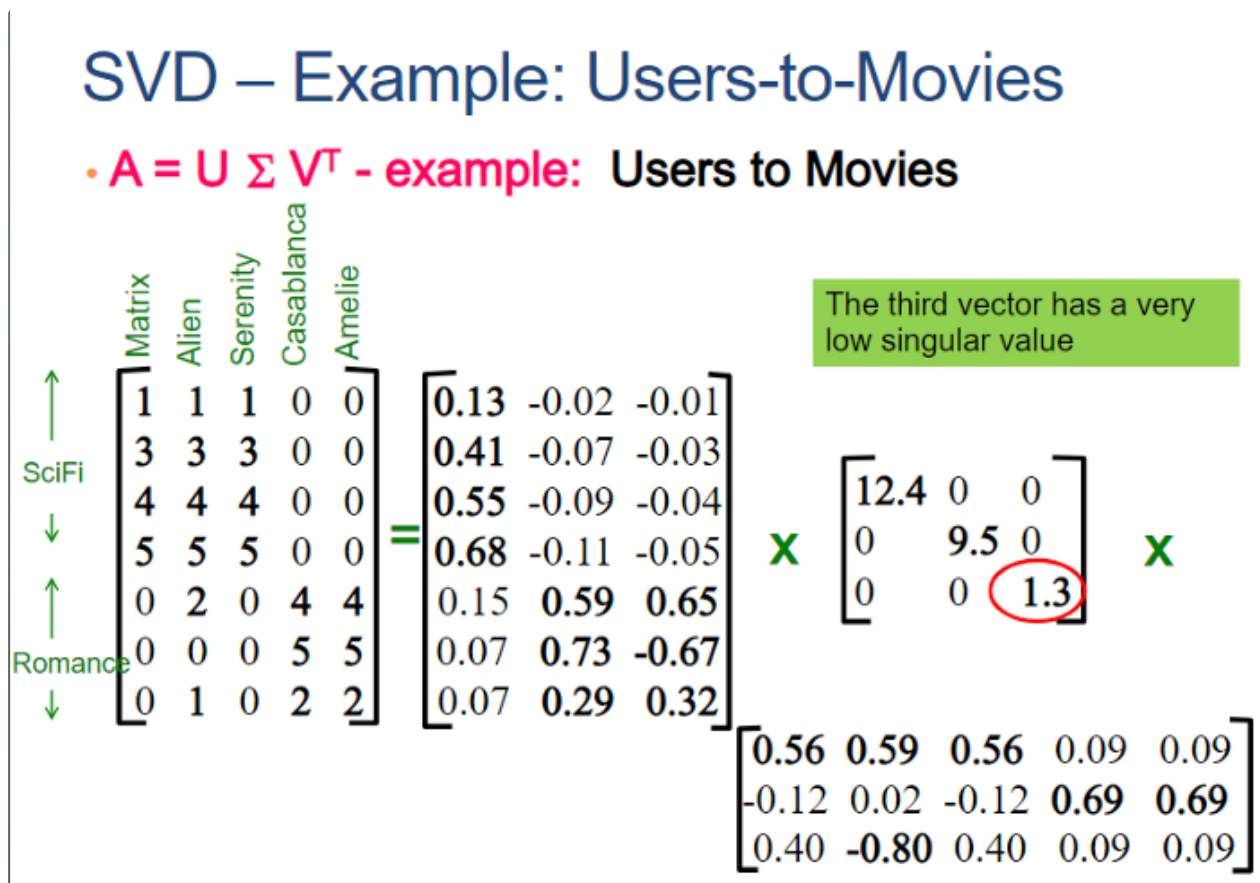
In the case above, A matrix($m \times n$) can be decomposed into U($m \times m$) orthogonal matrix, Σ ($m \times n$) non-negative diagonal matrix, and V ($n \times n$) orthogonal matrix

$$U^T U = V^T V = I$$

U is also referred to as the left singular vectors, Σ as singular values, and V as right singular vectors

Orthogonal Matrix

Let's take the example of movies rated by users in matrix A:



[View jobs at top tech companies](#)

TECH JOBS

highly. While the second column of U represents weights that match each user's preference to the romance category. We can see that the sixth user rates the romance category highly. For the third column, we need to look at the Σ value given.

For Σ , the first diagonal entry represents the weight (strength) of the sci-fi category (12.4) and the second diagonal entry represents the weight (strength) of the romance category (9.5). And as for the third diagonal entry, it is considered noise because the value is lower compared to other diagonal entries.

For V , the columns show the similarity of movies to a category. So, we can see from the first column of V that the first movie belongs to the sci-fi (0.56) category. Similarly, since the third row is discarded as noise the second movie also belongs to the sci-fi category.

After eliminating the third column from U , third row and third column from Σ , and third row, we get the below matrices.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & -0.02 \\ 0.41 & -0.07 \\ 0.55 & -0.09 \\ 0.68 & -0.11 \\ 0.15 & 0.59 \\ 0.07 & 0.73 \\ 0.07 & 0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ -0.12 & 0.02 & -0.12 & 0.69 & 0.69 \end{bmatrix}$$

We have reduced the initial $U\Sigma V$ dimension, so to check if we've lost any information we need to multiply all three matrices.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \end{bmatrix}$$

[View jobs at top tech companies](#)

TECH JOBS

$$\begin{bmatrix} 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \quad \begin{bmatrix} 0.70 & \mathbf{0.53} & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & \mathbf{0.23} & 0.32 & 2.01 & 2.01 \end{bmatrix}$$

After multiplying we see there is very little loss of information as the elements of a matrix are very similar. It is similarly used for predicting missing values among ratings and suggesting high ratings to the user.

So this is how we decompose a matrix without losing much of the important data and It helps to analyze and acquire important information concerning the matrix data.

References: [Data Mining Lecture 7: Dimensionality Reduction PCA–SVD](#)

HYBRID RECOMMENDER

Hybrid recommender is a recommender that leverages both content and collaborative data for suggestions. In a system, first the content recommender takes place as no user data is present, then after using the system the user preferences with similar users are established.

For example, Netflix deploys hybrid recommender on a large scale. When a new user subscribes to their service they are required to rate content already seen or rate particular genres. Once the user begins using the service, collaborative filtering is used and similar content is suggested to the customer.

ASSOCIATION RULES LEARNING

Association rules learning is used for recommending complementary products. It helps in associating one product with another product and tries to answer which products are associated with one another. It is mostly used in e-commerce as

[View jobs at top tech companies](#)

TECH JOBS

Code:

Content filtering:

Basic Content-Based Filtering Implementation

Frequently bought together



Total price: ₹24,939.00

Add both to Cart

! These items are dispatched from and sold by different sellers. [Show details](#)

✓ **This item:** Samsung Galaxy A8+ (Black, 6GB RAM, 64GB Storage) with Offers ₹23,990.00

✓ Spigen Liquid Air Case for Samsung Galaxy A8 Plus (2018) - Matte Black 591CS22757 ₹949.00

Amazon's Associative Recommendation

Importing Libraries and Loading Data

```
In [1]: 1 import pandas as pd
        2 import numpy as np
        3 import matplotlib.pyplot as plt

In [2]: 1 # Reading movies file
        2 movies = pd.read_csv('movies.csv', sep=',', encoding='latin-1', usecols=['title', 'genres'])

In [3]: 1 movies.head()

Out[3]:
```

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

Importing the MovieLens dataset and using only title and genres column

```
In [4]: 1 # Break up the big genre string into a string array
        2 movies['genres'] = movies['genres'].str.split('|')
        3 # Convert genres to string value
        4 movies['genres'] = movies['genres'].fillna('').astype('str')
```

Splitting the different genres and converting the values as string type.

View jobs at top tech companies

TECH JOBS

recommendations based on similar genres and having high cosine similarity.

Based on the genre of "The Dark Knight" (i.e., action, crime, drama, IMAX), closely similar genres are recommended.

Using TfidfVectorizer to convert titles in 2-gram words excluding stopwords,

Recommendation based on Genre

```
In [5]: 1 from sklearn.feature_extraction.text import TfidfVectorizer
2 tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
3 tfidf_matrix = tf.fit_transform(movies['genres'])
4 tfidf_matrix.shape
```

Out[5]: (9742, 177)

```
In [6]: 1 from sklearn.metrics.pairwise import cosine_similarity
2 cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
3 cosine_sim[:4, :4]
```

```
Out[6]: array([[1.          , 0.31379419, 0.0611029 , 0.05271111],
 [0.31379419, 1.          , 0.          , 0.          ],
 [0.0611029 , 0.          , 1.          , 0.35172407],
 [0.05271111, 0.          , 0.35172407, 1.          ]])
```

```
In [7]: 1 # Build a 1-dimensional array with movie titles
2 titles = movies['title']
3 indices = pd.Series(movies.index, index=movies['title'])
4
5 # Function that get movie recommendations based on the cosine similarity score of movie genres
6 def genre_recommendations(title):
7     idx = indices[title]
8     sim_scores = list(enumerate(cosine_sim[idx]))
9     sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
10    sim_scores = sim_scores[1:21]
11    movie_indices = [i[0] for i in sim_scores]
12    return titles.iloc[movie_indices]
```

```
In [8]: 1 genre_recommendations('Dark Knight ').head(20)
```

```
Out[8]: 8387          Need for Speed
8149      Grandmaster, The (Yi dai zong shi)
123          Apollo 13
8026          Life of Pi
8396          Noah
38          Dead Presidents
341          Bad Company
347      Faster Pussycat! Kill! Kill!
430          Menace II Society
568          Substitute, The
665          Nothing to Lose
1645      Untouchables, The
1696          Monument Ave.
2563          Death Wish
2574      Band of the Hand
3037          Foxy Brown
3124      Harley Davidson and the Marlboro Man
3167          Scarface
3217          Swordfish
```

cosine similarity is taken between matrix which is transformed. Generating recommendations based on similar genres and having high cosine similarity.

[Basic Collaborative Filtering Implementation](#)

Importing dictionaries

[View jobs at top tech companies](#)

TECH JOBS

found in the reviews of

each user and based on common movies the reviews are found on each movie.

Euclidean distance is calculated between the common movies between users. The similarity is indirectly proportional to distance. As the distance increases, similarity decreases and vice versa. Critic similarity is used to the fine similarity of

Recommendation based on Title

```
In [9]: 1 from sklearn.feature_extraction.text import TfidfVectorizer
2 tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
3 tfidf_matrix = tf.fit_transform(movies['title'])
4 tfidf_matrix.shape
```

Out[9]: (9742, 20558)

```
In [10]: 1 from sklearn.metrics.pairwise import cosine_similarity
2 cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
3 cosine_sim[:4, :4]
```

Out[10]: array([[1., 0., 0., 0.],
[0., 1., 0., 0.],
[0., 0., 1., 0.],
[0., 0., 0., 1.]])

```
In [11]: 1 # Build a 1-dimensional array with movie titles
2 titles = movies['title']
3 indices = pd.Series(movies.index, index=movies['title'])
4
5 # Function that get movie recommendations based on the cosine similarity score of movie genres
6 def title_recommendations(title):
7     idx = indices[title]
8     sim_scores = list(enumerate(cosine_sim[idx]))
9     sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
10    sim_scores = sim_scores[1:21]
11    movie_indices = [i[0] for i in sim_scores]
12    return titles.iloc[movie_indices]
```

```
In [12]: 1 title_recommendations('Dark Knight ').head(20)
```

```
Out[12]: 7768          Dark Knight Rises, The
8032    Batman: The Dark Knight Returns, Part 1
8080    Batman: The Dark Knight Returns, Part 2
140          First Knight
2417          Cry in the Dark, A
5778          Alone in the Dark
7375          Knight and Day
3576          Black Knight
3190          Knight's Tale, A
6858          Alone in the Dark II
4242          Dark Blue
5060          Dark Days
1305          Dark City
5483          Dark Star
6815    Batman: Gotham Knight
5934          Dark Water
4749          Shot in the Dark, A
7877          Dark Shadows
8766          The Dark Valley
```

users.

From above, we can see Martin Scorsese and Joe Pesci are very similar to each other as they have rated common movies very closely.

At first, all similarity scores are found out wrt to critics

[View jobs at top tech companies](#)

TECH JOBS

weight w.r.t each movie.

The overall weight is

normalized to make suggestions. Then recommended movies are sorted by weight.

Importing Libraries and Loading Data

```
In [1]: 1 import math
        2 import operator
        3
        4 #Building Custom Data for Movie Rating
        5 review = {
        6     'Marlon Brando': {
        7         'The Godfather': 5.00,
        8         'The Godfather Part II': 4.29,
        9         'Apocalypse Now': 5.00,
        10        'Jaws': 1.
        11    },
        12    'Stephen King': {
        13        'The Shawshank Redemption': 4.89,
        14        'The Shining': 4.93 ,
        15        'The Green Mile': 4.87,
        16        'The Godfather': 1.33,
        17    },
        18    'Steven Spielberg': {
        19        'Raiders of the Lost Ark': 5.0,
        20        'Jaws': 4.89,
        21        'Saving Private Ryan': 4.78,
        22        'Star Wars Episode IV - A New Hope': 4.33,
        23        'Close Encounters of the Third Kind': 4.77,
        24        'The Godfather': 1.25,
        25        'The Godfather Part II': 1.72
        26    },
```

We can see from above Marlon Brando has rated

```
In [2]: 1 # Function to get common movies b/w Users
        2 def get_common_movies(criticA,criticB):
        3     return [movie for movie in review[criticA] if movie in review[criticB]]
```

```
In [3]: 1 get_common_movies('Marlon Brando','Robert DeNiro')
```

```
Out[3]: ['The Godfather', 'The Godfather Part II']
```

```
In [4]: 1 get_common_movies('Steven Spielberg','Tom Hanks')
```

```
Out[4]: ['Saving Private Ryan', 'The Godfather', 'The Godfather Part II']
```

```
In [5]: 1 get_common_movies('Martin Scorsese','Joe Pesci')
```

```
Out[5]: ['Raging Bull', 'Goodfellas', 'The Godfather']
```

```
In [6]: 1 # Function to get reviews from the common movies
        2 def get_reviews(criticA,criticB):
        3     common_movies = get_common_movies(criticA,criticB)
        4     return [(review[criticA][movie], review[criticB][movie]) for movie in common_movies]
```

View jobs at top tech companies

TECH JOBS

```
In [9]: 1 get_reviews('Martin Scorsese','Joe Pesci')
Out[9]: [(5.0, 4.89), (4.87, 5.0), (4.0, 4.87)]
```

"Godfather" high, "Goodfellas" is also rated highly where "Godfather" is rated highly, thus "Goodfellas" is recommended to Marlon Brando.

Euclidean Distance Formula for Calculating similarity

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_1)^2 + (x_n - y_n)^2}$$

```
In [10]: 1 # Function to get Euclidean Distance b/w 2 points
2 def euclidean_distance(points):
3     squared_diffs = [(point[0] - point[1]) ** 2 for point in points]
4     summed_squared_diffs = sum(squared_diffs)
5     distance = math.sqrt(summed_squared_diffs)
6     return distance

In [11]: 1 # Function to calculate similarity more similar less the distance and vice versa
2 # Added 1 for if highly similar can make the distance zero and give NotDefined Error
3 def similarity(reviews):
4     return 1/ (1 + euclidean_distance(reviews))

In [12]: 1 # Function to get similarity b/w 2 users
2 def get_critic_similarity(criticA, criticB):
3     reviews = get_reviews(criticA,criticB)
4     return similarity(reviews)
```

This is a very easy implementation of collaborative filtering, just the crux the similarity between users is implemented. Whereas industry uses matrix factorization, autoencoders and deep learning.

Find the above code in this [Github Repo](#).

References: Flashcards of Chris Albon

from: <https://machinelearningflashcards.com/>

SIX EXAMPLES OF RECOMMENDER SYSTEMS

Recommender systems work behind the scenes on many of the world's most popular websites. E-commerce websites, for example, often use recommender systems to increase user engagement and drive purchases, but suggestions

View jobs at top tech companies

TECH JOBS

When we buy something or browse anything on Amazon, we see recommended products based on our taste or search results on the page.

Amazon Recommends As we can see above, I may have searched for products related to Sudha Murthy, but Amazon clusters it into segments based on other previous search results. A recommender system often biases the user's opinion.

2. IMDB

When we rate a TV show or movie on IMDb it recommends other shows or movies based on important details like cast, genre, sub-genre, plot and summary.

3. FACEBOOK & INSTAGRAM

IMDb Recommend's

As we can see above, I was recommended to rate *Frozen Planet* because I've watched David Attenborough's wildlife documentary series. In this case, IMDb suggested this to me based on the cast of the series.

Facebook and Instagram use recommender systems on a wider scale for suggesting friends and stories in the newsfeed.

Facebook Recommend's Often in the newsfeed section, the user is bombarded with articles similar to one another based on the likes and pages they follow. Some argue it creates an unconscious bias among the user, which may be bad or useful depending on how the user interprets something when they have only one side of the argument.

PS: I seldom use Facebook.

4. YOUTUBE

YouTube recommends videos based on previous viewing or clicks.

View jobs at top tech companies

TECH JOBS

5. GOOGLE

Whenever we are logged in and search for anything on Google, it's registered in our history and recommends products based on previous searches. Not surprisingly, more than 90% of the company's revenue is generated through advertising. Relevant searches also take into account the location of the user.

Google Recommend's When clicking on the sponsored button, Google shows us why these ads are targeted.

Google Ads From the above we can see we were targeted with these ads based on recent search results.

PS: I use ublock origin but for this article, I unblocked Google to track my activities.

6. ANOTHER GOOGLE SUBSIDIARY- GMAIL

Gmail's Sponsored Ad Gmail recommends advertisements based on my inbox and the newsletters I've signed up for and the tags they may have generated.

Other examples of recommender systems at work include movies on Netflix, songs on Spotify and profiles on Tinder.

PRIVACY VS PERSONALIZED CURATION

In George Orwell's classic novel 1984, "Big Brother" is a government that scrutinizes each and every move of its citizenry. In today's world, tech companies like Google and Facebook more accurately fill that role. Each company has immense troves of data about millions of user's and they harvest it for ad targeting and building things like recommender systems.

Most large tech companies offer their services for free. so you are the product.

[View jobs at top tech companies](#)

TECH JOBS

In the end, less is more, and if consumers feel like a company knows too much about them, they probably do.

Personalized content or services can be an addiction or menace depending on your point of view. It can help us find things we like, but it can also create an unconscious bias among users. Personalization helps us have a better customer user experience, but it's a tradeoff and we need to find a sweet spot between privacy and personalization .

I'll end with a couple of important quotes:

“Data is the new oil. It’s valuable, but if unrefined it cannot really be used. It has to be changed into gas, plastic, chemicals, etc to create a valuable entity that drives profitable activity; so must data be broken down, analyzed for it to have value.” — Clive Humby, UK mathematician and architect of Tesco’s Clubcard, [2006](#)

“The difference between oil and data is that the product of oil does not generate more oil (unfortunately), whereas the product of data (self-driving cars, drones, wearables, etc) will generate more data (where do you normally drive, how fast/well you drive, who is with you, etc).” — Piero Scaruffi, cognitive scientist and author of [“History of Silicon Valley”](#), [2016](#)

Check: [Bias, privacy, and personalization on the web](#), [Cambridge Analytica and Online Manipulation](#), [GDPR for Privacy](#).

View jobs at top tech companies

TECH JOBS

An In-Depth Guide to Supervised Machine Learning Classification

Curse of Dimensionality

RELATED

[Read More About Data Science](#)

Data Science

Great Companies Need Great People. **That's Where We Come In.**

RECRUIT WITH US



© Built In 2019

GET INVOLVED

Recruit With Built In

Become an Expert Contributor

Contact Us

Careers

View jobs at top tech companies

TECH JOBS

OUR COMMUNITIES

[Built in Austin](#)

[Built in Boston](#)

[Built in Chicago](#)

[Built in Colorado](#)

[Built in LA](#)

[Built in NYC](#)

[Built in Seattle](#)

[Built In San Francisco](#)

ABOUT

[Our Story](#)

[Our Staff Writers](#)

[Privacy Policy](#)

[Terms of Use](#)

[Copyright Policy](#)

[Do Not Sell My Personal Info](#)

[CA Notice of Collection](#)

Live your purpose.

Work your passion.

