Recommendation Systems made simple!

Chhavi Saluja · Follow

6 min read · Feb 12, 2018

Why Recommendation Systems?

With increasing number of transactions happening over internet and with tremendous number of choices available to users, there is a need to sift and analyze

relevant information on users and items. Recommendation systems exploit user preferences and traits to prioritize and recommend items which the users would like.

Recommendations systems are a big value-add for large companies like Google, Amazon, Facebook, Netflix etc. as they drive significant customer engagement and revenue. Analysts estimate that already 35% of what consumers purchase on Amazon and 75%of what they watch on Netflix come from product recommendations based on recommendation algorithms. Recommendation systems not only exploit users by tempting them to buy more products & services customized to their tastes, but also keep them engaged for a longer time to show them more ads and get more clients.

What are Recommendation Systems?

Recommendation systems are also known as recommender systems or recommender engine. There are numerous definitions available on internet. However, the one I found most appealing is from this post.

"Recommendation systems are systems that help users discover items they may like"

Recommendation systems have multitude of applications like product recommendations (like movies and videos, books, fashion items etc.), recommendations of news articles, job postings, Facebook friends and what not.

Types of Recommendation Systems

Popularity-based

These simply recommend the most popular items to users. Popularity-based systems are simplest of all and have minimal computational requirements. However, as these systems do not make personalized recommendations based on specific user's likes & behaviors, they tend to be less accurate than content-based or collaborative filtering based systems.

Content-based/ Context-based

Content-based systems exploit user and item **attributes/ features** to recommend items that match user profiles. These systems are suitable in applications where

contextual information is easy to get. Also, unlike collaborative filtering, these do not depend on user feedback or ratings.

Example- Lets say we have two attributes named *Romance* and *Action* to feature movies. Users can also be profiled based on preferences to these features. Hence, User-feature and Movie-feature matrices have been created as given below (I've used some random figures here).

User-Feature Matrix	Romance	Action
User A	0.9	0.1
User B	0.8	0.2
User C	0.3	0.7
User D	0.6	0.4
User E	0	1

Movie-Feature Matrix	Romance	Action
Titanic	0.9	0.1
Rocky	0.1	0.9
The Hobbit	0.5	0.5
Fight Club	0	1
Jurassic Park	0.2	0.8

Content-based example

Now to predict the ratings that users will give to each movie, a dot product of user vector and respective movie vector is taken. So, for example if we need to predict what rating user A has given to movie *Titanic,* it can be written as:

User A · Titanic = User A'× Titanic (= 0.9*0.9+0.1*0.1 = 0.82)

.....and so on and so forth for other users and movies. (Note that User A' implies transpose of vector User A.) The ratings can be further normalized and/or scaled and adjustments for user biases can be done. As, it is evident from the table below *Titanic* is the best recommendation for User A, while for User E, *Fight Club* and *Rocky* are top two recommendations.

Ratings Matrix	Titanic	Rocky	The Hobbit	Fight Club	Jurassic Park
User A	0.82	0.18	0.5	0.1	0.26
User B	0.74	0.26	0.5	0.2	0.32
User C	0.34	0.69	0.5	0.7	0.62
User D	0.58	0.42	0.5	0.4	0.44
User E	0.1	0.9	0.5	1	0.8

Ratings calculated for Content-based approach

Content-based systems depend on external information for creating user and item profiles and this information might not be easily available. Also, these do not take users' behavioral information into account and discount the fact that user interest and preferences may change over time.

Collaborative Filtering

Collaborative filtering based systems collect & analyze users' behavioral information in the form of their feedback, ratings, preferences and activities. Based on this information, these then exploit similarities amongst several users/ items to predict missing ratings and hence make suitable recommendations. Collaborative filtering systems can discover and learn features on its own without the need of explicit features to profile items or users.

Memory-based/ Neighborhood-based

1. User-based

The key notion here is to determine users, who are like the target user A, and recommend ratings for the unobserved ratings of A by calculating weighted averages of the ratings of this peer group

Example- Suppose we need to find if user A will like the movie "*The Hobbit*". User-based approach finds n users (n is a number, 2 in below example) who have rated the movies similar to user A. Note that in the table below, user C and user E have rated the movies in a fashion similar to user A. In practice, similarity measure like Cosine similarity, Pearson similarity are used to find most similar items. To predict the rating that user A will give to *The Hobbit,* a weighted average of ratings of k most similar users is taken. Similarity metrics are used as weights between user A and corresponding similar users. If the predicted rating is high (say 4 or 5) the movie is recommended to user A.

	Ratings Matrix	The Godfather	Rocky	The Hobbit	Fight Club	Jurassic Park
⇨	User A	5	3	?	1	4
	User B	4	3	2	?	?
⇨	User C	5	3	5	1	4
	User D	1	1	1	4	1
⇨	User E	4	2	5	1	5

User-based CF example

2. Item-based

Example- Suppose we need to find if user A will like the movie "*The Hobbit*". Item-based approach finds n movies (n is a number, 2 in below example) which have been rated similar to *The Hobbit* by other users. Note that in the table below, user ratings for *The Godfather* and *Jurassic Park* are similar to those for *The Hobbit*. To predict the rating that user A will give to *The Hobbit*, a weighted average of ratings of k most similar movies is taken. Similarity metrics like Cosine similarity, Pearson similarity etc. used to calculate corresponding similarity weights between *The Hobbit* and other similar movies. If the predicted rating is high (say 4 or 5) the movie is recommended to user A.

	⇩		⇩		⇩
Ratings Matrix	The Godfather	Rocky	The Hobbit	Fight Club	Jurassic Park
User A	5	3	?	1	?
User B	4	3	4	?	3
User C	5	?	5	?	4
User D	1	2	1	4	1
User E	4	2	5	1	5

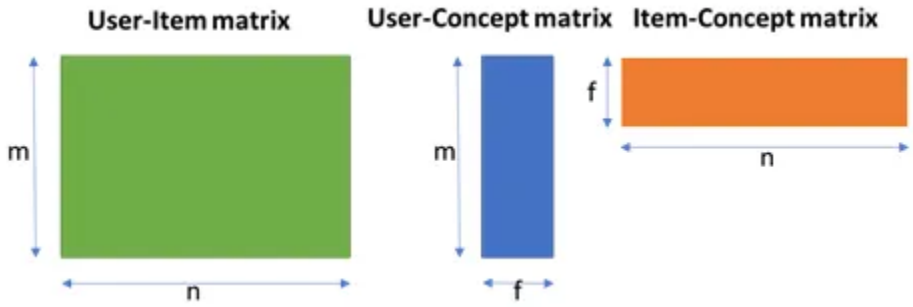
Item-based CF example

Model-based/Matrix Factorization

Model-based Collaborative Filtering approach employs dimensionality reduction techniques like matrix factorization (Singular Value Decomposition — SVD, Principal Component Analysis- PCA and Latent Factor models) to **discover hidden**

concepts and their relationship with users and items.

Matrix factorization allows User-Item rating matrix to be decomposed into smaller user-concept matrix and item-concept matrix, where each user and item is represented by f-dimensional vectors. Elements of user vector measure the extent of interest user has in each of the f concepts (features/factors). Elements of item vector measure the extent of strength of each of the concepts against that item. The dot product of user and item vectors $i \cdot u$ = **i'u** is used to estimate user's interest in a specific item. The error can be calculated by finding the difference between actual ratings and estimated rating values.



Matrix Factorization

Referring to the movie recommendation example above, user-movie rating matrix can be represented by lower dimensional user-concept and movie-concept matrix where romance and action could be the latent concepts (not known in advance) or features defining the user-movie interactions. Choice of f is a trade-off between discovering hidden concepts and avoiding overfitting, therefore, f should be chosen wisely.

Various machine learning algorithms like Stochastic Gradient Descent and Alternating Least Squares have been used to minimize error in rating predictions and hence increase the accuracy of recommendation systems. You can refer to this detailed report on matrix factorization.

For large number of users and items, collaborative filtering approaches however suffer from sparsity problems (as many of the items are not rated by users) and hence accuracy of the system falls-off. However, model-based CF systems are more

scalable as compared to memory-based systems and are more robust to deal with sparsity issues, hence improving the accuracy. Unlike CB recommender systems, CF recommender systems are more versatile and do not require feature modeling

Machine Learning · Recommendation System · Recommender Systems · Data Science

Hybrid Approach

Memory-based and model-based collaborative filtering approaches can be combined in practice to exploit the benefits each of the approaches provide. Also, content-based and collaborative filtering approaches can be combined in various ways to achieve greater synergies between them.

Thanks for reading. Did you find this article useful? Please share your views in the comments section below.

Follow

Written by Chhavi Saluja

631 Followers

Cloud Strategy Advisor at Capgemini Invent | Data Science & Machine Learning Enthusiast

---

More from Chhavi Saluja