Introduction to Recommender Systems

Nilavo Boral

11th April, 2022

E-commerce and retail companies are leveraging the power of data and boosting sales by implementing recommender systems on their websites. The use cases of these systems have been steadily increasing within the last years and it's a great time to dive deeper into this amazing machine learning technique.

In this project we will see the broad types of popular recommender systems, how they work, and how they are used by companies in the industry.

Further, we'll discuss the high level requirements for implementing recommender systems, and how to evaluate them.

What are recommender systems?

Recommender systems aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that online retailers implement in order to drive sales.

Data required for recommender systems stems from explicit user ratings after watching a movie or listening to a song, from implicit search engine queries and purchase histories, or from other knowledge about the users/items themselves.

Sites like Spotify, YouTube or Netflix use that data in order to suggest playlists, so-called Daily mixes, or to make video recommendations, respectively.

Why do we need recommender systems?

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

Recommendations typically speed up searches and make it easier for users to access content they’re interested in, and surprise them with offers they would have never searched for.

What is more, companies are able to gain and retain customers by sending out emails with links to new offers that meet the recipients' interests, or suggestions of films and TV shows that suit their profiles.

The user starts to feel known and understood and is more likely to buy additional products or consume more content. By knowing what a user wants, the company gains competitive advantage and the threat of losing a customer to a competitor decreases.

Providing that added value to users by including recommendations in systems and products is appealing. Furthermore, it allows companies to position ahead of their competitors and eventually increase their earnings.

How does a recommender system work?

Recommender systems function with two kinds of information:

- *Characteristic information*. This is information about items (keywords, categories, etc.) and users (preferences, profiles, etc.).
- *User-item interactions.* This is information such as ratings, number of purchases, likes, etc.

Based on this, we can distinguish between three algorithms used in recommender systems:

- *Content-based* systems, which use characteristic information.
- *Collaborative filtering* systems, which are based on user-item interactions.
- *Hybrid systems*, which combine both types of information with the aim of avoiding problems that are generated when working with just one kind.

Next, we will dig a little deeper into content-based and collaborative filtering systems and see how they are different.

Content-based systems

These systems make recommendations using a user's item and profile features. They hypothesize that if a user was interested in an item in the past, they will once again be interested in it in the future. Similar items are usually grouped based on their features. User profiles are constructed using historical interactions or by explicitly asking users about their interests. There are other

systems, not considered purely content-based, which utilize user personal and social data.

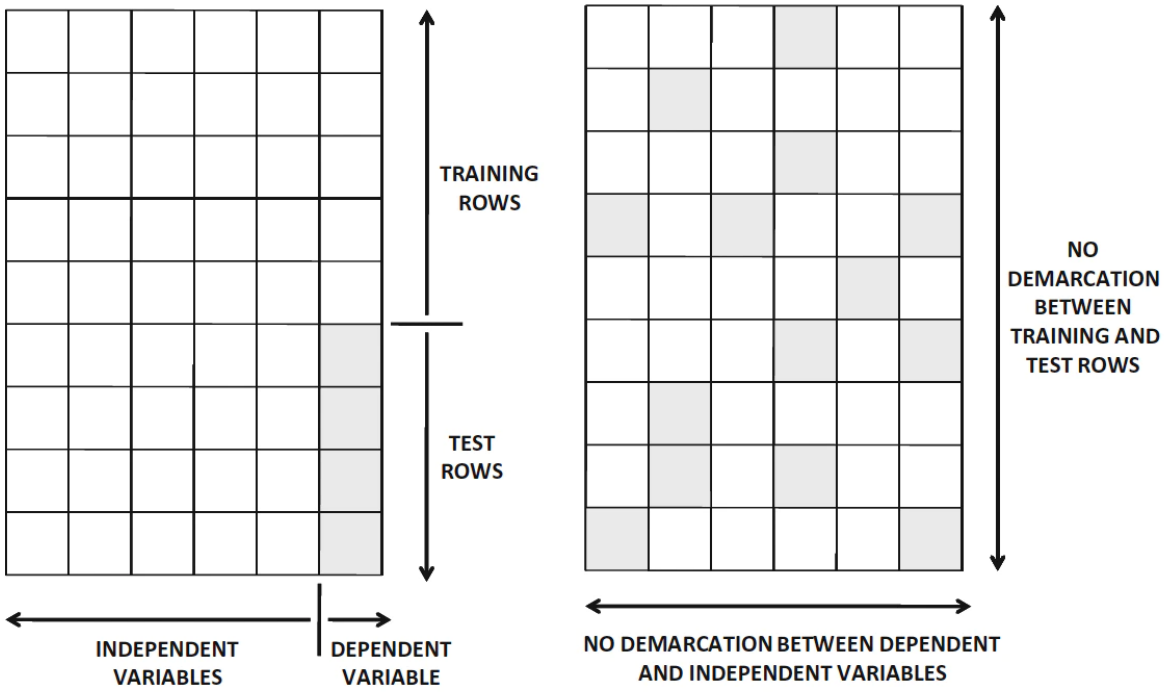
One issue that arises is making obvious recommendations because of excessive specialization (user A is only interested in categories B, C, and D, and the system is not able to recommend items outside those categories, even though they could be interesting to them).

Another common problem is that new users lack a defined profile unless they are explicitly asked for information. Nevertheless, it is relatively simple to add new items to the system. We just need to ensure that we assign them a group according to their features.

Collaborative filtering systems

Collaborative filtering is currently one of the most frequently used approaches and usually provides better results than content-based recommendations. Some examples of this are found in the recommendation systems of YouTube, Netflix, and Spotify.
These kinds of systems utilize user interactions to filter for items of interest. We can visualize the set of interactions with a matrix, where each entry (i,j) represents the interaction between user i and item j. An interesting way of looking at collaborative filtering is to think of it as a generalization of classification and regression. While in these cases we aim to predict a variable that directly depends on other variables (features), in collaborative filtering there is no such distinction of feature variables and class variables.

Visualizing the problem as a matrix, we don't look to predict the values of a unique column, but rather to predict the value of any given entry.



Classification vs Collaborative filtering

In short, collaborative filtering systems are based on the assumption that if a user likes item A and another user likes the same item A as well as another item, item B, the first user could also be interested in the second item. Hence, they aim to predict new interactions based on historical ones. There are two types of methods to achieve this goal: *memory-based* and *model-based*.

- ***Memory-based***

There are two approaches: the first one identifies clusters of users and utilizes the interactions of one specific user to predict the interactions of other similar users. The second approach identifies clusters of items that have been rated by user A and utilizes them to predict the interaction of user A with a different but similar item B. These methods usually encounter major problems with large sparse matrices, since the number of user-item interactions can be too low for generating high quality clusters.

- ***Model-based***

These methods are based on machine learning and data mining techniques. The goal is to train models to be able to make predictions. For example, we could use existing user-item interactions to train a model to predict the top-5 items that a user might like the most. One advantage of these methods is that they are able to recommend a larger number of items to a larger number of

users, compared to other methods like memory-based. We say they have large *coverage*, even when working with large sparse matrices.

Issues with collaborative filtering systems :

There are two main challenges that come up with these systems:

1. Cold start: we should have enough information (user-item interactions) for the system to work. If we setup a new e-commerce site, we cannot give recommendations until users have interacted with a significant number of items.
2. Adding new users/items to the system: whether it is a new user or item, we have no prior information about them since they don't have existing interactions.

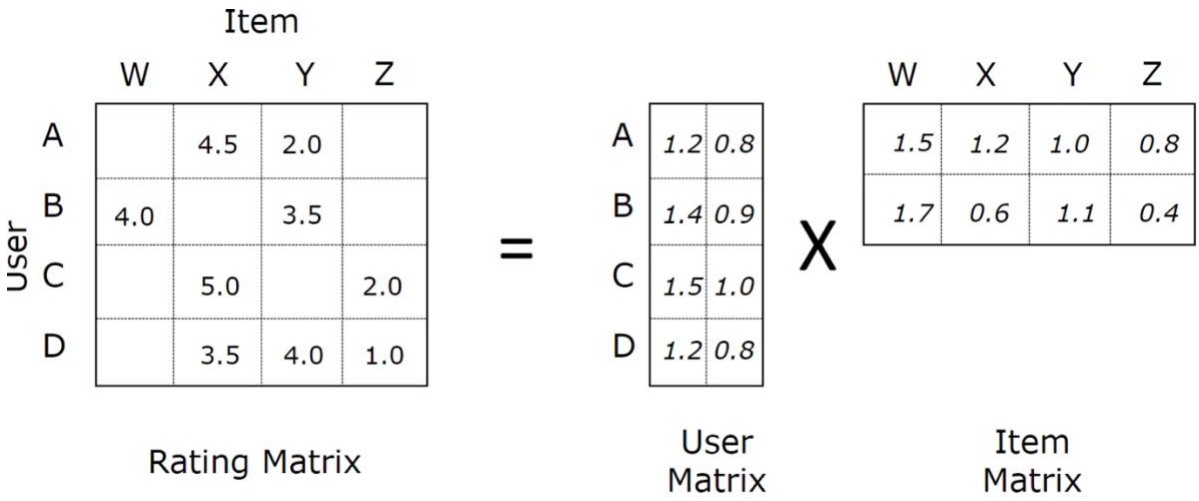
These problems can be alleviated by asking users for other type of data at the time of sign-up (gender, age, interests, etc), and using meta information from the items in order to be able to relate them to other existing items in the database.

What techniques are used to build recommender systems?

There are two techniques for building a collaborative filtering system: fully-connected neural networks and Item2vec.

Fully-connected neural networks

One classic approach is matrix factorization. The goal is to complete the unknowns in the matrix of user-items interactions (let's call it R). Imagine that we somehow, magically, have two matrices U and I, such that $U \times I$ is equal to R *in the known entries*. Using the $U \times I$ product we will also have values for the *unknown entries* of R, which can then be used to generate the recommendations.



Matrix factorizations

A smart way to find matrices U and I is by using a neural network.

First, we have to map each user and item to a vector with dimensions M and N, respectively. This means we need to learn *representations* of users and items, usually called *embeddings* (because we are embedding these concepts into a vector space). As we don't yet know the values of these vectors, we will have to start from a random initialization.

Then, for each user-item interaction (u,x) we will concatenate both embeddings of user u and item x to give us a single vector. As we already know the value of this user-item interaction, we can force the output of the network for this vector to be such. Then, the network will use backpropagation to adjust both its own weights and the embeddings themselves, so the result matches what we expect. Thus, the network will be learning the best way to represent users and items and will be useful to predict interactions which it hasn't seen before by feeding it with the resulting embeddings.

For example, let's look at the image above and suppose that 'User Matrix' and 'Item Matrix' are our randomly initialized embeddings. For interaction (A,X), we'll feed our neural network with the vector $[1.2,0.8,1.2,0.6]$ and force its output to equal 4.5. For this example, we could use MSE as the loss function. If we had a binary matrix of interactions, it would be appropriate to use a more common loss function in classification problems, like cross entropy.

A very interesting result of this approach is that the embeddings usually contain certain semantic information. Thus, we don't end up with only the predictions of unknown interactions, but we gather insights that we can make to be actionable. For examples, similar users will end up closer to each other in the user vector space. This could, for example, be useful for studying how customers behave.

Item2vec

Item2vec proposes that embeddings for the items can be found using the same technique as Word2vec. It utilizes store purchase orders as contextual information, which implies that items bought under analogous circumstances are comparable (and will have very similar representations within the space in which the embeddings live).

This approach neither *directly* involves the users nor considers them at the moment of making recommendations. Yet, it could be very useful if our goal is to show the user alternatives for a certain item they have chosen ("you bought this TV, you might also like these other ones").

The main issue we have here is that we need tons of data to produce good embeddings. In the Item2vec paper, two datasets were used; one with 9 million interactions, 732 thousand users, and 49 thousand items, and other with 379 thousand interactions, 1706 items, and no information about users.

When to implement a recommender system?

Now that we have some understanding of recommender systems, it's time to think about when it's worthwhile to implement one.

If you're running a successful business, you could probably survive without a recommender system. However, if you want to leverage the power of data to create a better user experience and to increase earnings, you should seriously consider implementing one.

Is it worth investing in a good recommendation system? A good way to answer this question is to look at how companies that have implemented such systems have fared:

- 35% of the purchases on Amazon are the result of their recommender system, according to McKinsey.
- During the Chinese global shopping festival of November 11, 2016, Alibaba achieved growth of up to 20% of their conversion rate using personalized landing pages, according to Alizila.
- Recommendations are responsible for 70% of the time people spend watching videos on YouTube.
- 75% of what people are watching on Netflix comes from recommendations, according to McKinsey.
- Employing a recommender system enables Netflix to save around $1 billion each year, according to this paper written by an executive:

"Reduction of monthly churn both increases the lifetime value of an existing subscriber, and reduces the number of new subscribers we need to acquire to replace canceled members. We think the combined effect of personalization and recommendations save us more than $1B per year."

- Cross-selling and category-penetration techniques increase sales by 20% and profits by 30%, according to McKinsey.

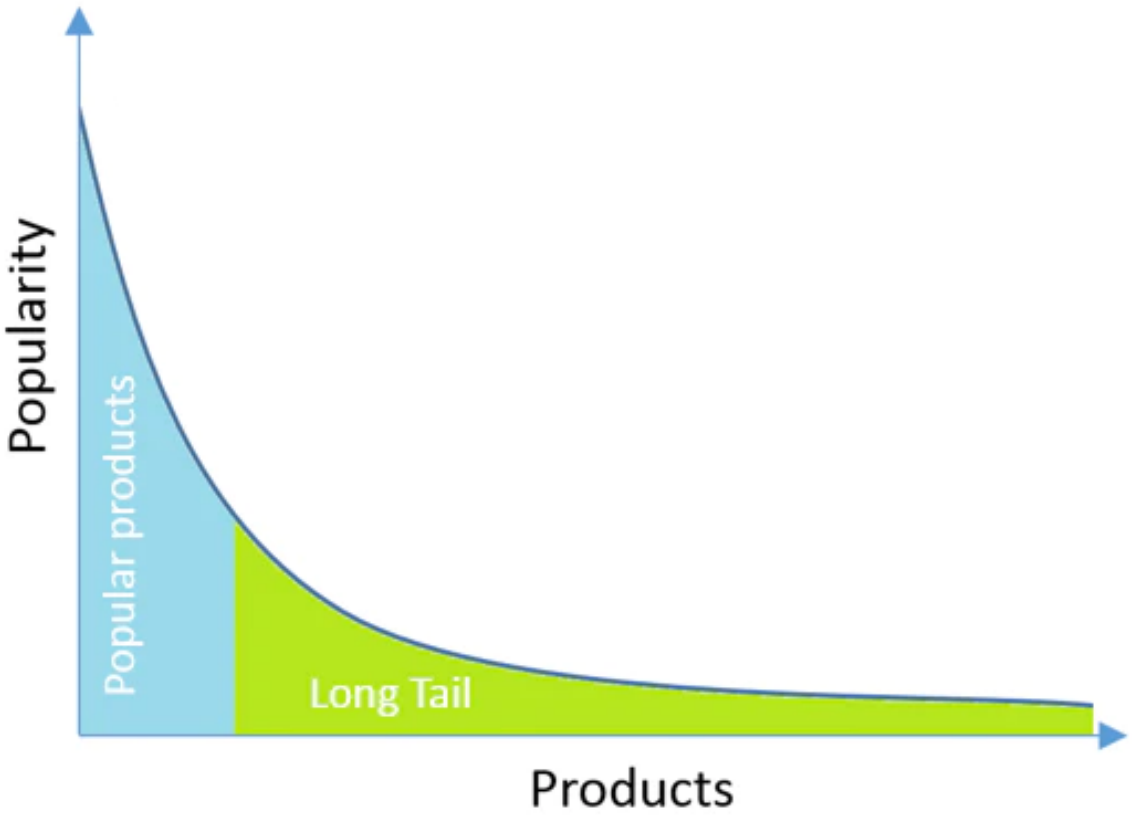
What are the prerequisites for building a recommender system?

Data is the single most important asset. Essentially, you need to know some details about your users and items. If metadata is all you have available, you can start with content-based approaches. If you have a large number of user interactions, you can experiment with more powerful collaborative filtering.

The larger the data set in your possession, the better your systems will work. Furthermore, you have to be sure you're staffed with a team that is able to understand the data and manipulate it correctly to allow for it to be ingested by the techniques you'll utilize.

Some things to keep in mind regarding the user-item interactions:

- You should define the interactions with respect to your system so that data can be extracted For example, if you're working on an e-commerce site, the interactions could include clicks on an item, searches, visits, favourite items, purchases, explicit ratings, elements in a shopping cart, or even discarded products, among others.
- The interactions can be defined as *explicit* or *implicit*. Explicit is characterized by situations such as when the user shows either positive or negative interest in an item, such as ranking it or leaving a review. Implicit is when the user's interest is derived from their actions, like searching for or buying an item.
- The larger the number of interactions per user and item, the better the final results will be.
- Typically, there are very popular items that users interact with a lot and others that they don't, which comprise what is known as the *Long Tail*. Recommender systems usually work pretty well on popular items, although that's probably not very interesting to users as they most likely already know about them. The items in the Long Tail are the most interesting ones, because they may not be considered by the user at all if they aren't recommended.



Long tail

Within the context of launching a new product, implementing a recommendation system from scratch won't be easy. A content-based approach would come in handy after the users start interacting, or you could ask them explicitly about their interests to help you at the beginning. Once the volume of users and interactions increases, it's time to start contemplating a collaborative-filtering approach to augment the potential of your system.

Finally, evaluating the system and thinking about different ways of improving its performance will likely be the most difficult task.

How to evaluate a recommender system?

Recommender systems have different ways of being evaluated and the answer which evaluation method to choose depends on your goal. If you're solely interested in recommending the top 5 items (i.e. the most probable items the user will interact with), you don't need to consider the predictions regarding the rest of the items when conducting the evaluation.

However, you could very well be interested in the order of priority of those 5 recommendations, so you would have to consider this. The chosen manner of

evaluating has an important effect on the way you design the system. Two types of recommender system evaluations are frequently discussed: *online* and *offline* approaches.

Online methods

With online methods (also called *A/B testing*), user reactions are measured given the recommendations made. For example, you can measure when the user clicks on the recommended items — as well as the conversion rate — and evaluate the direct impact of the system. This approach to evaluation is ideal, although its usually hard to implement since the only way to run the experiments is by interacting with the system that is already in production. Any failed experiment will likely have a direct impact on revenue and user experience. Moreover, using your real customers for experiments will be slower than if you already had the data beforehand.

Offline methods

The offline methods are ideal for experimental stages, since the user isn't directly involved, and unlike online methods, the system doesn't have to be deployed. The data is split into training and validation datasets, which means that part of the data will be used to construct the system and the other part to evaluate it. When using these methods, one needs to be careful because there may be factors that affect the results and cannot be adequately represented. For example, the time factor may be very important in the recommendations (seasonality, weather, etc), as might be the mood of the customer in a certain point in time.

Final remarks

As we saw in this post, including recommendations in systems is an attractive bet. From the user's point of view, it increases experience and creates engagement. For the business, it generates more revenue.

It's better to have a basic recommender system for a small set of users, and invest in more powerful techniques once the user base grows.

The most indispensable resource is the data. If you aren't managing or storing it properly, it's time to take the necessary steps to do so. Once you reach the implementation stage, digging deeper into the subject matter will surely be necessary.

Business goals will dictate the type of recommender system you should focus on at first: whether it is generating more engagement for already active users, or pushing those infrequent customers to become more active.

Besides defining the business goal, it is key that you're able to analyze and understand the information generated from your site. Given that, there's nothing that should stop you from a successful implementation of your recommender system.