# Recommender Systems

Onur Porsuk[1], Ayse Basar[2] and Suzan Uskudarli[3]

***Abstract*— There have been dramatic enhancements in recommender systems since they emerged. These developments allowed people in this field to take appropriate steps to satisfy their users. The users of the applications interact with items which may be a movie, song, document or game. As the data which is produced by users grows, today, a lot of items exist in applications to provide well utilization to users. A recommendation can be either personal or non-personal and depending on this, there have been many methods to do this successfully. In this paper, there will be sections about a recommender system and its fundamentals. From basic approaches to advanced techniques, different methods will be discussed and examples from the real world will take place.**

## I. INTRODUCTION

A recommender system, RS, is a sophisticated program that tries to recommend items to users in the application. For example, Amazon is one of the largest online shopping companies which uses RS at its core. Amazon provides millions of products to its customers and they improve the application day by day to be capable of serving well. It is crucial to have a good recommendation to make the users happy.

To start with, there are five types of recommender systems in general: Collaborative Filtering, Content-based recommendation, hybrid approaches, Knowledge-based recommendation and Demographic RS. Each method has its own benefits and pitfalls. Someone who works in the recommender systems field should be careful in choosing the method. The success of the recommendation highly depends on the problem, dataset and goals. Hence, it is important to make the right decision during the development of the recommender.

[1]Onur Porsuk, CMPE, Boğaziçi University
[2]Ayse Basar, CMPE, Boğaziçi University
[3]Suzan Uskudarli, CMPE, Boğaziçi University

In an application, there might be millions of users and items. One of the major problems in RS is that there is no real scenario that all users interact with all items. In large applications, users observe only a very small portion of items in the item space. When the application collects feedback from users after observations, for instance, the most popular way is the 5-star rating, then there will be very little information about the feedback. In other words, most items will remain unobserved by many users. There are different methods to overcome the missing information in RS.

In the paper, different subjects will be discussed under the relevant sections. In the following, different RS types will be handled. In the third section, one of the widely used algorithms in RS will be presented. The next section will be about the feedback types. In the fifth section, latent factor models will take place, and in the sixth, some additional topics about implicit and explicit feedback will be discussed.

## II. TYPES OF RECOMMENDATION SYSTEMS

As it was mentioned in the previous section, different types of RSs exist to perform a good recommendation. An engineer or developer may use different approaches for their task depending on the capabilities of the method. In this section, there will be brief information about each approach.

### A. Collaborative Filtering

This is one of the widely used methods in RS to make recommendations to users. It considers the similarity between either users or items. Collaborative filtering, CF, consists of two main types, which are memory-based and model-based methods. [1]

In memory-based, the implementation can be either user-based or item-based. In the user-based technique, the system tries to calculate how similar are users to each other by considering their past

observations. To give an example, in a song stream application, two users A and B may have listened to various songs. If we want to know, whether user A would like a specific item, we can check if user B or other similar users to A have listened to the song or not. If A's top n similar users listened to the song, we may recommend the song to A. This is just a basic example of how to use user-based CF. Instead of observation or no observation (1 or 0), a rating could be used as well. This time, the calculation would be on the ratings' values instead of interaction information. The similarity can be measured using items to make recommendations. To measure similarity, *Cosine* or *Pearson* similarity techniques can be used. When there is user A, if the application wants to decide whether it should recommend an item K to user A, it looks similar to items to K. If user A observed similar items to K, then the system can decide that A will probably observe K too. The concepts of these similarities are the same, the only difference is the objects. Usually, in real-life scenarios, item-based performs better than user-based CF. The data that is used is a matrix which contains users and items in rows and columns. This matrix is called *User Rating Matrix (URM)*, where there are rating values in cells. In memory-based is easy to implement and understand to explain results but due to the large matrix, it may not be scalable for performance issues. [2]

The second approach is a model-based method in CF. In this method, instead of measuring the similarity of users or items, Machine Learning (ML) is used to build a model which would learn the patterns of the feedback dataset. A diagram of the process was shown in Figure 1.

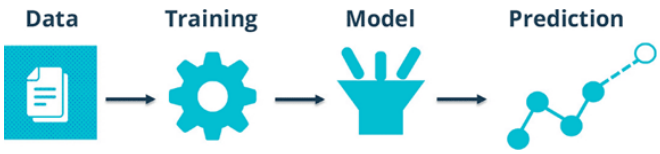


Fig. 1. Model-based CF

In an ML model development process, there is a dataset which is used in training the algorithm. After training, the model is created and it is utilized to make future predictions. This is the most general pipeline of an ML process. To give examples, K Nearest Neighbour (K-NN) or multi-layered neural networks can be used to perform the task. Model-based CF works well since it uses the power of ML. In this approach, the URM is not directly brought into the main memory of a computer and used the entire matrix to make predictions. Instead, ML play role in learning the similarities between user and item. Hence, unlike memory-based, method-based consumes less memory. Another advantage of the model-based approach is that it handles the *cold start* problem. The problem is defined as the new user or item problem in the system. When there is a new user or item comes into the system, there is no past information and this means the CF is unable to make a personalized recommendation. When model-based is used, not for new users but for new items, it overcomes the cold start problem. Also, with the help of dimensionality reduction techniques, the method can deal with sparsity. Unlike memory-based, explainability may be difficult due to hidden layers and factors.

In CF, the main advantage is that there is no domain knowledge requirement. CF can be applied to any domain without additional effort to make the system work. But the URM is quite sparse, there are many missing values in the matrix and this means that there is quite a little information to process. To point out the fact that ML algorithms require dense (complete) input to work.

### *B. Content-based RS*

As the name of the method indicates, the content information of items is used when making recommendations. The content refers to the characteristic features of items. For instance, in a document reading application, there are various attributes of a single document such as topic, author, release date, special tags and keywords etc. The advantage is that there is no need to have information about other users since the content information is used to make predictions. But as it was an advantage in CF, here domain knowledge is highly required. For the document application example, there should be intense information about the genres, articles, blogs, authors, publishers etc.

The advantage of Content-based RS over CF is that generally there are both new user and new

item cold start problems in CF when using user-based or item-based approaches but in Content-based, the problem remains only for a new user. When having information about the items, the new item can easily be processed during recommendation because its similarity based on attributes may comfortably be calculated. Depending on the problem and dataset, usually, Content-based RS is more accurate in the recommendation.

### C. Hybrid Approach

In the implementation of a recommender system, hybrid approaches are powerful. Instead of one single method, engineers and developers can use more than one method together. For example, Collaborative filtering and Content-based RS may be used as a hybrid method to make recommendations. This allows them to use the strengths of each method and overcome each method's disadvantages. In the real world, usually, a hybrid approach is implemented to build a recommendation system.

### D. Knowledge-based RS

A Knowledge-based RS is based on explicit knowledge about the item assortment, user preferences, and recommendation criteria [3]. This type of RS is applied when CF or Content-based RS is not applicable in specific scenarios. It inherently handles the cold start problem but requires deep information. These RS systems are well suited to complex domains where items are not purchased very often. For example apartments, land or cars. In such scenarios, there are fewer observations than in the other domains.

### E. Demographic RS

Demographic RS makes use of demographic attributes of users to recommend items. It does not require users' ratings or knowledge of the item [4]. Hence, it can overcome the cold start problem as Knowledge-based RS. This type of RS can be easily implemented and utilized.

## III. K-NN Algorithm

K Nearest Neighbour is one of the most widely used methods in similarity-based recommendation tasks. It has a straightforward way which is easy to understand. K-NN works fast and it can be quickly implemented. It basically works on distance measuring of samples (points) in the space. The general steps were shown in Figure 2.

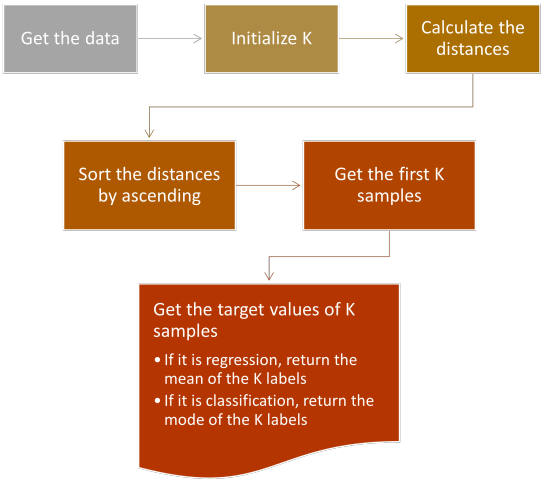


Fig. 2. K-NN - Steps

First, the dataset is captured and prepared for training. K-NN requires the *K* value to work. This number is the number of sample observations in each iteration. For example, if K is taken 5, in each iteration, the 5 closest neighbours are used to decide which class the sample belongs to. In distance measurement, different methods such as *Euclidean, Manhattan or Mahalanobis* can be used. After deciding K, for each sample in the set, their corresponding closest neighbours are found by measuring the distance. If there is no improvement, the naive way is to calculate distances from the observed sample to all other samples in the space. Next, they are sorted by distance. The closest K ones are gathered. Depending on the problem, *classification or regression*, the decision of assigning the observed value is done by considering the K closest neighbours. If it is classification, *majority voting* would be used, which means the most frequent class would be the class of the observed sample.

## IV. Types of Feedback

In a recommender system, there are mainly two types of feedback: explicit and implicit. Explicit means that users directly mention their opinions about the item by either giving a rating or leaving a text comment. On the other hand, implicit

feedback is not captured from users directly, they are the system monitoring-based information that the application collects. For instance, on a website, implicit feedback can be the time that the user spends on specific items, the clicks, the position of clicks, the user's preferences or settings etc.

Explicit feedback is more accurate than implicit because it is direct and exact information about the likes or dislikes of the user for the item. But in real scenarios, it is challenging to get feedback from users, the studies show that there is very little information about the ratings. Unlike explicit, there is abundant information in implicit feedback. A system can monitor the users' behaviours continually. However, the interpretation of implicit feedback is not as easy as explicit. Because an engineer cannot be sure whether the information indicates a user liked an item or disliked it. There are numerous records and log information from the system. It is challenging to make an implicit dataset structured to use it as easily as an explicit one. Apart from these two feedback types, people may use a combination of them in developing recommenders. Usually, it strengthens the feedback's positive impact on the performance of the recommender.

## V. Latent Factor Models

User Rating matrix is highly sparse due to the missing values and many non-observations between users and items. Instead of processing the whole matrix, it can be written as sub-matrices multiplication. This is useful to reduce the dimensionality and process smaller sizes of matrices. This approach is called matrix factorization. The models created by this approach are named latent factor models. As this method reduces the computational time, it generates some errors because of the transformation. Matrix factorization essentially maps the features into a smaller space and represents each user and item on the space as points. The item which any user observes or likes is close to the users in the space, in other words, the distance between them is short. The estimated rating formula is shown in Equation 1. In the formula, *u* and *i* represent user and item indices respectively. An estimated rating value equals to the multiplication of user and item *latent vectors*.

$$r_{ui} = q_i^T p_u \tag{1}$$

In matrix factorization, there are different methods and Singular Value Decomposition (SVD) is one of the widely used approaches. The difference between base matrix factorization in SVD is that it depends on what loss function and what properties are desired from the result. But essentially they work in the same way. The formula of SVD is shown in Equation 2. There is the third component in the formula which is a diagonal matrix that contains singular values.

$$A = U\Sigma V^T \tag{2}$$

It is important to note that SVD requires a dense matrix, otherwise, it does not work. Therefore, engineers must deal with the missing values in URM by imputing the missing values. Moreover, matrix factorization models learned by SVD have shown to be very prone to *overfitting* [5], which is a common problem in machine learning tasks. Therefore, some *regularization* must be applied to reduce the negative impact of intensive learning. The details of these topics will not be covered in this report, since they are relatively deep subjects.

## VI. More Topics About Feedback Types

As was discussed in the fourth section, two types of user feedback are utilized to train a model and create a recommendation system. In explicit type, there is a clear structure and is straightforward to use. However, it may be challenging to make use of implicit datasets. Inherently, they do not contain direct information about users whether they liked or disliked an item. Engineers can infer statements from the dataset. In addition, they have to handle the weight of positive and negative class examples in the dataset. Positive refers to the likes and negative refers to the dislikes of users. As explicit feedback is sparse, in implicit feedback, there are way more samples in the negative class rather than the positive. Hence, it is important to use some sort of sampling technique to reduce the undesired impacts of negative samples on the performance of the model.

### A. Negative Sampling

When using implicit feedback, different techniques exist in negative sampling. There are essentially two main categories which are Heuristic Negative Sampling (HNS) and Model-based Negative Sampling algorithms. Under HNS, Random Negative Sampling (RNS) and Popularity-biased Negative Sampling (PNS) take place. RNS is simple and easy to implement, as it picks samples randomly and does not consider any weight information. Unlike RNS, PNS weights the samples depending on their popularity and this increases the amount of information of negative samples [6].

There is a method called *Weighted Regularized Matrix Factorization, (WRMF)* which weights and regularizes the matrix factorization process. It uses a coefficient as a regularization term. To sum up the negative samples' impact, different techniques are used to reduce the impact of unobserved or negative samples in the training period.

### B. SVD++ and timeSVD++

Using explicit and implicit feedback together is usually more efficient than using a single one. When these two types are combined, their impact on training should be carefully handled. An improvement of SVD, which is called SVD++ is a method to include the effect of implicit information. In the name, the '++' part means incorporating implicit feedback [7]. In the paper, Y. Koren presents SVD++ in detail with corresponding formulas. This method may work well on a combination of explicit and inferred implicit datasets.

Another version of SVD is called timeSVD++, where the bias and factor terms are a function of time, in other words, they change by time. This allows using time information in the recommendation system. The context information in a recommender may be critical since users' behaviours change by time, location or even their mood. Adding and using this information increases the success of the recommender.

## VII. Conclusion

In this report, some fundamental topics in recommender systems were discussed. A recommendation system can be built using various methods and datasets. Each one has its benefits and pitfalls. Usually, hybrid versions outperform the single approaches due to the utilization of each method's strengths. There are more subjects in recommender systems such as Bayesian Personalized Ranking (BPR), context awareness, evaluation of recommendations systems and blockchain-based recommenders. In further studies, these subjects will be covered and explained in another report.

## References


[1] Aggarwal, C. C. "Recommender Systems: The Textbook". Springer 2016. ISBN 978-3-319-29657-9

[2] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., Recommender Systems Handbook (2011).

[3] Wikipedia contributors. (2020, March 7). Knowledge-based recommender system. Wikipedia. https://en.wikipedia.org/wiki/Knowledge-based_recommender_system

[4] Sridevi, M., and R. Rajeswara Rao. "Decors: A simple and efficient demographic collaborative recommender system for movie recommendation." Adv. Comput. Sci. Technol 10.7 (2017): 1969-1979.

[5] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L., 2009, June. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence (pp. 452-461). AUAI Press

[6] Jiang, J. (2021) Overview negative sampling on recommendation systems, Medium. MLearning.ai. Available at: https://medium.com/mlearning-ai/overview-negative-sampling-on-recommendation-systems-230a051c6cd7

[7] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008.