Recommender Systems

BILICI, M. Şafak
safakk.bilici.2112@gmail.com

UYSAL, Enes S.
enessadi@gmail.com

Contents

1 Motivation

How do we recommend our friends movies that we have seen or books as humans? What do we consider when recommending movies or books to someone we do not know?

Recommendation Systems try to recommend things to us as well as our friends with this pipeline in Figure 1.

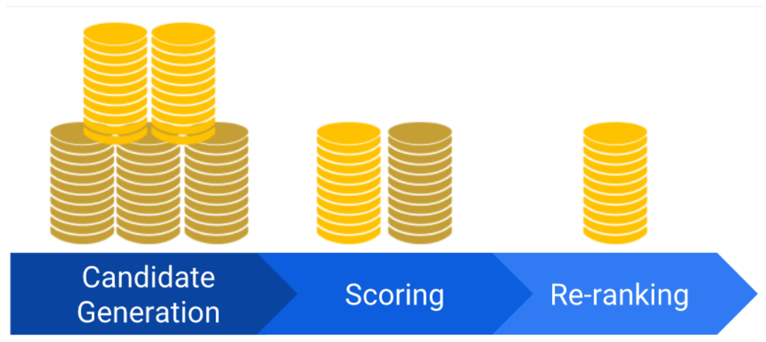
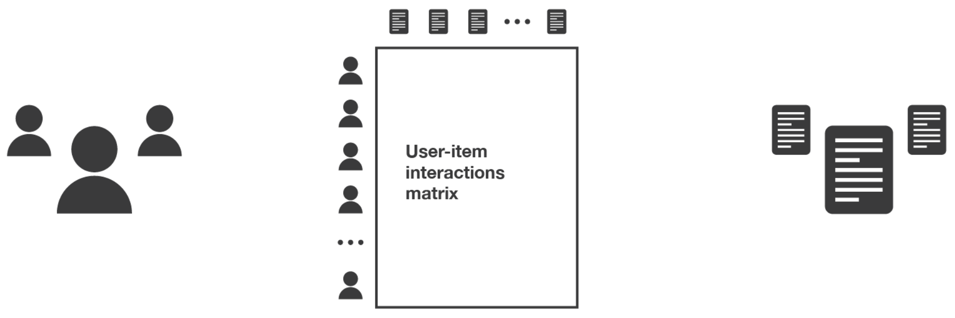


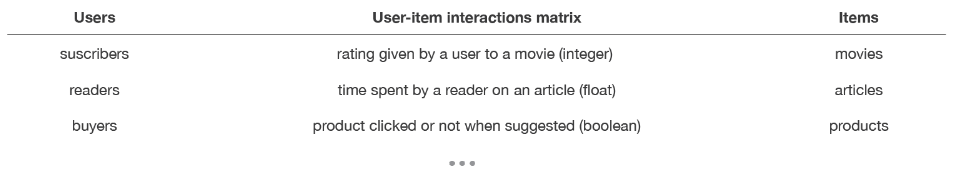
Figure 1: Recommendation Pipeline

Candidate generation can be part of many approaches like **Collaborative Approaches** and **Content Based Approaches**. Then the system scores these item candidates and eliminate some of them. Finally looks at their scores and re-rank them for recommend to the user and pick the most relevant (sometimes pick more than once or do not pick the most relevant).

2 Collaborative Approaches

Recommending a movie without watching it to someone who do not know, just by looking at other people's ratings to other movies.
Technically this approach uses only "user-item interactions matrix". Interactions can be numbers like 1 to 10 or just like/dislike, these kind of interactions are **explicit interactions**. Also interaction means just clicking the recommendation or watcing the movie that recommended, these kind of interactions are **implicit interactions**.





Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products
	...	

Figure 2: User-Item Interactions Matrix

Basic purpose of this approach is finding the most similar users (with Locality-sensitive hashing, which implements the nearest neighbor mechanism in linear time) or the most similar items with using different algorithms like Pearson Correlation, Vector Cosine or building models. The methods which do not build any model are called **Memory Based Methods** which use algorithms like kNN, the other methods which build models are called **Model Based Methods**.

2.0.1 Similarity Measures

When finding similarity between item vectors or user vectors, cosine similarity, dot product or euclidean distance can be used. Here is a comparison for these measures.

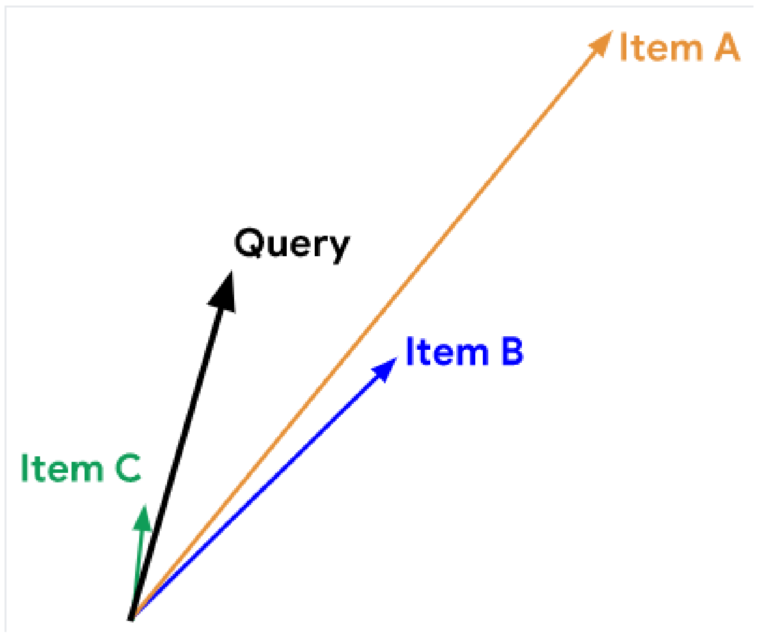


Figure 3: Item Vectors

If dot-product is used : Item A > Item B > Item C
If cosine similarity is used : Item C > Item A > Item B
If euclidean distance is used : Item B > Item C > Item A

Dot-product is not the right measure for like YouTube environment. (Rich-get-richer effect)

2.1 Memory Based Methods

Based Methods usually use "k nearest neighbour" algorithm for finding the most similar users or items to each other.

The complexity of kNN method for this task is "$O(ndk)$". n is for number of users, d is for number of items and k is for the number of neighbours considered. With this complexity, the method can have high computation cost and consume high amount of time. We can say that this method is not scalable for this reason.

As a solution for scalability problem "approximate nearest neighbours (ANN)" can be used or the advantage of sparsity of user-item interactions matrix can be used.

In memory based methods, the problem called "rich-get-richer" occurs if the algorithm is used without any restriction. This problem means popular items get more popular while the system recommend them or users get close to popular users while system recommends same items for everyone. Diversity of the system is important, so this problem has to taken into consideration.

Memory based methods separated from each other **item-item similarity method** and **user-user similarity method**.

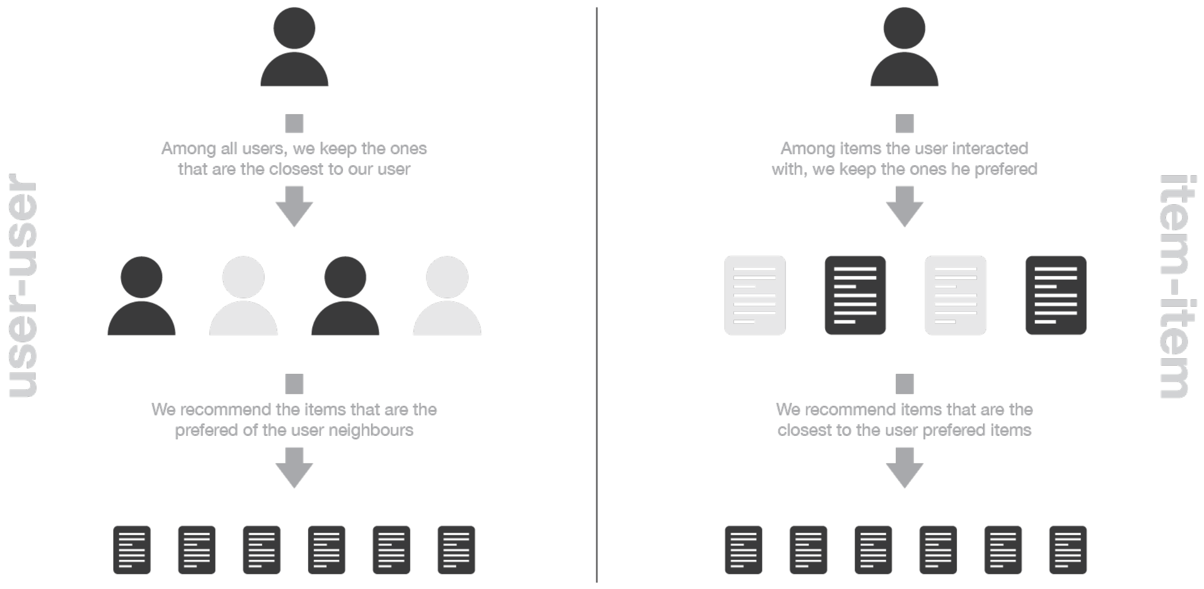

user-user
Among all users, we keep the ones that are the closest to our user
We recommend the items that are the prefered of the user neighbours
Among items the user interacted with, we keep the ones he prefered
We recommend items that are the closest to the user prefered items
item-item

Figure 4: Visual Explanation

2.1.1 User-User Similarity Method

User-User Similarity Method (also called "user-centred") recommends items to users by finding most similar "interactions profile" (nearest neighbours for users). This means while finding similar users, method looks at common interactions.

Generally users do not interact many items in item set. This causes the method becomes more sensitive to new interactions that means also has high variance. The method finds similar users by looking at a small number of interactions. So results are more personalized.

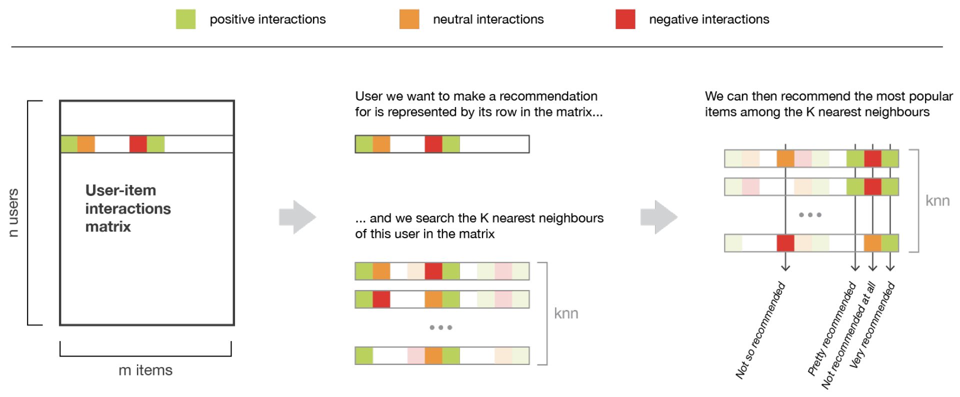
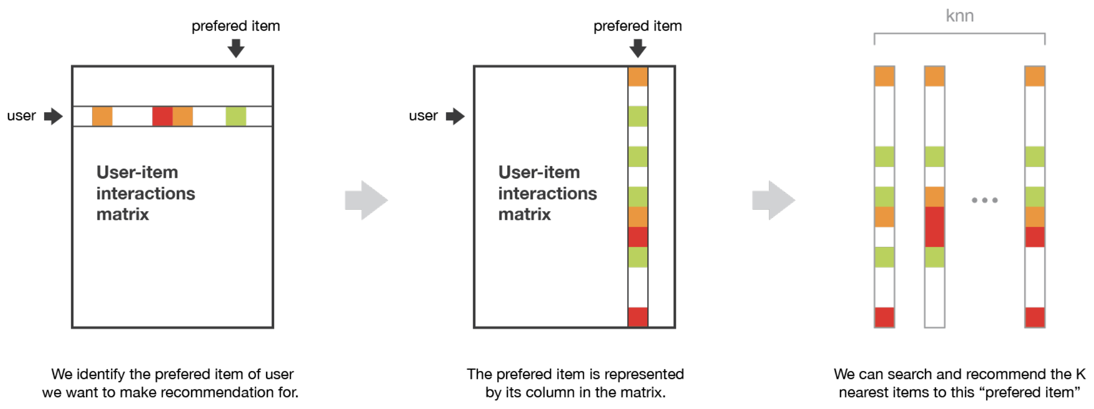

positive interactions
neutral interactions
negative interactions
n users
User-item interactions matrix
m items
User we want to make a recommendation for is represented by its row in the matrix...
... and we search the K nearest neighbours of this user in the matrix
knn
We can then recommend the most popular items among the K nearest neighbours
knn
Not so recommended
Pretty recommended
Not recommended at all
Very recommended

Figure 5: User-User Similarity Method

2.1.2 Item-Item Similarity Method

Item-Item Similarity Method (also called "item-centred") recommends items to users by finding similar items that the user already interacted with "positively" with using kNN algorithm mostly. Finds the "best item" that is new to the user and recommend it.

Items are rated by many users mostly. So the method has lower variance than user-user similarity method. Also the method looks at the items, not users so the results are less personalized than user-user similarity method which means higher bias.
This method is more robust than the other because of not effected new interactions that much.



prefered item
user
User-item interactions matrix
We identify the prefered item of user we want to make recommendation for.
prefered item
user
User-item interactions matrix
The prefered item is represented by its column in the matrix.
knn
We can search and recommend the K nearest items to this "prefered item"

Figure 6: Item-Item Similarity Method

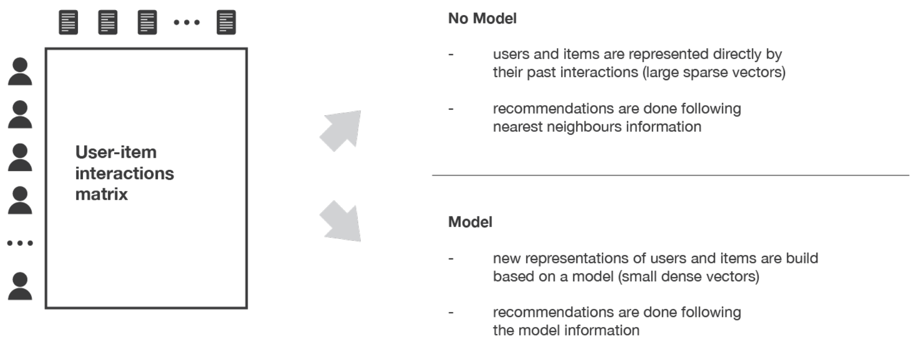
2.1.3 Advantages of Memory Based Collaborative Approaches

The explainability of the results, which is an important aspect of recommendation systems, is possible. Easy creation and use, easy facilitation of new data, content-independence of the items being recommended, good scaling with co-rated items.

2.2 Model Based Approaches

This approach build a model by using only user-item interactions matrix. The system recommends items to users just by using the created generative model, not by looking at all interactions.

Thanks to this approach, computation cost of the system is reduced and sparsity in interaction matrix is not a problem for the system anymore.



User-item interactions matrix
No Model
- users and items are represented directly by their past interactions (large sparse vectors)
- recommendations are done following nearest neighbours information
Model
- new representations of users and items are build based on a model (small dense vectors)
- recommendations are done following the model information

Figure 7: Approaches with/without Model

This model is an advantage because it learns dense vectors which represents users and items separately. Latent spaces of these vectors are hyper parameters and vectors themselves are parameters which are learnable. We can call every dimension of representation vectors as a feature of users/items. **Matrix Factorisation** is a method for this.

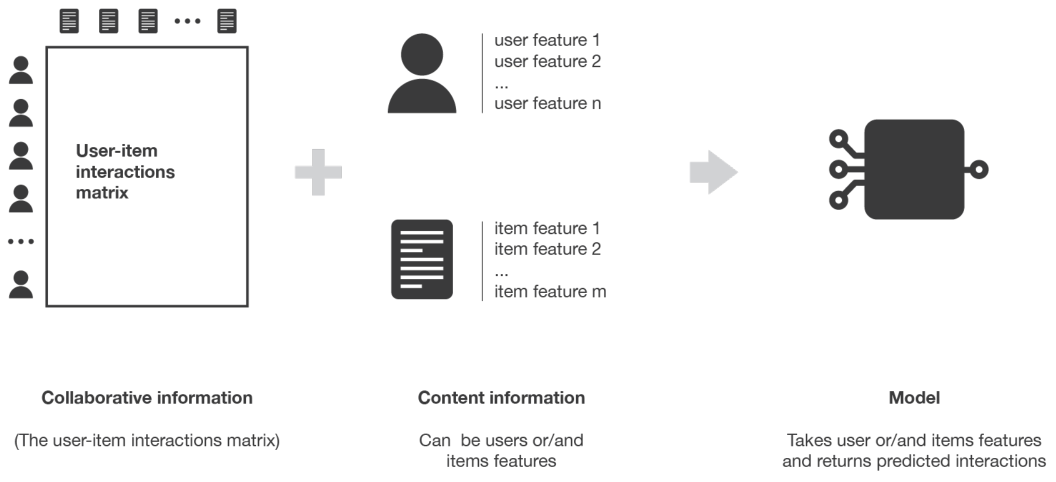

User-item interactions matrix
user feature 1
user feature 2
...
user feature n
item feature 1
item feature 2
...
item feature m
Collaborative information
(The user-item interactions matrix)
Content information
Can be users or/and items features
Model
Takes user or/and items features and returns predicted interactions

Figure 8: Model Creation

2.2.1 Matrix Factorisation

This method splits "user-item interaction matrix" into two parts as "user-factor matrix" for user representations and "factor-item matrix" for item representations. The model learns the features of items or users just by looking at interactions matrix. They cannot be said like this feature is for this characteristic. (Dimensionality Reduction)

While training, random initialized dense vectors are created with given latent space size. These matrices dot product with each other and created a reconstruction matrix, by looking at this matrix reconstruction error is calculated. At the end of the training model learns the vector representations well and make recommendations by multiplying them.

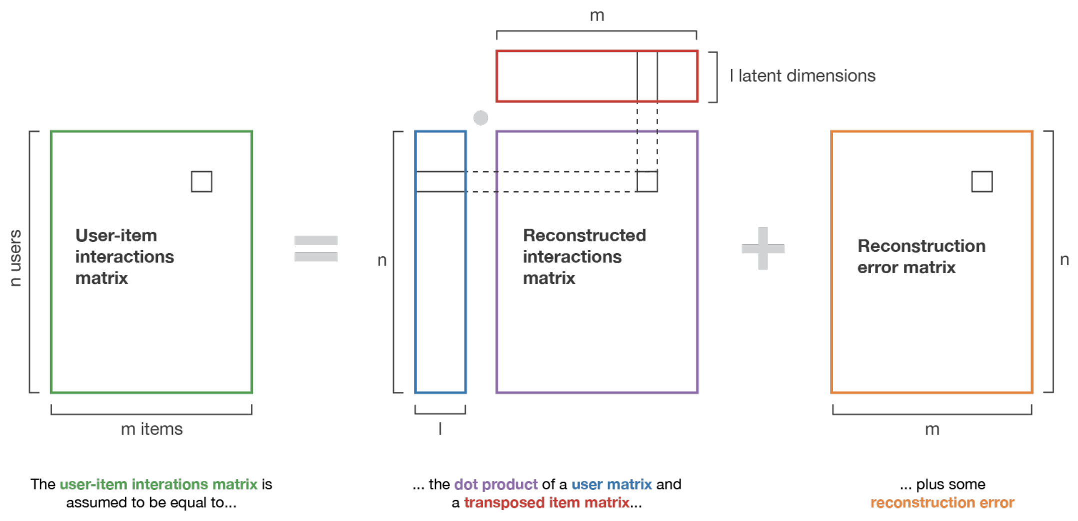

m
l latent dimensions
n users
User-item interactions matrix
m items
n
Reconstructed interactions matrix
l
Reconstruction error matrix
n
m
The user-item interations matrix is assumed to be equal to...
... the dot product of a user matrix and a transposed item matrix...
... plus some reconstruction error

Figure 9: Training Process Visualization

Let M to be "user-item interaction matrix". Size of this matrix is *(n x m)* which n is the number of users and m is the number of items.
This matrix divide into two matrices which named $X \in \mathbb{R}^{n \times l}$ "user matrix" and $Y \in \mathbb{R}^{m \times l}$ "item matrix". l is the given dimension for latent space. So this equation is the purpose of the system: $M \approx X \cdot Y^T$

Error function for training of this model is "rating reconstruction error". This measures how close the result of dot product to the original M matrix.

$$(X, Y) = \underset{X,Y}{\operatorname{argmin}} \sum_{(i,j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2$$

Regularization term can be added by using X and Y matrices:

$$(X, Y) = \underset{X,Y}{\operatorname{argmin}} \ \frac{1}{2} \sum_{(i,j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2 + \frac{\lambda}{2} \left(\sum_{i,k} (X_{ik})^2 + \sum_{j,k} (Y_{jk})^2 \right)$$

This model can also build as a neural network. This version called *"decomposition"* instead of *"factorization"* mostly. *f(.)* is the logistic function and algorithm optimizes model according to the logistic function. Deeper neural networks work on complex recommendation systems and mostly state of the art models are deep neural networks.

$$(X, Y) = \underset{X,Y}{\operatorname{argmin}} \ \frac{1}{2} \sum_{(i,j) \in E} [f((X_i)(Y_j)^T) - M_{ij}]^2 + \frac{\lambda}{2} \left(\sum_{i,k} (X_{ik})^2 + \sum_{j,k} (Y_{jk})^2 \right)$$

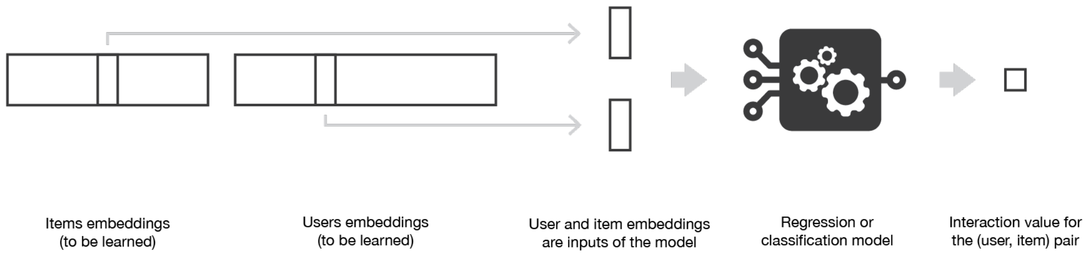

Items embeddings (to be learned)
Users embeddings (to be learned)
User and item embeddings are inputs of the model
Regression or classification model
Interaction value for the (user, item) pair

Figure 10: Another Visualization for Training Process

2.3 Problems with Collaborative Approaches

Collaborative approaches be like recommending a movie that do not watch to a stranger just by looking other people's ratings for other movies. So, it is all about interactions. More interaction means more accurate and more effective systems.

When new user or new item comes into system, system does not know anything about it, because anyone interact with it or new user interact with anything. While system recommending items to this user it does not know the characteristics of the user. This problem called *"Cold Start Problem"*.

- Data Sparsity: User-Item Matrix is large and sparse so this effects the performance of the system.
- Scalability: Dimensions of matrix get bigger, complexity gets bigger faster.

- Synonyms: Same item but different names like "children's movie" and "children's film". Topic Modeling (like the Latent Dirichlet Allocation technique) could solve this by grouping different words belonging to the same topic.
- Gray Sheep: Gray sheep refers to the users whose opinions do not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering. Black sheep are a group whose idiosyncratic tastes make recommendations nearly impossible.
- Shilling Attacks: Manipulation on ratings. Robust collaborative filtering, where recommendation is stable towards efforts of manipulation, can be used for this problem. This research area is still active and not completely solved.
- Diversity and The Long Tail: Rich-get-richer effect, prevents diversity in recommendations. Long tail and diversity problem can be reduced by recommending unexpected, serendipitous items.

2.4 Context-Aware Collaborative Filtering

This method uses also information like time, location, social information, and type of the device that user is using beside "user-item interaction matrix".

Method uses additional dimension to the existing user-item rating matrix. Thus, instead of using user-item matrix, we may use tensor of order 3 (or higher for considering other contexts) to represent context-sensitive users' preferences matrix factorization methods. Tensor factorization techniques are used instead of matrix factorisation in this method.

3 Content Based Approaches

Recommending a movie that have seen to your good friend by looking at the movie's features and your friend's characteristics, also looking at other people's ratings to other movies. Technically this approach not only uses "user-item interactions matrix". It also uses the features of items and the features of users which are predefined like the age of that person or sex of the person. Model does not have to learn latent vectors for users or items like with the Collaborative Approach.

Cold start problem is less than the collaborative approach because new users or new items come with their feature values. If new features are added the system, cold start problem still occurs with feature level.

3.1 Model with These Features

Many traditional or neural network models like bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks can be used in this approach. These models divide into three parts as *"user-centred"*, *"item-centred"* or *"hybrid"* models.

User-centred models use item features while making recommendations. These models has important advantage that they do not ask questions to users for their features. So it makes system better for user experience.

Item-centred models use user features. This models are much more robust than user-centred ones because, users interacts with only few items mostly but items get rated by many users.

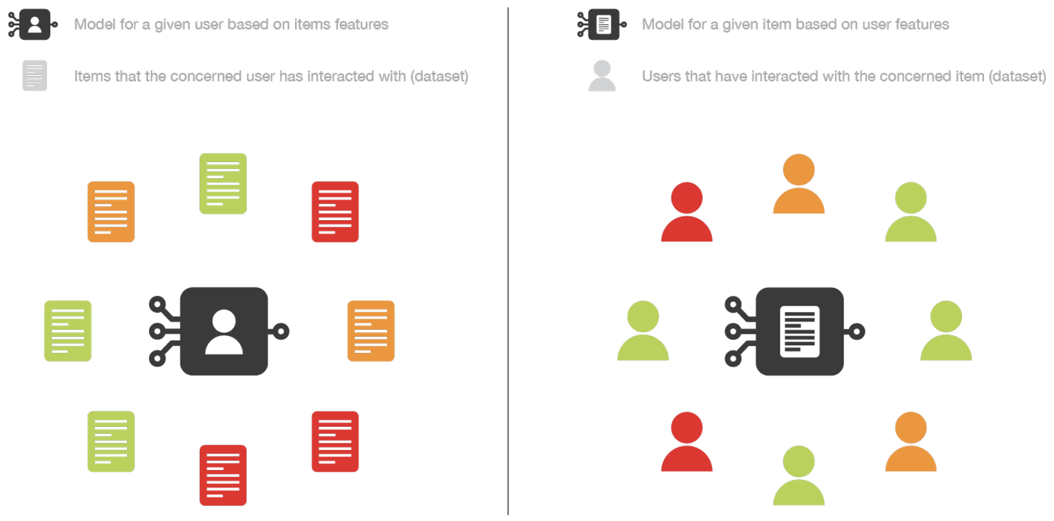

Model for a given user based on items features
Items that the concerned user has interacted with (dataset)
Model for a given item based on user features
Users that have interacted with the concerned item (dataset)

Figure 11: User-Centred & Item-Centred Models

Hybrid models works with both item and user features. They mostly are deep learning models.

3.1.1 Item-Centred Bayesian Classifier

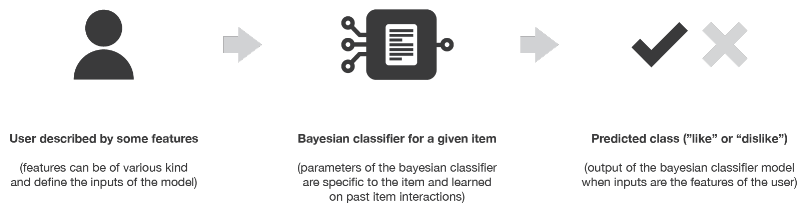

User described by some features
(features can be of various kind and define the inputs of the model)
Bayesian classifier for a given item
(parameters of the bayesian classifier are specific to the item and learned on past item interactions)
Predicted class ("like" or "dislike")
(output of the bayesian classifier model when inputs are the features of the user)

Figure 12: Item-Centred Bayesian Classifier

This is an example of item-centred classification model. Model learns how to recommend by predicting the user-item couples to like/dislike.
We want to compute that equation that probability of likes item or dislikes item.

$$\frac{\mathbb{P}_{item}(like \mid user\_features)}{\mathbb{P}_{item}(dislike \mid user\_features)}$$

$$\mathbb{P}_{item}(like \mid user\_features) = \frac{\mathbb{P}_{item}(user\_features \mid like) \times \mathbb{P}_{item}(like)}{\mathbb{P}_{item}(user\_features)} \tag{1}$$

$$\mathbb{P}_{item}(dislike \mid user\_features) = \frac{\mathbb{P}_{item}(user\_features \mid dislike) \times \mathbb{P}_{item}(dislike)}{\mathbb{P}_{item}(user\_features)} \tag{2}$$

$$\frac{\mathbb{P}_{item}(like \mid user\_features)}{\mathbb{P}_{item}(dislike \mid user\_features)} = \frac{\mathbb{P}_{item}(user\_features \mid like) \times \mathbb{P}_{item}(like)}{\mathbb{P}_{item}(user\_features \mid dislike) \times \mathbb{P}_{item}(dislike)} \tag{3}$$

3.1.2 User-Centred Linear Regression

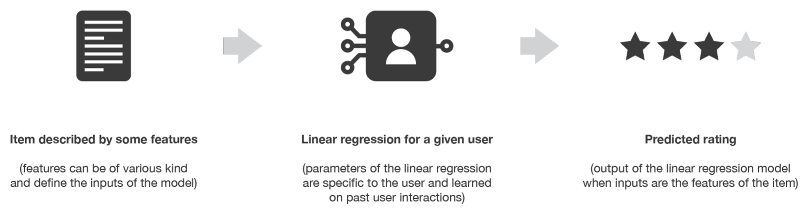

Item described by some features
(features can be of various kind and define the inputs of the model)
Linear regression for a given user
(parameters of the linear regression are specific to the user and learned on past user interactions)
Predicted rating
(output of the linear regression model when inputs are the features of the item)

Figure 13: User-Centred Linear Regression

This is an example of user-centred regression model. Model learns how to recommend by predicting the user-item couples as a number between prediction range.

For each user we want to train a simple linear regression model that takes item features as inputs and output the rating for this item.

For given user i, we have to solve this optimisation problem

$$X_i = \underset{X_i}{\operatorname{argmin}} \ \frac{1}{2} \sum_{(i,j) \in E} [(X_i)(Y_j)^T - M_{ij}]^2 + \frac{\lambda}{2} \left(\sum_k (X_{ik})^2 \right)$$

3.2 Advantages and Disadvantages of Content Based Approaches

Advantages:

The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.

The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

Disadvantages:

Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.

The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

4 Softmax DNN for Recommendation

With this approach, Deep Neural Networks can be used for recommendations. Input for this DNN model is user query (features or latent space for user). Also location info or other related informations can be added to the input. Output of the model is a softmax layer which has number of items dimension. Each node in softmax layer, represent the probability of user liking the item.

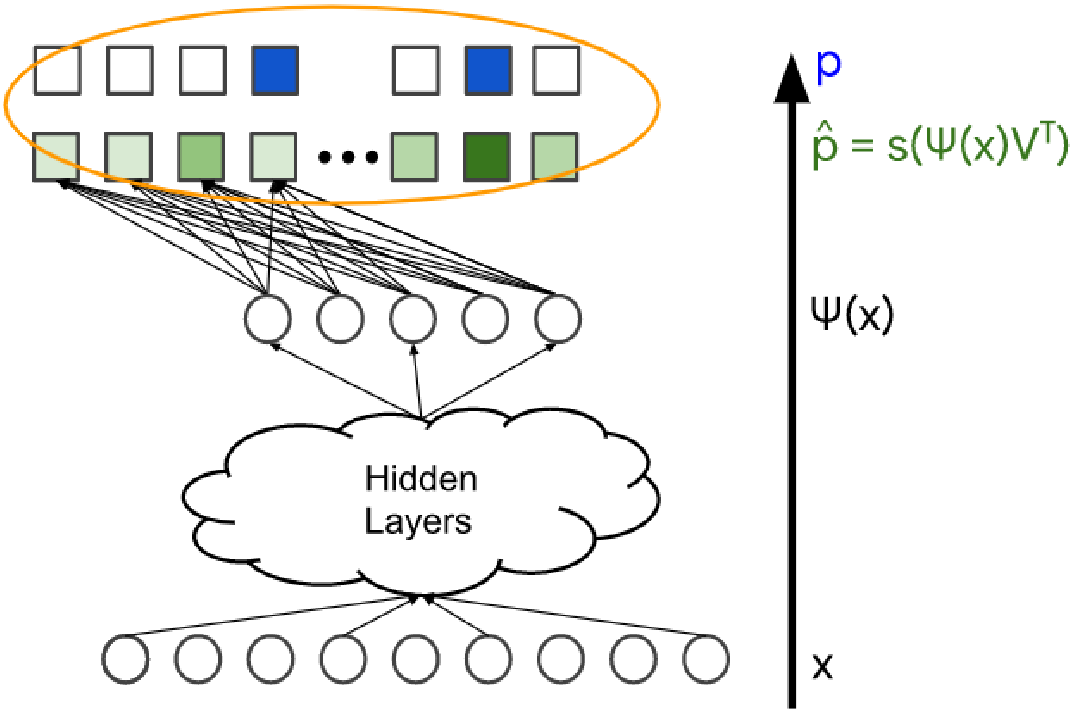


Figure 14: Softmax DNN Approach

Item set can be extremely large in some systems. It means updating all nodes in last layer every iteration is very time consuming. Solution of this problem is **negative sampling**. Instead of using all items to compute the gradient (which can be too expensive) or using only positive items (which makes the model prone to folding), you can use negative sampling. More precisely, you compute an approximate gradient, using the following items: all positive items (the ones that appear in the target label) and a sample of negative items.

DNN Models can better capture personalized preferences, but are harder to train and more expensive to query.

DNN Models are preferable to matrix factorization for scoring because DNN models can use more features to better capture relevance.

In large systems, usually matrix factorisation is used because it is more scalable and cheaper to query.

5 Specialized Recommendation Systems

There are many kind of recommendation systems that specialized for one purpose. Their usage areas are not that wide as classical recommendations systems.

- **Session Based Recommender Systems:** They are useful when history of a user is not available or not relevant in current user session. Most instances of session-based recommender systems rely on the sequence of recent interactions within a session without requiring any additional details of the user. Techniques for session-based recommendations are mainly based on generative sequential models such as Recurrent Neural Networks, Transformers and other deep learning based approaches.
- **Reinforcement Learning for Recommender Systems:** This method depends on punishment and rewarding system according to users actions while training.
- **Multi-Criteria Recommender Systems:** This can be defined as recommender systems that incorporate preference information on multiple criteria.
- **Risk-Aware Recommender Systems:** Time that system sends notification to users, is as important as what system recommended. Risk-Aware RS takes into consideration also that time.
- **Mobile Recommender Systems (Location Based):** Make recommendation according to the location of user or general users of this area.
- **Knowledge Based Recommender Systems:**
  Constraint-Based RS are also Knowledge Based, they use filters while recommend items to users like upper-bound for sale price of a house, model limitation of a car...
  Case-Based RS receive an item and retrieve the similar items to it.
  Conversational Systems iteratively in the context of a feedback loop so feedback is the knowledge for them.
  Search-Based Systems depends on preset sequence of questions like what kind of house do you search.
  Navigation-Based RS recommends by looking at the location. Example: "I would like a similar house about 5 miles west of the currently recommended house."
- **Time-Sensitive RS:** Do not recommend surf board in winter.
- **Utility-Based RS:** This kind of systems use prior utility function, a utility function is defined on the product features in order to compute the probability of a user liking the item, before recommending the item.
- **Social Recommender Systems:** Used in web-enabled networks, recommend by looking at tags or mentions. For example, Twitter has a new feature that recommends topics to users.
- **Trustworthy Recommender Systems:** User-based neighborhood method should be computed with the use of trustworthy peers to obtain robust recommendations.
- **Attack-Resistant Recommender Systems:** Protect the system from shilling attacks.
- **Group Recommender Systems:** Recommend a group, not one user. These systems are used in gyms or cafes.

While recommending items to users, system does not have to look at one recommender system's result. Geographic features, user features, matrix factorisation results, popular trending items or a social graph like your friends' favorite items can be combined.

6 Evaluation of Recommender Systems

We need to evaluate our system for knowing how much accurate or robust. Evaluation can be done with two ways: **Metric Based** and **Human Judgment**.

6.1 Metric Based Evaluation

If the model of the system based on numbers like Linear Regression, metrics like Mean Squared Error can be used for evaluation.

If the model is classification model, metrics like accuracy, precision, recall can be used. Train and test split must done carefully with these metrics.

6.2 Human Based Evaluation

It is all about feed-backs from user. The system can be evaluated observing users' experiences.

The system should not recommend too similar items to users. If the system recommends movies for example, we do not want to see only Matrix II or Matrix III after watching Matrix I. The system should be able to create wide **"confinement area"**. The system should diverse its recommendations, that means it should has **"serendipity"**.

Explainability is also an important metric for human based evaluation because users want to know that why is this item recommended to me.

Recommender System must be updated. In real conditions, it should not rely much on old user-item interactions or old recommendations.

Evaluation methods can be divided also online and offline methods.

6.3 Online Methods

Online methods (also called A/B testing) involved users' reactions to given recommendations. Click rate or conversion rate can be considered to evaluate directly. This has a drawback that you have to have a production for testing this way, also updates must be done on final production. If the system is bad, this costs negative user experience. Online methods are also slower than offline methods but this is the ideal one.

6.4 Offline Methods

Users do not involved directly with this approach. There is no need to have a final production for evaluation. Offline methods can be done with classical train-test split. (Metric Based Evaluation). Time is an important factor for this approach, system should not evaluate summer season recommendations after training with winter data.

6.5 Beyond Accuracy

Recommender persistence: In some situations, it is more effective to re-show recommendations or let users re-rate items than showing new items.

Privacy: Sometimes reviews or comments which feeds the Recommender include private information about user.

Trust can be built by a recommender system by explaining how it generates recommendations, and why it recommends an item.

7 Recommender Systems in Real Life

Recommendations are responsible for 70% of the time people spend watching videos on YouTube. Also 75% of what people are watching on Netflix comes from recommendations, according to McKinsey.

Amazon.com recommendations based on Item-Item Collaborative Filtering Approach according to Greg Linden, Brent Smith, and Jeremy York.

Spotify's Discover Weekly playlist or Release Radar playlist are algorithmically powered tools that update personal playlists on a weekly basis.

YouTube has also interactive recommendation algorithms. You can feed the algorithm just by clicking videos or liking them. Search history is also used for feeding the algorithm. YouTube allow us to control our recommendations with "not interested" button, "clear search history" option or "delete feed-backs" option.

The RS is driven by the Google Brain deep learning artificial intelligence project and is comprised of two neural networks. The first collects and collates information on users' watch history and uses collaborative filtering to select hundreds of videos. This process, known as candidate generation, uses feedback from users to train the model. The second neural network ranks the selected videos in order to make recommendations to users.
According to YouTube after implementation of the RS for more than a year, it has been successful in terms of their stated goals, with recommendations accounting for around 60 percent of video clicks from the homepage.

Incorporate item profitability: Instead of having recommendations based solely on a customer's browsing history and past purchases, this would allow businesses to control how much a profit-based recommendation differs from the traditional recommendation and to set a balance so that customer trust would not be compromised.

Reach shoppers through multiple channels: Next generation recommendation systems should be able to reach customers across a range of channels including email, social media, on an off-site shopping widgets, mobile apps, and the retail customer service centers.

8 Conclusion

Recommender Systems have important role in our lives. They must be robust, they must be thoughtful, they must understand what we would like to watch or read.

Recommender System Algorithms can be built with different approaches or different methods. Here is a summary schema for all approaches or methods in Figure 18.

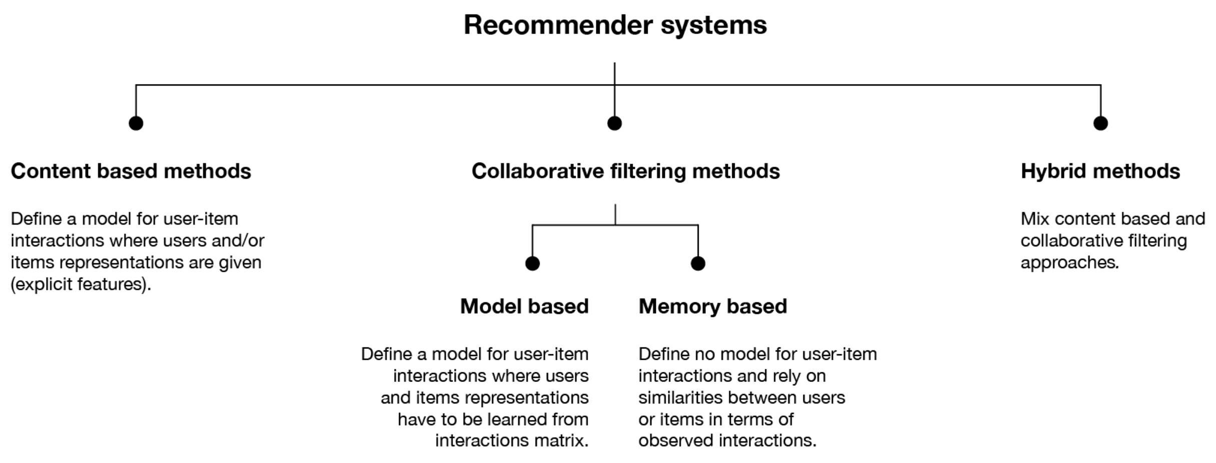

Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

Figure 15: Summary Schema

9 References

- Introduction to Recommender Systems at Towards Data Science
- Introduction to Recommender Systems at tryo.labs
- Use Cases of Recommendation Systems
- Collaborative Filtering at Wikipedia
- Recommender System at Wikipedia
- Recommender Systems Python at datacamp
- Recommendation at Google Developers
- Recommender Systems The Textbook (Aggarwal, Charu C.)
- Item2Vec: Neural Item Embedding for Collaborative Filtering
- Deep Learning Based Recommender System: A Survey and New Perspectives
- Amazon Recommendation
- The Netflix Recommender System: Algorithms, Business Value, and Innovation
- Deep Neural Networks for YouTube Recommendations