Algorithms in Recommender Systems

Summary of class presentation
(Group 5)

Modern web platforms dealing with large number of items use recommender systems to automatically suggest new interesting items to users and, hence, to keep them using the platform. From the users' perspective, recommender systems help them handle information overload. In our presentation, we discussed methods and algorithms used in recommender systems. We started with collaborative filtering, which traditionally is the most used approach, then we talked about content-based and knowledge-based recommender systems and highlighted their merits.

Collaborative filtering takes advantage of the community by using the user-item rating matrix in order to predict user's rating for an unrated item or to output a list of recommended items for a user. There are two ways of doing that: (i) user-based, by searching similar users and compute the rating for an item based on their ratings or (ii) item-based, by considering similar items and predict the rating based on their ratings. Collaborative filtering comes with some major challenges in contemporary web platforms. The naive algorithms do not scale since there may be millions of users and millions of items. Moreover, the ratings matrix is sparse and when new users or new items are introduced into the system, there is a period of time when no recommendation can be done, a phenomenon known as cold start. In addition, collaborative filtering is sensitive to attacks. Malicious users can rate high their items (push attack) or rate low competitor items (nuke attack). The attacks can be more effective when done automatically by bots. These attacks are knows as shilling attacks.

Item-based collaborative filtering helps deal with the scalability problem inherent in user-based collaborative filtering and is less affected by shilling attack. It uses similarity between items rather than similarity between users to predict user ratings. The underlying principle is that people tend to buy products similar to what they like. Item similarities not only tend to be more stable compared to user similarities but the computation of the item similarity matrix which is expensive can also be done offline. This matrix is then used to make real-time predictions. As such, the system can scale independently of the number of users or items. The approach is also less affected by shilling attack since rating prediction for an item is made by comparing its rating vector with those of other items and the attacker has no control over ratings given to any item by other users.

With respect to item-based collaborative filtering, if the number of users and items are very large (e.g., Amazon has $\sim 10^8$ users and $\sim 10^7$ items), the cost to compute the similarity matrix will be very high. There are various ways to deal with scalability problem; we present one approach, i.e. via clustering. Basically, there are two main approaches for clustering: item-clustering (clustering the items) and user-clustering. For item-clustering, we can create the clusters based on the item's type (e.g., books, electronics, etc.), and for user-clustering, we can employ clustering algorithm to create the clusters. After we have clustered the items/users, the similarity matrix is then computed using only items/users in the respective clusters. Since the number of items/users inside the clusters is smaller, the total cost to compute the matrix similarity is also reduced. In our presentation, we showed one clustering algorithm which is commonly used in collaborative filtering, i.e. modularity maximization, which is also popular in community detection problem.

There are numerous algorithms to deal with the problem of data sparsity. The most effective ones are based on matrix factorization. We have seen the matrix factorization technique which is widely known as the Simon Funk method. The rating matrix is factorized into two

lower rank matrices, one with respect to the users' characteristics and another with respect to items' characteristics. Basically, each user and item are represented by a set of k constant factors. The estimated rating of a user for a new item is obtained by multiplying the factors of the item with that of the user. Matrix factorization also resolves the problem of (i) first rater for new items, (ii) population bias, i.e. individuals with unique taste, and (iii) scalability.

The part on Collaborative filtering was concluded with the example of Google News. The evolution in the Google News recommender system from 2007 to 2010 also showed that collaborative filtering alone was not enough.

Unlike collaborative filtering that merely take advantage of similar users' rating profiles, content-based recommender systems make use of the genuine content of the items and the past preferences of a user, which is much more reliable. Content-based RS can also prevent cold start for new items. In our talk, we first introduced a high level architecture of a content-based RS, and then discussed about item representation for both structured data and unstructured data (vector space models: boolean term vector and weighted term vector). Content-based RS can be regarded as a classification problem. We discussed about two simple algorithms for content-based RS: (i) K-nearest neighbors classification algorithms; (ii) Naive probabilistic methods. Apart from a list of commonly used classification algorithms, we also talked about Rocchio relevance feedback algorithm that takes advantages of users' relevance judgments in the retrieval process. In particular, we discussed the updating process of the algorithm in detail and some examples were given. Relevance feedback algorithms would fail under two conditions: (i) the user lacking initial knowledge of the goal items; (ii) the existence of several prototypes of relevant documents.

The Knowledge Based recommender system offers a dialog that effectively walks the user down a discrimination tree of product features. It does not have a ramp-up or cold start problem since its recommendations do not depend on a base of user ratings. It is time independent, highly scalable and robust. It has a conversational style of approach. The user in this case can specify, modify and even provide explicit feedback. Based on this knowledge, the system generates appropriate recommendations. If no such item exists satisfying all the constraints, products satisfying a maximal set of constraints are showed in a rank wise manner with proper explanations as to why the recommendation was done.

Finally, we saw that recommender systems are highly application oriented and, in real world situations, different approaches are combined to form a hybrid recommender system for better results.

Presenters:

Dumitrel Loghin
Anuja Meetoo Appavoo
Suhendry Effendy
Paramasiven Appavoo
Lu Bingxin
Li Jing
Suman Sourav