Collaborative Filtering

With an Insight into Recommendation System architecture

What is Collaborative Filtering?

- Collaborative filtering (CF) is a technique commonly used to build personalized recommendations on the Web. Some popular websites that make use of the collaborative filtering technology include Amazon, Netflix, iTunes, IMDB, LastFM, Delicious and StumbleUpon.

- In collaborative filtering, algorithms are used to make automatic predictions about a user's interests by compiling preferences from several users.

Methodology used for Collaborative Filtering

- Two Subcategories

✓ **User-based collaborative filtering::**
* Look for users who share the same rating patterns with the active user (the user whom the prediction is for).
* Use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user

✓ **Item-based collaborative filtering::**
* Build an item-item matrix determining relationships between pairs of items
* Infer the tastes of the current user by examining the matrix and matching that user's data

Different types of collaborative filtering

- **Memory Based:** This method makes use of user rating information to calculate the likeness between the users or items (User based /Item based). This calculated likeness is then used to make recommendations.

- **Model Based:** Models are created by using data mining, and the system learns algorithms to look for habits according to training data. These models are then used to come up with predictions for actual data.

- **Hybrid:** Various programs combine the model-based and memory-based CF algorithms.

Memory Based Collaborative Filtering

- This approach uses user rating data to compute the similarity between users or items. This was an early approach used in many commercial systems. Typical examples of this approach are **neighbourhood-based CF** and **item-based/user-based top-N recommendations.**

- The neighborhood-based algorithm calculates the similarity between two users or items, produces a prediction for the user by taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple measures, such as Pearson correlation and vector cosine based similarity are used for this.

Calculating User similarity

Pearson Correlation similarity for two users X & Y, where I_{xy} is the set of items rated by both user x and user y.

$$\text{simil}(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2 \sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}}$$

Cosine similarity between two users X and Y used in cosine-based approach.

$$\text{simil}(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}}$$

r_u is the average rating of user u for all the items rated by u.

Calculating User similarity

Constrained Pearson Correlation: With an absolute reference, it is possible to correlate absolute like/dislike rather than relative deviation (as the standard Pearson r does.

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_z)(r_{v,i} - r_z)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - r_z)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - r_z)^2}}$$

Spearman Rank Similarity: The Spearman rank correlation coefficient is another candidate for a similarity function. For the Spearman correlation, the items a user has rated are ranked such that their highest-rated item is at rank 1 and lower rated items have higher ranks. Items with the same rating are assigned the average rank for their position. The computation is then the same as that of the Pearson correlation, except that ranks are used in place of ratings.

Calculating prediction for User-User collaborative filtering

- A prediction/recommendation for user u, user-user CF uses s to compute a neighbourhood N€U of neighbours of u. Once N has been computed, the system combines the ratings of users In N to generate predictions for user u's preference for an item i. This is typically done by computing the weighted average of the neighboring users' ratings i using similarity as the weights:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|}$$

- Subtracting the user's mean rating ū ucompensates for differences in users' use of the rating scale

$$p_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})/\sigma_{u'}}{\sum_{u' \in N} |s(u,u')|}$$

Calculating prediction for item-item collaborative filtering

After collecting a set S of items similar to i , $p_{u,I}$ can be predicted as follows:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|}$$

NOTE: *The first comes to light when it is possible for similarity scores to be negative and ratings are constrained to be nonnegative: some of the ratings averaged together to compute the prediction may be negative after weightings. While this will not affect the relative ordering of items by predicted value, it will bias the predicted values so they no longer map back to the user rating domain.*

This can be corrected either by thresholding similarities so only items with nonnegative similarities are considered or by averaging distance from the baseline predictor:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j)(r_{u,j} - b_{u,i})}{\sum_{j \in S} |s(i,j)|} + b_{u,i}$$

Overcoming the difficulties in the model

The other difficulty is with non-real-valued ratings scales, particularly the unary scales common on e-commerce sites without ratings data. In this case, the averaging does not work: if all similarities are positive and $r_{u,i} = 1$, if user u have purchased item i, the prediction always comes 1. With negative similarities, it is similarly ill behaved. To work around this, we can compute pseudo-prediction as

$$\tilde{p}_{u,i} = \sum_{j \in I_u} s(i,j)$$

This is not in a meaningful scale to predict any particular user behavior, but the predict task is typically not as important in unary contexts. This pseudo-prediction can, however, be used to rank candidate items for recommendation, forming a good basis for using item–item CF to recommend items based on user purchase histories

Overcoming the difficulties in the model

It is also possible to use weights other than the similarity function for computing the final prediction. In this, computes item weights directly by estimating, for each user–item pair u,i the solution to the linear equation **Aw=b.** The solution w is such that w_j is the optimal weight to use for u's rating of j in computing their rating of i, and A and b are given as follows:

$$a_{j,k} = \sum_{v \neq u} \pi_{v,j} \pi_{v,k}$$

$$b_j = \sum_{v \neq u} \pi_{v,j} \pi_{v,i}$$

The computed weights, differing for each user–item pair, are then used to compute the prediction

$$p_{u,i} = \sum_{j \in S} w_j r_{u,j}.$$

User and Item based CF using boolean (0-1)

Less research is available for situations where no large amount of detailed directly elicited rating data is available. However, this is a common situation and occurs when users do not want to directly reveal their preferences by rating an item. Here we only have information about which pages were viewed but not why some pages were not viewed. This situation leads to binary data or more exactly to 0-1 data where 1 means that we inferred that the user has a preference for an item and 0 means that either the user does not like the item or does not know about it.

Recommendations for 0-1 Data Based on Association Rules

The binary profile matrix R is seen as a database where each user is treated as a transaction that contains the subset of items in I with a rating of 1.

To make a recommendation for an active user Ua given the set of items Ta the user likes and the set of association rules R (dependency model), the following steps are necessary:

1. Find all matching rules $X \rightarrow Y$ for which $X \subseteq Ta$ In R.
2. Recommend N unique right-hand-sides (Y) of the matching rules with the highest confidence (or another measure of interestingness)

Evaluation of Recommender Algorithms

- **Splitting:** We can randomly assign a predefined proportion of the users to the training set and all others to the test set.
- **Bootstrap sampling:** We can sample from U test with replacement to create the training set and then use the users not in the training set as the test set. This procedure has the advantage that for smaller data sets we can create larger training set s and still have users left for testing.
- **k-fold cross-validation:** Here we split U Into k sets (called folds) of approximately the same size. Then we evaluate k times, always using one fold for testing and all other folds for leaning. The k results can be averaged. This approach makes sure that each user is at least once in the test set and the averaging produces more robust results and error estimates.

Pre-computing and Truncating the model

Pre-computation and truncation is essential to deploying collaborative filtering in practice, as it places an upper bound on the number of items which must be considered to produce a recommendation and eliminates the query-time cost of similarity computation. It comes with the small expense of reducing the number of items for which predictions can be generated (the coverage of the recommender), but the unrecommendable items will usually have low predicted preferences anyway.

Challenges of collaborative filtering

- **Data Sparsity:** Practically, recommender systems are based on large datasets. As a result, the user-item mat used for collaborative filtering could be extremely large and sparse, which brings about the challenges in th performances of the recommendation.

  Solution:

  1) Implicit rating: trying to increase the number of ratings by inferring them from the user's behavior.

  2) Dimensionality Reduction: By reducing the dimensionality of the information space, the ratings of two users can be used for predictions even if they did not rate the same items.

  3) Content Description: Using the content of an item instead of the item itself could increase the amount of information people have in common.

- **Scalability:** Clusters of machine could be a solution. Another recent development **Item2Vec** produces embedding for items in a latent space and is capable of inferring item-to-item relations even when user information is not available.

- **Synonyms:** Synonyms refers to the tendency of a number of the same or very similar items to have different names or entries.

Challenges of collaborative filtering

- **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.
- **Shilling Attacks:** In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors.
- **Diversity and the long tail:** Collaborative filters are expected to increase diversity because they help us discover new products. Some algorithms, however, may unintentionally do the opposite. Because collaborative filters recommend products based on past sales or ratings, they cannot usually recommend products with limited historical data. This can create a rich-get-richer effect for popular products, akin to positive feedback. This bias toward popularity can prevent what are otherwise better consumer-product matches.

Recommender System Softwares

Software	Description	Language
Apache Mahout	ML libraries including CF	JAVA
Cofi	CF Library	Java
Crab	Components to create recommender systesm.	Python
Easyrec	Recommender for web pages	Java
Lenskit	CF algorithms from Grouplens research	Java
MyMediaLite	Recommender System Algorithm	C#/Mono
PREA	Personalized recommendation system algo.	Java
SVD Feature	Toolkit for feature based matrix factorization	C++
Vogoo PHP Lib	CF algorithm for personalizing websites	PHP

Model Based Approach

- Models are developed using data mining, machine learning algorithms to find patterns based on training data. These are used to make predictions for real data. There are many model-based CF algorithms. These include Bayesian networks, neural embedding models, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factor, latent Dirichlet allocation and Markov decision process based models.

Hybrid

- A number of applications combines the memory-based and the model-based CF algorithms. These overcome the limitations of native CF approaches. It improves the prediction performance. Importantly, it overcomes the CF problems such as sparsity and loss of information. However, they have increased complexity and are expensive to implement. Usually most of the commercial recommender systems are hybrid, for example, Google news recommender system.

Recommender systems using collaborative filtering

- Recommender systems typically produce a list of recommendations in one of two ways – through collaborative or content-based filtering. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in. Content-based filtering approaches utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties. These approaches are often combined.

- R integration with SAP HANA is one of the very popular movie recommendation engine made by leveraging R. It is not only fast in memory database, but also embeds powerful analytical capabilities.

Resources

- Wikipedia
- http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf
- http://recommender-systems.org/collaborative-filtering/
- http://link.springer.com/chapter/10.1007/978-3-540-72079-9_9#page-1
- http://dl.acm.org/citation.cfm?id=2185828
- http://dl.acm.org/citation.cfm?id=963772