Efficient Neighbourhood Estimation for Recommenders with Large Datasets

Li-Tung Weng[*]*,Yue Xu, Yuefeng Li, Richi Nayak*
Faulty of Information Technology
Queensland University of Technology
QLD 4001, Australia
[*]*soloman1124@hotmail.com, {yue.xu, y2.li, r.nayak}@qut.edu.au*

Abstract *In this paper, we present a novel neighbourhood estimation method which is not only both memory and computation efficient but can also achieves better estimation accuracy than other cluster based neighbourhood formation techniques. In this paper we have successfully incorporated the proposed technique with a taxonomy based product recommender, and with the proposed neighbourhood formation technique both time efficiency and recommendation quality of the recommender are improved.*

Keywords Recommender Systems, Neighbourhood Estimation, Product Taxonomy

1. Introduction

Our main contribution in this paper is a novel neighbourhood estimation method called "relative distance filtering" (RDF), it is based on pre-computing a small set of relative distances between users, and using the pre-computed distances to eliminate most unnecessary similarity comparisons between users. The proposed RDF method is also capable of dynamic handling frequent data update; whenever the user preferences in the dataset are added, deleted or modified, the pre-computed structure cache can also be efficiently updated.

In this paper, we applied the proposed RDF technique to a well-known taxonomy recommender, namely taxonomy-driven product recommender (TPR), proposed by Ziegler[1]. TPR utilizes the taxonomy information of the products to solve the data sparsity and cold-start problems, and it outperforms standard collaborative filtering systems with respect to the recommendation accuracy when producing recommendations for sites with data sparsity. However, despite these benefits in TPR, the time efficiency of TPR drops significantly when dealing with huge number of users, because the user preferences in TPR are represented by very high dimensional vectors. After combining RDF with TPR, our experiment result shows that both the accuracy and efficiency of TPR are improved.

2. System Model

We envision a world with a finite set of users $U = \{u_1, u_2, \dots, u_n\}$ and a finite set of items $T = \{t_1, t_2, \dots, t_m\}$. For each user $u_i \in U$, he or she is associated with a set of corresponding implicit ratings R_i, where $R_i \subseteq T$. Unlike explicit ratings in which users are asked to supply their perceptions to items explicitly in a numeric scale, implicit ratings such as transaction histories, browsing histories, product mentions, etc., are more common and obtainable for most e-commerce sites and communities.

The TPR technique proposed by Ziegler[1] represents user profiles as taxonomy vectors, and the taxonomy vectors are constructed using users' implicit ratings. Next, TPR makes recommendations to a given target user based on the common opinion of the user's neighbourhood. For more information about the taxonomy vector construction and the TPR technique, please refer to [1].

3. Proposed Approach

In this paper, we propose a novel neighbourhood estimation method which is both memory and computation efficient. By substituting the proposed technique with the standard "best-n-neighbours" in TPR, the following two improvements are achieved:

- The computation efficiency of TPR is greatly improved.
- The recommendation quality of TPR is also improved as the impact of the "fixed n neighbours" problem has been reduced. That is, the proposed technique can help TPR locate the true neighbours for a given target user (the number of true neighbours might be smaller than n), therefore the recommendation quality can be improved as only these truly closed neighbours of the target user can be included into the computation.

3.1. Relative Distance Filtering

Forming neighbourhood for a given user $u_i \in U$ with standard "best-n-neighbours" technique involves computing the distances between u_i and all other users and selecting the top n neighbours with shortest distances to u_i. However, unless the distances between all users can be pre-computed offline or the number of users in the dataset is small, forming neighbourhood dynamically can be an expensive operation.

Clearly, for the standard neighbourhood formation technique described above, there is a significant amount of overhead in computing distances for users that are obviously far away (i.e., dissimilar users). The performance of the neighbourhood formation can be drastically improved if we exclude most of these very dissimilar users from the detailed distance computation. In the proposed RDF method, this exclusion or filtering process is achieved with a simple geometrical implication: if two points are very close to each other in a space, then their distances to a given randomly selected point in the space should be similar.

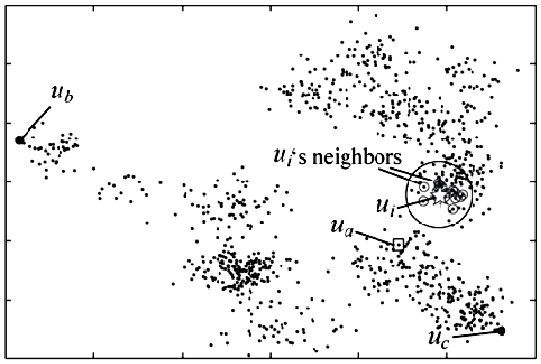

Figure 1: projected user profiles

In Figure 1, a user set U is projected onto a two-dimensional plane where each user is depicted as a dot on the plane. In the figure, u_i is the target user, and the dots embraced by small circles are the top 15 neighbours ofu_i. The RDF method starts by randomly selecting a reference user u_a in the user set, and thenu_a's distances to all other users are computed and sorted.

Based on the triangle inequality theme, it is easy to observe that all u_i's neighbours have similar distances to u_a. This means, in the process of formingu_i's neighbourhood, we only need to compute distances between u_i and the users in set U_θ^a which is defined as:

$$U_\theta^a = \{u_j \in U | u_j \neq u_i, (|\bar{a}_i - \bar{a}_j| < \vartheta)\} \quad (1)$$

where $\bar{a}_i$denotes the distance from u_a to u_i.

In equation (1), $|\bar{a}_i - \bar{a}_j|$ is the difference of the distances from u_i to u_a and u_j to u_a. ϑ is a distance threshold. If $|\bar{a}_i - \bar{a}_j|$ is larger than ϑ, the user $u_j \in U$ can be excluded from the u_i 's neighbourhood. If ϑ is set to a larger value, the distance threshold is relaxed, thus more users can be included in the neighbourhood. In this case, the performance will be decreased because more users will be included in the actual distance computations. In our experiment, ϑ is set to the one tenth of the distance between the reference user and its furthest neighbour $u_j \in U$.

By incorporating more than one reference users, the performance of RDF can be further improved. For example, by introducing two other reference users u_b and u_c, the actually search space can be reduced to $U_\theta^a \cup U_\theta^b \cup U_\theta^c$.

3.2. Proposed RDF Implementation

In the RDF implementation, the distances between users and reference users are computed offline into a data structure called RDF searching cache, and it will be loaded into the memory in the initialization stage of the online recommendation process.

In the searching cache, each user is associated with a data structure called "user node". For any user $u_j \in U$, η_j denotes u_j 's user node. A user node basically stores two types of information for a user:

1. **User ID**.
2. **Distances to the reference users**. The distances from the user node's corresponding user to the reference users are stored in a vector. In our implementation, we have only three reference users u_a, u_b an du_c, and therefore the distance vector for user node η_j is $(\bar{a}_j, \bar{b}_j, \bar{c}_j)$. We denote the distance vector of η_j as $\lambda_i = (\lambda_i^a, \lambda_i^b, \lambda_i^c)$ where λ_i^a corresponds t o$\bar{a}_j$, λ_i^b corresponds to$\bar{b}_j$and λ_i^ccorresponds to $\bar{c}_j$ respectively.

In order to efficiently retrieve the estimated searching space as described in equation (1), a binary tree structure is used to index and sort the user nodes. The index keys used for each user node are the distance between the user and the reference users, that is, the index keys for η_j are λ_i^a, λ_i^b and λ_i^c. With the three different index keys, the user nodes can be efficiently sorted with different index key settings, that is, the user nodes can be sorted by any one of the three index keys. An example of RDF searching cache is shown in Figure 2.

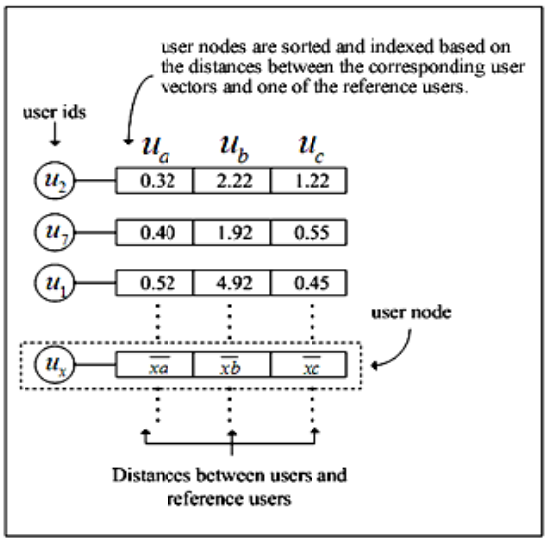

Figure 2: structure for the RDF searching cache

Given that the RDF searching cache is properly initialized, the detailed RDF procedure is described below:

RDF Algorithm

1) Let u_i be the target user, n be the pre-specified number of neighbours for u_i.

2) Use the indexed tree structure to locate the minimal user nodes set within the given boundary:

$$\xi_i = \{\eta_j | u_j \in U, (\bar{x}_i - \vartheta) < \lambda_j^x < (\bar{x}_i + \vartheta)\}$$

where $u_x \in \{u_a, u_b, u_c\}$ which achieves minimal search space. Note, the actual implementation of u_x's computation can be very efficient. By utilizing the pre-computed searching cache, the estimation of user nodes size does not involve looping through the user nodes one by one.

3) Based on step 2, u_x is the primary index key used to sort and retrieve ξ_i, and it is one of u_a, u_b and u_c. The rest two index keys (also in $\{u_a, u_b, u_c\}$) are denoted as u_y and u_z.

4) We refine the searching space ξ_i by using reference users u_y and u_z.

FOR $\eta_x \in \xi_i$ **DO**
IF $\lambda_x^y < (\bar{y}_i - \vartheta)$ **or** $\lambda_x^y > (\bar{y}_i + \vartheta)$ **or** $\lambda_x^z < (\bar{z}_i - \vartheta)$ **or** $\lambda_x^z > (\bar{z}_i + \vartheta)$
THEN remove η_x **from** ξ_i

5) Do the standard "best- n -neighbours" search against the estimated searching space ξ_i, and return the result neighbourhood for u_i.

4. Experiments and Evaluation

This section presents empirical results obtained from our experiment.

4.1. Data Acquisition

The dataset used in this experiment is the "Book-Crossing" dataset (http://www.informatik.uni-freiburg.de/~cziegler/BX/), which contains 278,858 users providing 1,149,780 ratings about 271,379 books. Because the TPR uses only implicit user ratings, therefore we further removed all explicit user ratings from the dataset and kept the remaining 716,109 implicit ratings for the experiment.

The taxonomy tree and book descriptors for our experiment are obtained from Amazon.com. Amazon.com's book classification taxonomy is tree-structured (i.e. limited to "single inheritance") and therefore is perfectly suitable to TPR.

4.2. Experiment Setup

The goal of our experiment in this paper is to compare the recommendation performance and computation efficiency between standard TPR [1] and the RDF-based TPR proposed in this paper.

The k-folding technique is applied (where k is set to 5 in our setting) for the recommendation performance evaluation. With k -folding, every user u_j's implicit rating list R_j is divided into 5 equal size portions. With these portions, one of them is selected as u_j's training set R_j^x, and the rest 4 portions are combined into a test set $T_j^x = R_j \backslash R_j^x$. Totally we have five combinations (R_j^x, T_j^x), $1 \le x \le 5$ for user u_j. In the experiment, the recommenders will use the training set R_j^x to learn u_j 's interest, and the recommendation list P_j^x generated for u_j will then be evaluated according to T_j^x. Moreover, the size for the neighbourhood formation is set to 20 and the number of items within each recommendation list is set to 20 too.

For the computation efficiency evaluation, we implemented four different versions of TPRs, each of them is equipped with different neighbourhood formation algorithms. The four TPR versions are:

- **Standard TPR:** the neighbourhood formation method is based on comparing the target user to all users in the dataset.
- **RDF based TPR:** the proposed RDF method is used to find the neighbourhood.
- **RTree based TPR:** the RTree[2] is used to find the neighbourhood. RTree is a tree structure based neighbourhood formation method, and it has been widely applied in many applications.
- **Random TPR:** this TPR forms its neighbourhood with randomly chosen users. It is used as the baseline for the recommendation quality evaluation.

The average time required by standard, RTree based and the RDF based TPRs to make a recommendation will be compared. We incrementally increase the number of users in the dataset (from 1000, 2000, 3000 until 14000), and observe how the computation times are affected by the increments.

4.3. Evaluation Metrics

In this paper, the precision and recall metric is used for the evaluation of TPR, and its formulas are listed below:

$$Recall = 100 \times (|T_j^x \cap P_j^x| / |T_j^x|) \quad (2)$$

$$Precision = 100 \times (|T_j^x \cap P_j^x| / |P_j^x|) \quad (3)$$

4.4. Result Analysis

Figure 3 shows the performance comparison between the standard TPR and the proposed RDF based TPR using the precision and recall metrics. The horizontal axis for both precision and recall charts indicates the minimum number of ratings in the user's profile. Therefore larger x-coordinates imply that fewer users are considered for the evaluation. It can be seen from the result that the proposed RDF based TPR outperformed standard TPR for both recall and precision. The result confirms that when the dissimilar users are removed from the neighbourhood, the quality of the result recommendations become better. RTree based TPR performs much worse than both the

RDF based TPR and the standard TPR, as it is unable to accurately allocate neighbours for target users

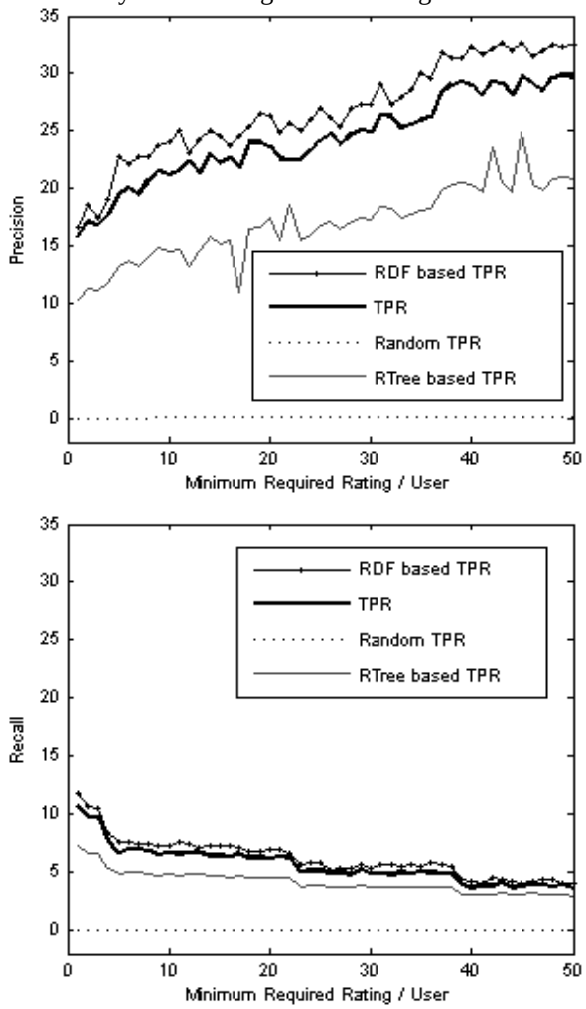

Figure 3: Recommendation precision and recall

The efficiency evaluation is shown in Figure 4.The time efficiency for standard TPR drops drastically when the number of users in the dataset increases. For dataset with 15000 users, the system needs about 14 seconds to produce a recommendation for a user, and it is not acceptable for most commercial systems. By comparison, the RDF based TPR is much efficient, and it only needs less than 4 seconds to produce a recommendation for dataset with 15000 users. The RTree based TPR greatly outperforms the proposed method when the number of users in the dataset is under 8000. However, as the number of users increases in the dataset, the differences between RDF and RTree based TPR becomes smaller, and RDF starts outperforms RTree when the number of users in the dataset is over 9000. This is because RTree is only efficient when the tree level is small. However, as the tree level increases (i.e. when number of users increases) RTree's performance drops drastically because the chance for high dimensional vector comparison increases quadratically in accordance to the number of tree level. The proposed RDF method outperforms RTree method because its indexing strategy is single value based, and it reduces the possibility for the high dimensional vector correlation computation.

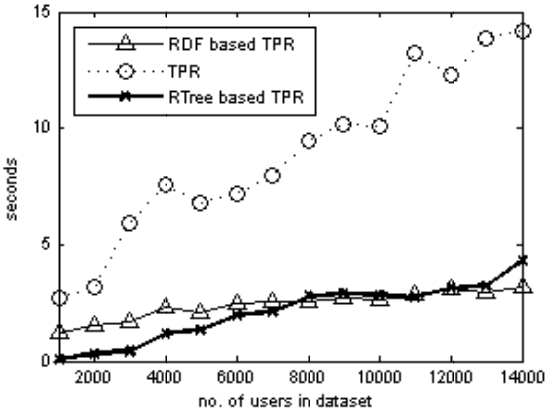

Figure 4: Average recommendation time

5. Conclusion

In this paper, we presented a novel neighbourhood estimation method for recommenders, namely RDF. By embedding RDF with a TPR based recommender, not only the computation efficiency of the system is improved, the recommendation quality is also improved. The RDF method is different from the clustering based neighbourhood formation methods that use offline computed clusters as the neighbourhoods. Instead, our method forms neighbourhood for any given target users dynamically from scratch (thus is more accurate than cluster based approaches) in an efficient manner.

In our experiment, it is shown that the proposed method improves both recommendation quality and computation efficiency for the standard TPR recommender.

References

[1] C.-N. Ziegler, G. Lausen, and L. Schmidt Thieme, "Taxonomy-driven Computation of Product Recommendations" in *International Conference on Information and Knowledge Management* Washington D.C., USA 2004.

[2] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications*: Springer, 2005.