EFFICIENT NEIGHBOURHOOD ESTIMATION FOR RECOMMENDATION MAKING

Li-Tung Weng, Yue Xu, Yuefeng Li and Richi Nayak
Faculty of Information Technology, Queensland University of Technology, 4001 Queensland, Australia

Keywords: Recommender System, Neighbourhood Formation, Taxonomic Information.

Abstract: Recommender systems produce personalized product recommendations during a live customer interaction, and they have achieved widespread success in e-commerce nowadays. For many recommender systems, especially the collaborative filtering based ones, neighbourhood formation is an essential algorithm component. Because in order for collaborative-filtering based recommender to make a recommendation, it is required to form a set of users sharing similar interests to the target user. "Best-k-neighbours" is a popular neighbourhood formation technique commonly used by recommender systems, however as tremendous growth of customers and products in recent years, the computation efficiency become one of the key challenges for recommender systems. Forming neighbourhood by going through all neighbours in the dataset is not desirable for large datasets containing million items and users. In this paper, we presented a novel neighbourhood estimation method which is both memory and computation efficient. Moreover, the proposed technique also leverages the common "fixed-n-neighbours" problem for standard "best-k-neighbours" techniques, therefore allows better recommendation quality for recommenders. We combined the proposed technique with a taxonomy-driven product recommender, and in our experiment, both time efficiency and recommendation quality of the recommender are improved.

1 INTRODUCTION

Recommender systems are designed to benefit humans' information extracting experiences by giving information recommendations according to their information needs. User based collaborative filtering is the most fundamental and widely applied recommendation technique(Schafer et al., 2000), it generates recommendations based on finding items that are commonly preferred by the neighbourhoods of the target users. Specifically, a target user's neighbourhood is a set of users sharing similar preferences to the target user(Awerbuch et al., 2005). Neighbourhood formation in collaborative filtering techniques requires comparing the target users' preferences to the preferences of all users in the dataset, and such preference comparison process can become a major computation efficiency bottleneck for recommenders. For large datasets, neighbourhood formation process requires a large amount of I/O to retrieve user profiles, and each user profile may be represented by a very high dimension vector, hence the similarity computation between the vectors can be very expensive.

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.

Part of our research is to develop a novel collaborative filtering based recommender that utilizes the item taxonomy information for its user preference representation. Our work is based on a well-known taxonomy recommender, namely taxonomy product recommender (TPR), proposed by Ziegler (Ziegler et al., 2004) which utilizes the taxonomy information of the products to solve the data sparsity and cold-start problems. TPR outperforms standard collaborative filtering systems with respect to the recommendation accuracy when producing recommendations for sites with data sparsity. However, the time efficiency of TPR drops

Weng L., Xu Y., Li Y. and Nayak R. (2008).
EFFICIENT NEIGHBOURHOOD ESTIMATION FOR RECOMMENDATION MAKING.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - SAIC*, pages 12-19
DOI: 10.5220/0001695000120019

significantly when dealing with huge number of users, because the user preferences in TPR are represented by high dimensional vectors. We applied the proposed RDF technique to the TPR and the experiment results show that by utilizing the proposed technique, both the accuracy and efficiency of TPR are significantly improved.

2 RELATED WORK

Neighbourhood formation is a process required by most collaborative filtering based recommenders to find users with similar interests to the target user. Sarwar (Sarwar et al., 2002) proposed an efficient neighbourhood selection method by pre-computing users into clusters. However, clustering is an expensive process and can only be done offline. Datasets keep changing over time. Therefore the overall quality of the result neighbourhood based on existing clusters will degrade until the next clustering update. Moreover, clustering based neighbourhood selection favours target users nearby cluster centres, and for other users located at surrounding cluster edges the quality of their result neighbourhoods are usually poor because their actual neighbours are very likely in other clusters (Sarwar et al., 2002). There are also several neighbourhood formation algorithms developed specifically for high dimensional data, such as RTree (Manolopoulos et al., 2005), kd-Tree (Bentley, 1990), etc. The basic idea behind these algorithms is to index these high dimensional data into a search tree structure, and within each level, the children nodes subdivides the cluster their parent node holds into finer clusters and each tree node holds one of the cluster spaces. The search efficiency of these algorithms is very impressive, because the search space are quadratically reduced in each tree level (i.e. O(logN)). However, they suffer from similar problems to cluster based neighbourhood search, which is "loss of precision". In fact, these algorithms usually produce worse result than clustering based method. Moreover, because the internal tree structures for indexing the data are fairly complex, therefore these algorithms are usually memory intensive and slow in initialization. The proposed RDF technique is not as good as these tree-structure based methods in terms of computation efficiency, however it is still more efficient than cluster based search method. In terms of accuracy, the proposed method produces much better result than these tree-structure based methods because it does not constrain neighbourhood search within local clusters. The internal structure of the proposed RDF technique can be updated dynamically in real time and requires only very small amount of physical memory.

3 TAXONOMY PRODUCT RECOMMENDER

An overview of taxonomy-driven product recommender (TPR) proposed by Ziegler (Ziegler et al., 2005, Ziegler et al., 2004) is given in this section.

3.1 Item Taxonomy 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, etc., are more common and obtainable for e-commerce sites and communities.

In standard collaborative filtering recommenders, user profiles are represented by m-dimensional vectors, where $m = |T|$ and each dimension represents an explicit item rating. However, for many systems, m can be very large and the number of ratings made by each user can be very small. This problem is often addressed as cold start problem or data sparsity problem.

Data sparsity problem is relieved with TPR, because instead of using the product-rating vectors with $|T|$ dimensionalities as user profiles, TPR uses taxonomy vectors with k dimensionalities, where k is the number of topics in the product taxonomy space. Specifically, we denote the taxonomy vector for u_i as $\vec{v}_i = (v_i^1, v_i^2, \dots, v_i^k)$, and each dimension of $\vec{v}_i$ indicates the degree of u_i 's interest to the corresponding topic. The taxonomy vector in TPR has three advantages over standard product rating vector. Firstly, for most e-commerce sites k is much smaller than $|T|$, and therefore it can yield better computational performances. Secondly, because the taxonomy vector records the user taxonomy preferences instead of item preference, and different items can share their descriptors entirely or partially, thus, even for users with no common item interests, their profiles can still be correlated. Thirdly, the construction of the taxonomy vector can be done

with only implicit ratings, and therefore it effectively solved the data sparsity problem.

3.2 Recommendation Generation

In this paper, the distances between user taxonomy vectors are computed by Euclidean distance, specifically:

$$dist(u_i, u_j) = \sqrt{\sum_{k=0}^{|C|}(v_i^k - v_j^k)^2} \quad (1)$$

Based on the distance measure, target user u_i 's neighbourhood clique(u_i) can be formed by selecting n users from $u_j \in U \backslash \{u_i\}$ with shortest distances to u_i. By extracting the items implicitly rated by the neighbourhood, a candidate item list is formed for u_i's personalized recommendation list, formally:

$$B_i = \cup\{R_j | u_j \in clique(u_i)\} \backslash R_i \quad (2)$$

The items in the candidate list B_i need to be ranked according to their closeness to the target user's personal interest. The ranking equation to weight u_i's possible interest towards t_k's shown below:

$$w_i(t_k) = -1 \times \left(\frac{dist(u_i, u(t_k)) \times \sum_{u_j \in A_i(t_k)} dist(u_i, u_j)}{|A_i(t_k)|}\right) \quad (3)$$

,where $A_i(t_k) = \{u_j \in clique(u_i) | t_k \in R_j\}$.

In equation (3), the computed score is negated because the proximity measure is distance based (i.e. small value indicates strong similarity), thus, by negating the result score we allow larger weight values of $w_i(t_k)$ indicating higher item interests.

$u(t_k)$ creates a dummy user for item t_k, so the proximity of the taxonomy vectors between u_i and t_k can be measured. The conversion process simply creates a user u_θ with $R_\theta = \{t_k\}$.

Finally, after the candidate item weights are computed, the top m items with highest weight values are recommended to the target user.

4 PROPOSED APPROACH

In this paper, we identified two aspects in TPR that can be improved.

Firstly, even though the product rating vectors are compressed into taxonomy vectors with smaller numbers of dimensionalities, however, for datasets with a large amount of users and extensive taxonomy structures, the neighbourhood formation will become one of the computation efficiency bottlenecks in TPR, because it requires an extensive amount of I/O to retrieve user profiles (i.e. taxonomy vectors) from the database, and the proximity computation (i.e. equation (1)) for high dimensional vectors is expensive as well.

Next, in Ziegler's TPR implementation (Ziegler et al., 2004), the "best-n-neighbours" is applied as the neighbourhood selection method since "best-n-neighbours" performs better than "correlation-threshold" for sparse dataset (Ziegler et al., 2004). However, because the value of n is pre-specified in "best-n-neighbours", it means that the resulting neighbourhoods will be biased for users with true neighbours of less than n (Li et al., 2003). This issue is particularly sensible for users with unusual tastes, as it is likely that a portion of their neighbourhoods formed by "best-n-neighbours" might contain neighbours that are dissimilar to them. For example, if a user has distinct tastes, then he or she might only share similar tastes with only 2 other users, the recommendation result for this user might be biased if a neighbourhood with 20 users are used.

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.*

4.1 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.

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 of u_i. The RDF method starts by randomly selecting a reference user u_a in the user set, and then u_a's distances to all other users are computed and sorted (u_b and u_c are also reference users).

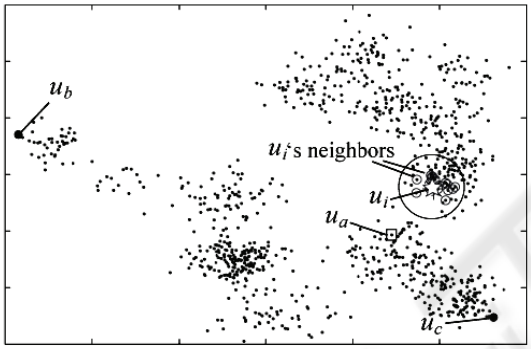

Figure 1: Projected user profiles.

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 forming u_i's neighbourhood, we only need to compute distances between u_i and the users in set U_θ^a which is defined as:

$$U_\theta^a = \{u_j \in U | u_j \neq u_i, (|\bar{a}_i - \bar{a}_i| < \vartheta)\} \quad (4)$$

where $\bar{a}_i$ is an abbreviated denotation for $dist(u_a, u_i)$.

In equation (4), $|\bar{a}_i - \bar{a}_i|$ is the difference of the distances from u_i to u_a and u_j to u_a. According to Modus tolens inference rule, i.e., if the consequent of an implication is false, the antecedent of the implication must be false, from the geometrical implication mentioned above, if $|\bar{a}_i - \bar{a}_i|$ is large, then u_i and u_j are not close to each other. ϑ is a distance threshold. If $|\bar{a}_i - \bar{a}_i|$ 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 $u_j \in U$.

To further optimize the neighborhood estimation, we can select more reference users (for example u_b and u_c) into the estimation process to obtain more estimated searching spaces (i.e. U_θ^b and U_θ^c). With multiple estimated searching spaces, the final estimated searching space can be drastically reduced by intersecting these spaces (i.e. $U_\theta^a \cap U_\theta^b \cap U_\theta^c$). It can be observed in Figure 2 that, the intersected searching space is much smaller than the entire set, and most importantly, it covers u_i's most close users. Only the users in the intersection area need to be checked for determine u_i's neighbourhood. The actual I/O and distance computations only need to be conducted within the intersected space, thus the efficiency is greatly improved.

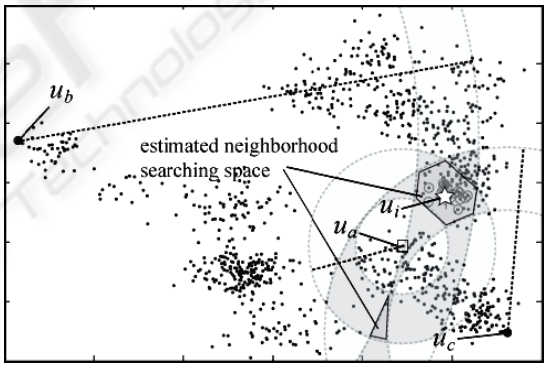

Figure 2: Estimated searching space with three reference users.

4.2 Reference User Selection

The reference user selection is important for RDF. In order to optimize the performance of TPR, the final estimated searching space (i.e.$U_\theta^a \cap U_\theta^b \cap U_\theta^c$) needs to be as small as possible for any given target users. In order to achieve it, the distances between the reference users need to be as far as possible. It is because if the reference users are close to each other, the ring borders of their search spaces will result large overlap (since they all have similar centres and radiuses). Moreover, the number of reference users should be kept small (we only use 3 reference users for all our experiments), because when the number of reference users increases, the time required for the

offline reference user initialization and the memory required for caching the sorted distances increase too.

In our implementation, the reference users are initialized with a simple two-pass technique. The first reference user u_a is chosen randomly, and we compute its distances to all other users in U. Next, with the computed distances we can obtain the second reference user u_b such that $u_b = \arg\max_{u_i \in U\{u_a\}} dist\,(u_a, u_j)$. Finally, we again find the furthest neighbour u_c for u_b and u_a such that $u_c = \arg\max_{u_i \in U\{u_b\}} dist\,(u_a, u_j) + dist(u_b, u_j)$, and set u_c as the third reference user. With this method, it is ensured that the initialization process is kept simple and efficient, and the result reference users are also very distant from each other.

4.3 Proposed RDF Implementation

This section describes in detail the implementation of the proposed RDF method discussed in sections 4.1 and 4.2. With the proposed implementation, the power of RDF is maximized.

First of all, it is important to note that the distances between users and reference users are not meant to be computed online, because the computation efficiency of this process is more expensive than the one by one search. Instead, these distances are computed, structured and indexed offline into a data structure called RDF searching cache, and the searching cache will be loaded into the memory in the initialization stage of the online recommendation process. This pre-computed searching cache is shared by all neighbourhood formation processes. The detailed structure is depicted in Figure 3.

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:** Instead of fitting the entire user profiles or the user taxonomy vector into memory, only the user id is required to be stored in the cache. The user ids are used to identify and retrieve the actual user profiles in the database.
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 and u_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 $\lambda_j = (\lambda_j^a, \lambda_j^b, \lambda_j^c)$ where λ_j^a corresponds to $\bar{a}_j$, λ_j^b corresponds to $\bar{b}_j$ and λ_j^c corresponds to $\bar{c}_j$ respectively.

In order to efficiently retrieve the estimated searching space as described in equation (4), 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_j^a, λ_j^b and λ_j^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.

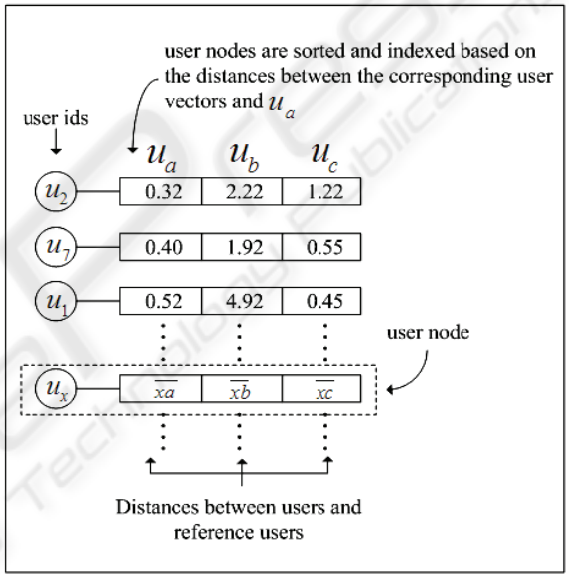

Figure 3: Structure for the RDF searching cache.

Because the user nodes are stored in this binary tree structure, the computation efficiency for equation (4) is optimized to $O(logN)$, where $N = |U|$. Note, this estimated user space retrieval process is very efficient, not only because the whole computation can be done within a small amount of memory (thus no database I/O is required), it is also because each index key lookup involves only a comparison of two double values. Finally, because distances between the target users and the reference users are needed during the neighbourhood formation process, the user profiles for the reference users are required to be stored in the cache. The memory requirement for the reference user profiles is trivial, because there are only three reference users.

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. This process is similar to finding the intersected space $U_\theta^a \cap U_\theta^b \cap U_\theta^c$ as described in section 4.2

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
  END IF
END FOR

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

5 EXPERIMENTS

This section presents empirical results obtained from our experiment.

5.1 Experiment Setup

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 goal of our experiment in this paper is to compare the recommendation performance and computation efficiency between standard TPR (Ziegler et al., 2004) 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 (Manolopoulos et al., 2005) 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.

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

$$Recall = 100 \times (|T_j^x \cap P_j^x| / |T_j^x|) \quad (5)$$

$$Precision = 100 \times (|T_j^x \cap P_j^x| / |P_j^x|) \quad (6)$$

5.2 Result Analysis

Figure 4 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 (i.e. $|R_j|$). Therefore larger x-coordinates imply that fewer users are considered for the evaluation. It can be observed 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.

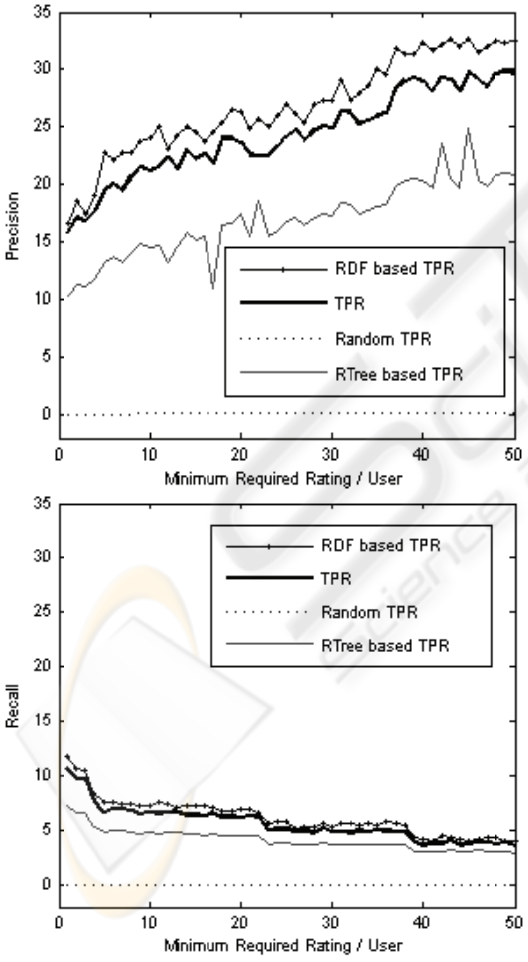

Figure 4: Recommendation precision and recall.

The efficiency evaluation is shown in Figure 5. It can be seen from Figure 5 that, 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.

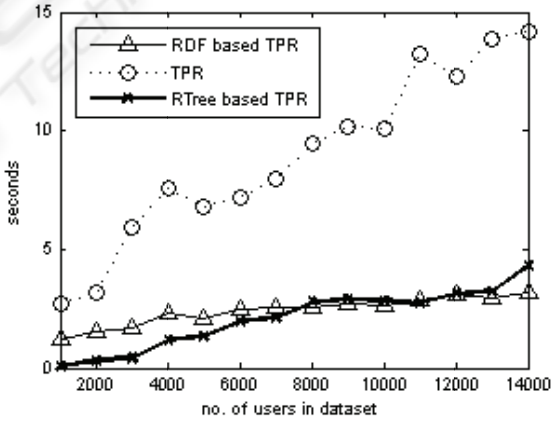

Figure 5: Average recommendation time.

6 CONCLUSIONS

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

awerbuch, B., Patt-Shamir, B., Peleg, D. & Tuttle, M. (2005) Improved recommendation systems. *Proceedings of 16th Annual ACM-SIAM symposium on Discrete algorithms.* Vancouver, British Columbia.

Bentley, J. L. (1990) K-d Trees for Semidynamic Point Sets. *6th Annual Symposium on Computational Geometry* Berkley, California, United States, ACM Press.

Li, B., Yu, S. & Lu, Q. (2003) An Improved k-Nearest Neighbor Algorithm for Text Categorization. *Proceedings of the 20th International Conference on Computer Processing of Oriental Languages.* Shenyang, China.

Manolopoulos, Y., Nanopoulos, A., Papadopoulos, A. N. & Theodoridis, Y. (2005) *R-Trees: Theory and Applications*, Springer.

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2002) Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of 5th International Conference on Computer and Information Technology.*

Schafer, J. B., Konstan, J. A. & Riedl, J. (2000) E-Commerce Recommendation Applications. *Journal of Data Mining and Knowledge Discovery,* 5**,** 115-152.

Ziegler, C.-N., Lausen, G. & Schmidt-Thieme, L. (2004) Taxonomy-driven Computation of Product Recommendations *International Conference on Information and Knowledge Management* Washington D.C., USA

Ziegler, C.-N., Mcnee, S. M., Konstan, J. A. & Lausen, G. (2005) Improving Recommendation Lists Through Topic Diversification. *Proceedings of 14th International World Wide Web Conference.* Chiba, Japan.