International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-4 Issue-13 December-2013

Generating Trust Based Recommendations for Multiple Independent Domains

Harmeet Kaur[1], Deepali Jain[2]

Abstract

Recommender systems generate responses and suggest items in the required domain. This paper proposes a domain independent trust based recommender system where personalized recommendations can be generated for any domain known to the recommenders. In this recommender system, there exists a web of trust which is formed on the basis of trust among agents in application domain. Here each user captures his likes and tastes for various domains in the form of upper or lower bound of each attribute of items belonging to domain under consideration and stores this information in his profile which then forwarded to his recommender agents. The recommender agents pass on only those recommendations to the user agent that matches its tastes leading to the personalization of recommendations. Experiments were conducted on a real dataset and results illustrate the proficiency and effectiveness of the proposed method.

Keywords

Trust, Domain independent, fuzzy sets

1. Introduction

The growing popularity and usage of web-based systems has inundated modern consumers with choices [1]. In this situation users are not able to assimilate all the available information, and therefore, they need to spend too much time surfing among thousands of products or pages until they find an interesting one that meets their necessities [2]. To get rid of this inconvenience, many tools came into existence. The most successful ones are the Recommender Systems [3], [4], [5], [6]. Precise and accurate recommender systems provide aid to users who often find himself in a large pool of information.

Harmeet Kaur, Department of Computer Science, Hansraj College, University of Delhi, Delhi, India.
Deepali Jain, Department of Computer Science, Hansraj College, University of Delhi, Delhi, India.

Most widely used recommendation systems are either content-based or collaborative filtering methods (CF). The first one tries to suggest to items similar to the ones user liked in the past. To do this, it needs a representation of the items in term of features. In CF approach, the items that users similar to the target user liked are recommended to the target user [**7**]. A disadvantage of content based methods is that they will continue to recommend related items only, and do not explore other interests. Whereas recommender systems based on CF suffer some inherent weaknesses that are intrinsic in the process of finding similar users. They offer poor prediction when the number of similar users is limited.

Moreover, these systems do not take account of how people seek recommendations from their social network of known individuals. Sinha et al. [8] note that people prefer recommendations from friends rather than general recommendation systems and further show in [9] that users prefer recommendations from systems they trust. Thus, trust must be treated as an important ingredient in the process of generating recommendations.

This paper proposes a trust-enhanced recommender system that generates trustworthy responses by mining the trust network referred to as Web of Trust (WoT) among its users. In WoT each user is represented by his agent. WoT is a virtual community of agents where agents interact and cooperate with each other in order to find valuable information for their human users. The agent uses its level of trust on the recommenders to decide which recommendation to accept. One important feature of trust is the vagueness associated while assigning trust to other party. To capture this attribute of trust, presented trust based recommender system uses fuzzy approach and employs degree of trust instead of absolute value of trust, where this degree ranges from 0 to 1.

Proposed trust enabled recommender system asks its users about their preferences in various domains. On the basis of these preferences, a set of favorites for each domain is populated with preferred upper/lower bound of each attribute of the items belonging to that domain. All these sets are clubbed together and captured in user's profiles which are then distributed

among user's recommenders. In this manner recommenders have fair knowledge of user's tastes and likes in various domains which helps them to generate personalized recommendations in various diversified domains. Recommenders on the basis of the set of favourites provided by information seeker and their judgment will generate list of items which may be liked by the information seeker. This list is known as recommendation list, which is later on aggregated by the user according to his taste and degree of trust on each recommender.

Main contributions of the paper are summarized as follows:

(1) This paper proposes a domain independent technique for generation of and accumulation of trust based recommendations.
(2) Some attributes may positively influence the aggregate factor and some negatively. For example, if crime rate is an attribute to determine how safe a city is then a higher value of crime rate negatively influences safety. In this paper, this direct or indirect proportionality of the attributes to the aggregate factor that is used to decide whether an object should be recommended or not is also taken into consideration.

Organization of this paper is as follows. Section 2presents related study and then section 3 discusses some preliminary details of Web of Trust. Initialization and maintenance of set of favourites is discussed in Section 4. Section 5 discusses the process of generation of recommendation lists and response accumulation procedure. Experimentation and results hence obtained are reported in section 6. Finally, Section 7 concludes the paper and presents some directions for future work.

2. Related Study

Recommendation techniques have been studied for years. Such techniques are often used in domains where widespread content is available, such as for recommending websites, movies, books, and news articles. In 1996,Pazzani et al. [6]asked users to rate websites and then from the ratings created profiles for individual users using which websites were then recommended according to how well their contents matched the user profiles. In 2000, Bradley, Rafter and Smyth [10], proposed a system specifically in the context of a job-seeking site. They presented a system that filters search results by comparing these to a user profile based on jobs user has previously viewed and rated [10]. Major disadvantage with this approach is that, the method requires extensive and ongoing input from the user. In 2003, Jeh and Widom [11]presented the procedure where the user must actively view and rate job advertisements in order to receive personalized results. Again main weakness is that the system is domain-specific; although advantage of the method is that it could be extended to allow the capture of viewing and rating data for any amount of items. In2006 ,J Golbeck and J Hendler in [12] proposed a Film Trust website, where trust in social networks has been used to create predictive rating recommendations for movies domain, which once again emphasis its domain specific approach. In 2004, Chiang et al.[13] developed a more sophisticated news article recommender system by using a hierarchal mixture model which generates recommendations for news articles. In 2011, Yijie Hu [14]have proposed an approach of recommending songs one by one based on user behavior. The approach considered genre, recording year, favor and time pattern as factors to recommend songs, but is applicable only in the case of recommending songs. In 2013, Preeti, Umesh[15]proposed a stock market portfolio recommender system based on association rule mining (ARM) that analyzes stock data and suggests a ranked basket of stocks. A major disadvantage of the proposed system is that it can be used only to predict stock market information. The above mentioned techniques have employed domain dependent methods for generation of recommendations whereas this paper proposes a trust based recommender system which includes a domain independent procedure to produce responses for varied domains. Not only the presented method can be used for various domains known to its users but information regarding new domains can also be appended and retrieved whenever required.

3. Web of Trust

Web of Trust (WoT) an integral part of trust based recommender systems is a trust network of agents where nodes represent agents and an edge from agent a_i to agent a_j corresponds to trust between these agents. It is a virtual community of agents where each agent is associated with a human user and these agents collaborate with each other to find valuable information for their human users [16]. The goal of a trust based recommendation system is to generate personalized recommendations by accumulating the opinions of users in their trust network. In WoT each

agent is connected to a number of agents in web of trust which forms its neighborhood [17].

WoT emerges from the real world scenario where human users wants to set up a new social networking system or to be part of an existing one. Such users create their agent and initialize their profiles in these agents so that on the basis of these user's profiles and interpersonal trust among agents, they can retrieve personalized responses for their users. For explanation purpose, consider an example where users from 1 to m (m = number of users) wish to join a social network (here WoT) as shown in figure 1 (a). All the human users who want to be associated with WoT create their agents as illustrated in figure1 (b). Assume in real life user 1 knows user 5 and user 7, and it has some degree of trust on 5 denoted by t_{15} and degree of trust on 7 as t_{17} as depicted in figure 1 (c). Transferring this information from real world to virtual network one can obtain WoT shown in figure 1 (d). Continuing in this manner all other edges and their weights can be obtained [17].

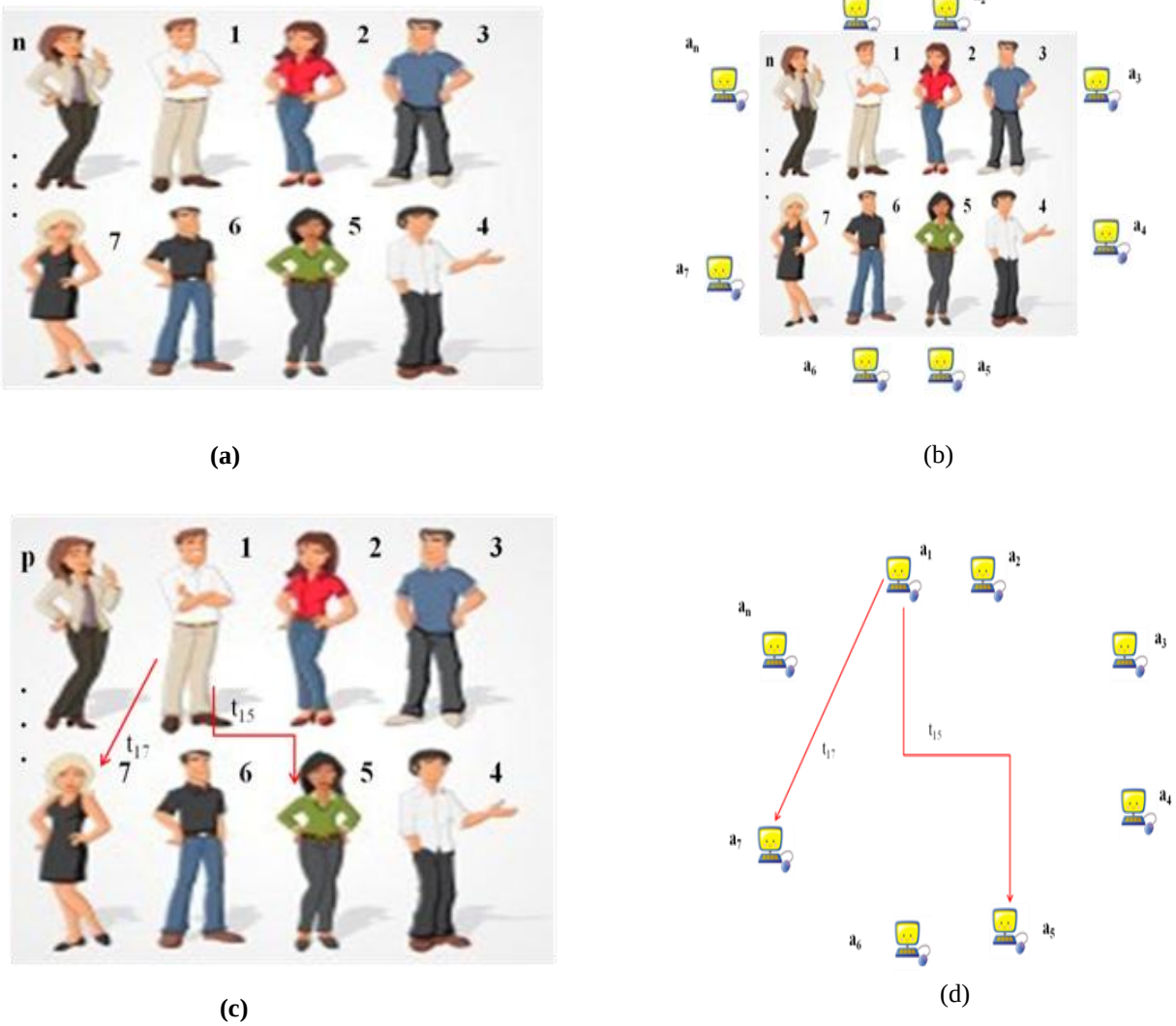

Figure 1. Sequence of steps of formation of WoT

Figure 2 depicts a sample web of trust where nodes symbolizing agents are connected through directed edges. Presence of the directed edge from agent a_i to agent a_j furnishes the information that a_i trusts a_j and the weight of this edge that is $t_{ij}$is the degree of trust from a_i to a_j which stands for the extent to which a_i trusts a_j to give good and useful recommendations.

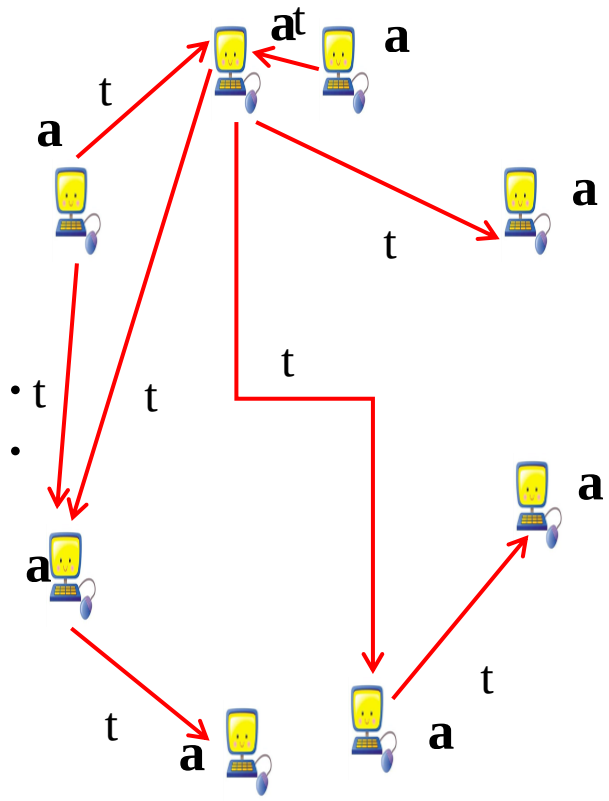

Figure 2. Web of Trust

Thus Web of Trust (WoT) can be viewed as a directed graph where:

- Agents are represented by nodes of the graph.
- Directed link from source vertex to the target vertex represents the fact that agent associated with source trusts agent linked to the target vertex.
- Weights of edges of the directed graph are annotated with the degree of trust from source to target, where this degree ranges from 0 to 1(taking trust as fuzzy value)

There is no role of centralized authority in web of trust to maintain data repository and performing calculations to generate and process recommendations. Each agent is responsible for maintaining its data and carrying out computations to generate and aggregate recommendations.

4. Sets of Favourites

4.1 Generation of Sets of Favourites

At the time of creation of an agent, human user linked with that agent prepares his profile, populates various sets of favourites for different domains according to his likes and tastes and assigns this profile to his agent. Sets of Favourites embedded in user profile captures user's taste which aids in generation of personalized responses. The recommendations are useful to a user if they are custom-made according to the taste of the user [18]. Inclusion of list of favourites in user profile facilitates the process of generating tailor made results.

The model used to describe the generation and maintenance of sets of favourites has following constructs:

- Set D is a universal set of domains, where each domain is defined on number of parameters which describes that domain.
- For each agent a_i,D(i) is a set of domains, containing all those domains for which user associated with agent $a_i$has expressed his likes and interest, D(i) is subset of D. Agents can include more domains in this set whenever recommendations are required for some other domain.
- M(i) = Number of elements in D(i).
- $D(i)_j$ stands for j^{th}element of D(i), thus D(i) = $\{D(i)_j | j= 1 \text{to } M(i)\}$.
- $M(i)_j$ denotes number of parameters in $D(i)_j^{th}$domain.
- $D(i)_{jk}$ symbolizes k^{th} parameter of $D(i)_j^{th}$domain, k =1 to M(i)j.
- $F(i)_j$,represents a set associated with $D(i)_j^{th}$domain. It is a set of favourites for $D(i)_j^{th}$domain, containing $M(i)_j$ number of elements. Each element is an ordered pair, where in k^{th} pair first member is $D(i)_{jk}$ that is k^{th} parameter of D(i)jth domain and second member signifies its lower bound or upper bound (according to the nature of the parameter) as given by the user. Distinction between lower and upper bound is provided by the sign associated with the boundary values, values describing upper bounds carries negative sign where as positive sign appears with lower boundary values.

Thus, if $D(i)_j^{th}$ domain has $M(i)_j$ number of attributes on which items belonging to this domain can be described then there will be $M(i)_j$ number of pairs in set $F(i)_j$,
Here,
$F(i)_j$= {(x, L(x)/-U(x)) | x is a parameter of $D(i)_j^{th}$ domain}, x = 1 to $M(i)_j$ (1)
L(x)/U(x) is the lower/upper bound of the attribute x by the user which represents the minimum or maximum value of the attribute x in the item perceived to be liked by the user.

For the purpose of explanation, consider

- D the universal set of domains.
- For agent a_i, assume D(i) = {movies, jokes, apartments, books}.
- M(i) = 4.
- $D(i)_1$ = movies, $D(i)_2$ = jokes, $D(i)_3$ = apartments, $D(i)_4$ = books.
- For apartment domain $D(i)_3$, $M(i)_3$ = 7 as in apartment domain users recommend apartments to each other to live in which are judged on seven basic parameters like Parking, Maintenance, Construction, Noise, Grounds, Safety and Office Staff.
- $D(i)_{j1}$ = Parking, $D(i)_{j2}$ = Maintenance, $D(i)_{j3}$ = Construction, $D(i)_{j4}$ = Noise, $D(i)_{j5}$ = Grounds, $D(i)_{j6}$ = Safety, and $D(i)_{j7}$ = Office Staff.
- Thus for apartment domain, Set of favourite $F(i)_j$ is a set of pairs, shown in table 1 where first row depicts first element of each pair and second row provides corresponding boundary values with appropriate signs. These parameters with their lower or upper bound are furnished by human user associated with agent a_i.

Table 1. Set $F(i)_j$

Parameters	Boundary values
Parking	0.534974
Maintenance	0.7708025
Construction	0.4660804
Noise	-0.4879519
Grounds	0.8288427
Safety	0.292887
Office Staff	0.2763541

Each Boundary value ∈ [-1, 1] and interpretation of values in this range is according to the parameter, for example for parking user has provided lower bound, where
parking value = 0 => Parking is a total nightmare
parking value = 1 => Parking is ample, well-lit, and secure.
An intermediate value represents cases between the extremes mentioned above.
Thus user chooses this boundary value according to his taste and convenience and wants others to recommend items having parameter's values higher than lower bound and less than upper bound.Each apartment 'A' known to the recommender is judged on the basis of values of its attributes. (1)

For each item I to be recommended, the recommender agent calculates the likeness index which is the cumulative effect of the attributes taking into consideration whether an attribute has a direct or indirect influence. This is explained in Section 5.1.

4.2 Maintenance of Set of Favourites

According to the changing needs of the human users, agents can add or delete domains from their set of domains. Moreover with the change in users taste the boundary values in the sets of favourites can also be updated easily. In case of change in the sets of favourites, the updated profile is given to all the neighbors of the agent as and when there is a change. This allows the user to propagate the changes in its taste. As an example, consider the books domain. One might prefer the books of a particular genre, but over a period of time many new authors and subjects come up and he may also like their work. But if the profile is not adapted according to the new information, then it is not possible to recommend books of new variety. Thus maintenance of set of favourites includes capturing updated tastes of users and propagating these changes to their neighbors. This change in taste can be of following types:

- Addition or deletion of domains from set of domains.
- Changing number of parameters in domain.
- Updating degree of preference for the attribute.

5. Generation and Accumulation of Recommendation lists

For further discussion, an agent a_i who seeks recommendations will be termed as *source* and its neighbors will generate list of recommendations which will be accumulated by the source. Source has m number of neighbors as listed in table 2 along with their degree of trust.

Table 2.The degree of trust on the neighbours maintained by the source

Neighbours	1	2	.	.	m
Degree of Trust	DoT_1	DoT_2	.	.	DoT_m

Assume one specific domain say D(i)j, where products under consideration are defined on T number attributes. Here T denotes $M(i)_j$ that is number of parameters in $D(i)_j^{th}$domain. For such cases set of favourites as populated by the source (say agent a_i) and maintained by its neighbors is as shown in table 3.

Table3. Set of Favourites$F(i)_j$ for $D(i)_j^{th}$ domain maintained neighbors of the source

Parameters	$D(i)_{j1}$	$D(i)_{j2}$	.	.	$D(i)_{jT}$
Boundary values	$BV(i)_{j1}$	$BV(i)_{j2}$	.	.	$BV(i)_{jT}$

$BV(i)_{jk}$ represents boundary value for the k^{th} parameter of j^{th} domain ($D(i)_j$) in the profile of agent a_i. Neighbours of the source captures source's profile which contains set of favourite for the concerned domain and generate recommendations by comparing values of parameters of items known to them and boundary values as listed by the source, hence producing list of items whose parameter values are close to the likeliness of the source. When source agent say a_i wishes to get recommendations for items of $D(i)_j^{th}$ domain from agents in its neighborhood, it carries out the process of obtaining recommendations. Procedure of obtaining recommendations involves two main steps:

1. eneration of personalized recommendation lists by neighbours of source agent.
2. cumulation of responses from neighbors.

These steps are discussed in detail in the following sub-sections.

5.1 Generating Recommendation lists

As a part of the process of generation of recommendations, the source prepares a request with the following 4-tuple query
<request_id, domain, trust_threshold_neighbour, likeliness_threshold>
where

- **request _id** is the unique identification number of the request,

domainspecifies the particular area in which search for an item has to be carried out,

trust_threshold_neighbour defines the minimum value of trust in a neighbour so that the request can be propagated to that neighbour,

source'sneighbours searches their list of known items from domain and computes the likeliness_index of items known to them by comparing values of parameters and boundary values of parameters as stated by the source. Any item for which likeliness_index comes out to be greater than **likeliness_threshold**,will be included in recommendation list prepared by source neighbours.

Source agent a_i prepares the request and finds the trust t_{ip} (degree of trust from agent a_i on agent a_p) on all its neighbouring agents a_p. For all the neighbouring agents a_p such that t_{ip}>trust_threshold_neighbour, $a_i$sends a request to generate recommendation list. When a neighbouring agent a_p of $a_i$receives a request in the form of a 4-tuple from the source, it undertakes steps as outlined in Algorithm 1.

Algorithm 1: Recommendation Generation

1. $a_p$searches its database for given domain, let d be this domain
2. If such domain d exists then
   - 2.1 For each item I in d do the following
     - 2.1.1 LI = 0 (2)
       LI is likeliness_index of an item I
     - 2.1.2 For each k^{th} parameter of I, its value is V_k, and $a_p$retrieves its value boundary value $BV(i)_{jk}$
       - 1.1.2.1 $a_p$compares values of parameters of item I and corresponding boundary values and computes

     2.1.2.2 if ($BV(i)_{jk}$< 0) then
     LI = LI +$|BV(i)_{jk}|$-V_k (3)
     else
     LI = LI +V_k - $|BV(i)_{jk}|$(4)

     2.1.3 $LI = \frac{LI}{\text{number of parameters in domain d}}$ (5)

     2.1.4 If LI>likeliness_threshold, include I in the list of recommended items
   - 2.2 Sort the list of recommended items according to the increasing order of likeliness_index.
   - 2.3 Send response as <a_p, recommendation_list> to the sender of the request, a_i
3. Else send response as <a_p, nil_list> to the sender of the request, a_i

Here likeliness_index (LI) will provide the amount of correspondence between the suggested item and the item favored by the source.

5.2 Accumulating Recommendation lists

The source receives the ordered recommendation lists from its neighboring agents. The lists consist of the likeliness index of the products. Source, computes the degree of significance (DoSIG) of the all the distinct products in the recommendation and for all the items for which DoSIG exceed the recommendation threshold (cut-off set by the human associated with the source agent) are then suggested to human.

If a recommender agent is unable to find items of the required domain then a response having nil list is obtained which will be discarded by the source otherwise a response is a tuple of the form <sender, recommendation_list> where
sender is the one who is sending the response towards the source, and **recommendation_list** is the list containing items suggested by the sender and is ordered in descending order as per the likeness_index.

Algorithm 2 outlines the steps taken by the source when it receives all such responses from all its neighbors, where every response is of the form<a_p, recommendation_list>.

Algorithm 2: Response Accumulation

1.For each recommendation_list$a_i$computes normalized rank of each item as follows:

1.1 Normalized rank ($NR_{q,p}$) of q^{th}item in the a_p's recommendation_ list is

$$NRq,p = \frac{position\_of\_item\_in\_list\,(p)}{total\_no\_of\_items\_in\_list\,(p)}$$

(6)Where, list (p) is the recommendation_list of agent a_p.

2.Compute the degree of significance (DoSIG) of every item I_x (x^{th} item) as follows:

$$DoSIG_x = DoT(a_1) * LI_x(a_1) * NR_{x1} \cap DoT(a_2) * LI_x(a_2) * NR_{x2} \cap \ldots \cap DoT(a_k) * LI_x(a_k) * NR_{xk} \quad (7)$$

where,

$DoSIG_x$ is degree of significance of I_x,
$\cap$ is the fuzzy intersection operator,
$a_p$is the p^{th} recommender,
$DoT(a_p)$ is the degree of trust of source on a_p,
$LI_x(a_p)$ is the likeliness_index of $I_x$in the recommendation list of a_p,
NR_{xp} is the normalized position of I_x in the recommendation list of a_p,
k is the total number of recommenders who have recommended I_x.

3.For all the distinct products, I_x of step 2 if$DOSIG_x$>recommendation_threshold
then
Item I_x is recommended to the human user corresponding to the source agent
Else
I_x is not recommended

A threshold mechanism is used to govern the decision making process that determines whether a user is or is not interested in an item. Thus an item whose degree of significance (how useful that product will be for the source) exceeds recommendation_thresholdit is suggested to the source.

In this manner by utilizing algorithms 1 and 2, source can obtain personalized recommendations and human user associated with source agent is free to select any item of his choice from the accumulated list of items.

6. Experiment and Discussion

Experiments were carried out to determine the precision and recall of the set of recommendations retrieved using algorithms 1 and 2. The dataset for experiments was derived from web community of Apartmentratings.com. The data set rates thousands of apartments in USA on the seven criteria viz. Parking, Maintenance, Construction, Noise, Grounds, Safety and Office Staff. The above set of parameters describes basic features of an apartment, according to which recommender will describe the apartment and probable user will choose the apartment to live in. There is one more attribute called Overall Index which provides the overall rating of the apartment taking all the attributes and reviews under consideration. For experiments the data has been collected directly from the Apartment ratings Web site [19]. The dataset consists of approximately 500,000 raters who rated a total of almost 1000 different apartments at least once. The total numbers of reviews are around 1,000,000. Out of 500,000 raters, 10 different sets of 10 raters were chosen as a sample to study algorithms. Thus in total 100 raters

were chosen. For each set of 10 raters their corresponding 10 agents were created using JADE and profile of each user was placed in its agent. The system is implemented using Java and JADE platforms. Algorithms of recommendation generation and response accumulation are developed and implemented as Java classes and are integrated with the JADE platform. The interaction among different agents for developing trust relationships were implemented as agent behaviors. In the initial phase of the experiment for each of its 10 users their list of acquaintances along with the degree of trust that they can place on each other were generated randomly. According to these lists initial web of trust was spawned which is similar to web of trust in figure 2 but with 10 agents. Web of trust thus contains a directed edge from an agent to all the agents in its list of acquaintances weighted by the degree of trust as reported in the randomly generated list and hence become its neighbors. This was done for each set of 10 agents.

6.1 Discussion

To determine the precision and recall, simulations were carried out with for each set of 10 agents by making each agent as source. In each simulation source agent initiated the process of recommendation generation. The two metrics commonly used to evaluate the recommender systems are precision and recall. *Precision* is defined as the fraction of the selected items that are relevant to the user's needs. It measures the selection effectiveness of the system and represents the probability that the item is relevant.

$$\text{Precision} = \frac{\text{Number of relevant recommendations retrieved}}{\text{Total number of recommendations retrieved}} \quad (8)$$

Recall is defined as the ratio of the relevant items selected to the total number of relevant items available. Recall represents the probability that a relevant item will be selected.

$$\text{Recall} = \frac{\text{Number of relevant recommendations retrieved}}{\text{Total number of relevant recommendations available}} \quad (9)$$

The actual recommendations received by the user agent were considered as the total number of recommendations retrieved and relevancy of products was determined using step 3 of algorithm 2.

For each set of 10 agents experiments were carried out and their results were documented. Figure 3 illustrates the precision obtained after running experiment for the first set of 10 agents.

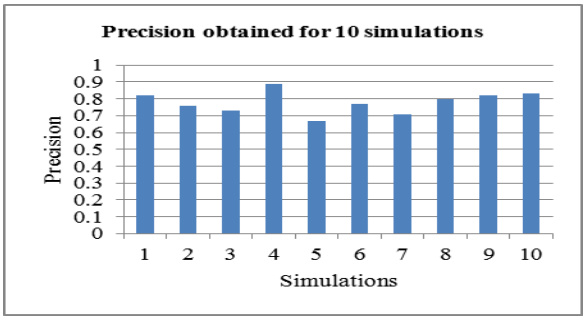

Figure 3. Precision obtained for a set of 10 simulations

For the purpose of obtaining recall, the relevant recommendations retrieved are computed in the same way as in precision and to determine total number of relevant recommendations available, all the products known to all the recommenders are considered. Figure 4 shows the recall of the recommendations for ten simulations of a set of 10 agents.

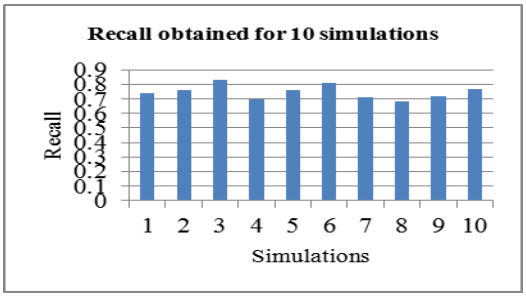

Figure 4. Recall obtained for a set of 10 simulations

As evident from figures 3 and 4 one can easily verify that proposed method of generation of trust based recommendations yield high values of recall and precision. Moreover, results of simulations of all the sets of 10 agents were recorded and average out. It was found that precision averages around 63% and recall around 68% which again proves the efficiency of proposed method. A set of 5 simulations were carried out for each set of 10 agents, to find the difference between the degree of significance (DoSIG) of apartment A_x received by the source in recommendation lists and Overall Index of A_x as reported on the site to further confirm the capability of the presented procedures. The Mean Absolute Error (MAE) which is the difference between actual and the predicted values is represented in Figure 5.

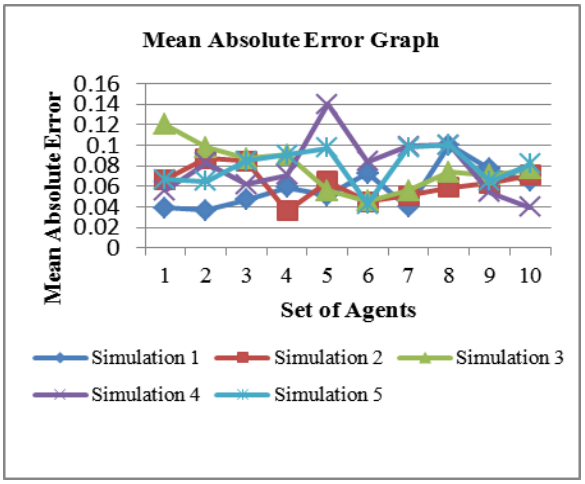

Figure 5. Mean Absolute Error Graph between the Manual and System generated recommendations

The graph in Figure 5 shows that the mean absolute error is not very large between the recommendations provided by actual reviewers in the form of Overall Index and those generated by the system, implying that the recommendations generated by the system are similar to those given by recommenders reviewing apartments.

7. Conclusion and Future Work

In this paper trust based recommender system is proposed which can generate personalized responses for products belonging to several independent domains. A recommendation generation procedure based on matching of items with the preferences of user mentioned in their profiles has been discussed. Proposed method involves recommendation generation and response accumulation algorithms to filter best suited items for users from a huge set of available items. Experiments have been performed for apartment domain and results exhibited that the proposed model has a good value of precision and recall. Same model can be applied to various domains and set of favourites of new domains can be added as and when required. More experiments are being conducted with some other real social network datasets to further validate the results.

References

[1] NadavGolbandi, Yehuda Koren and Ronny Lempel, "Adaptive Bootstrapping of Recommender Systems Using Decision Trees", International Conference on Web Search and Data Mining, Hong Kong, China, February 9–12, 2011.

[2] Luis Martinez, Luis G. Perez and Manuel J. Barranco, "Incomplete preference relations to smooth out the cold-start in Collaborative Recommender Systems", 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009), Cincinnati, Ohio, USA - June 14 - 17, 2009.

[3] ChumkiBasu, Haym Hirsh and William Cohen, "Recommendation as classification: Using social and content-based information in recommendation", fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, Menlo Park, CA, USA, American Association for Artificial Intelligence, pp. 714—720, 1998.

[4] Robin Burke, "Hybrid recommender systems: Survey and experiments", User Modeling and User-Adapted Interaction, Vol. 12, no. 4, pp. 331– 370, 2002.

[5] D. Goldberg, D.Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Communications of the ACM, vol. 35, no. 12, pp. 61 – 70, 1992.

[6] Michael J. Pazzani, Jack Muramatsu and Daniel Billsus, "Syskillwebert: Identifying interesting web sites," AAAI/IAAI, Portland, USA, Vol. 1, pp. 54–61, August, 1996.

[7] Paolo Massa and Paolo Avesani, "Trust-aware recommender systems", RecSys'07, USA, 2007.

[8] R.Sinha and K. Swearingen, "Comparing recommendations made by online systems and friends", DELOS-NSF Workshop on Personalization and Recommender Systems, Dublin, Ireland, 2001.

[9] K. Swearingen and R. Sinha, "Beyond algorithms: An HCI perspective on recommender systems" ACM SIGIR 2001 Workshop on Recommender Systems, New Orleans, Louisiana, 2001.

[10] Rachael Rafter, Keith Bradley and Barry Smyth, "Adaptive Hypermedia and Adaptive Web-Based Systems" International Conference, AH 2000, Trento, Italy, August 2000.

[11] Glen Jeh and Jennifer Widom, "Scaling personalized web search", WWW '03,12th international conference on World Wide Web, New York, NY, USA, pp. 271–279, 2003.

[12] Jennifer Golbeck and James Hendler, "FilmTrust: movie recommendations using trust in web-based social networks", IEEE Consumer communications and networking conference, 2006.

[13] J. Chiang, and Y. Chen, "An intelligent news recommender agent for filtering and categorizing large volumes of text corpus", International Journal of Intelligent Systems, Vol.19, 3, pp. 201–216, March 2004.

[14] Yajie Hu and MitsunoriOgihara, "NEXTONE PLAYER: A Music Recommendation System Based on User Behavior" 12th International Music Information Retrieval Conference (ISMIR), United States, pp. 103–108, October 2011.
[15] PreetiParanjape-Voditel and UmeshDeshpande, " A stock market portfolio recommender system based on association rule mining", Appl. Soft Comput, Vol. 13(2), pp. 1055–1063, 2013.
[16] Sean.MMcNee, John Riedl and Joseph A. Konstan, "Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems", Conference on Computer-Human Interaction, Montréal, Québec, Canada pp. 1097—1101, 2006.
[17] Harmeet Kaur and Deepali Jain, "Optimizing the Number of Neighbors in Trust Based Recommender Systems", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2,pp. 230—238, July 2013.
[18] PoonamBedi and HarmeetKaur, "Trust based Personalized Recommender System" INFOCOM Journal of Computer Science, Vol. 5, N.1, pp. 19--26.2006.
[19] http://www. apartmentratings.com (as of 1May 2013).

Dr. Harmeet Kaur received her Ph.D. in Computer Science from the Department of Computer Science, University of Delhi, Delhi, India in 2007 and M.C.A from Panjab University. She is an Associate Professor in the Department of Computer Science, Hans Raj College, University of Delhi. She has about 15 years of teaching and research experience and has published more than 20 research papers in National/International Journals/Conferences. Her research interests include Multi-agent Systems, Intelligent Information Retrieval Systems, Trust and Personalization.

Ms. Deepali Jain is a Research Scholar and working as an Assistant Professor in the Department Of Computer Science, Hans Raj College, University of Delhi. She received her B.Sc(Hons) Mathematics and M.C.A degree from University of Delhi and M.Phil in computer science from Madurai Kamaraj University. Her research area is Knowledge Based Systems and is currently pursuing PhD under Dr. HarmeetKaur from Department of Computer Science, University of Delhi.