A Modified Random Walk Framework for Handling Negative Ratings and Generating Explanations

YU-CHIH CHEN, YU-SHI LIN, YU-CHUN SHEN, and SHOU-DE LIN, National Taiwan University, Taipei

The concept of random walk (RW) has been widely applied in the design of recommendation systems. RW-based approaches are effective in handling locality problem and taking extra information, such as the relationships between items or users, into consideration. However, the traditional RW-based approach has a serious limitation in handling bidirectional opinions. The propagation of positive and negative information simultaneously in a graph is nontrivial using random walk. To address the problem, this article presents a novel and efficient RW-based model that can handle both positive and negative comments with the guarantee of convergence. Furthermore, we argue that a good recommendation system should provide users not only a list of recommended items but also reasonable explanations for the decisions. Therefore, we propose a technique that generates explanations by backtracking the influential paths and subgraphs. The results of experiments on the MovieLens and Netflix datasets show that our model significantly outperforms state-of-the-art RW-based algorithms, and is capable of improving the overall performance in the ensemble with other models.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information Filtering*

General Terms: Algorithms, Experimentation, Performance

Additional Key Words and Phrases: Ranking, Collaborative Filtering, Random Walk, Integration, Explanation

ACM Reference Format:
Chen, Y.-C., Lin, Y.-S., Shen, Y.-C., and Lin, S.-D. 2013. A modified random walk framework for handling negative ratings and generating explanations. ACM Trans. Intell. Syst. Technol. 4, 1, Article 12 (January 2013), 21 pages.
DOI = 10.1145/2414425.2414437 http://doi.acm.org/10.1145/2414425.2414437

1. INTRODUCTION

The goal of a recommendation system is to provide a (usually ranked) list of items that has a good chance of being accepted by the user. The more the users believe the items are relevant to their interests, the better their perception of the recommendation system will be. Such systems have been widely applied in different domains, such as music, books and movies. Even a search engine can be regarded as a recommendation system for documents.

Authors' addresses: Y.-C. Chen, Department of Computer Science and Information Engineering, National Taiwan University, email: r97922148@ntu.deu.tw; Y.-S. Lin, Department of Computer Science and Information Engineering, National Taiwan University, email: yushi0223@gmail.com; Y.-C. Shen, Department of Computer Science and Information Engineering, National Taiwan University, email: melonclass@gmail.com; S.-D. Lin, Department of Computer Science and Information Engineering, National Taiwan University, email: sdlin@csie.ntu.edu.tw.

DOI 10.1145/2414425.2414437 http://doi.acm.org/10.1145/2414425.2414437

A recommendation system must consider the following elements.

(1) Users: $U = \{u_1 \sim u_m\}$.
(2) Items: $I = \{i_1 \sim i_n\}$.
(3) Ratings: $R = \{r_{11} \sim r_{ij}\}$, where r_{ij} is the rating of user u for item j. Thus, R is an m^*n matrix, where m is the size of the user set, and n is the size of the item set.

The goal of a recommendation system is twofold.

(1) Predict ratings. Given a data pair comprised of a user i and an item j, and some known ratings, the system predicts the unknown rating, the value of $r_{ij.}$
(2) Predict rankings. Sort items based on the inferred preference of the user.

Most recommendation systems are based on the concept of similarity. They exploit the similarity between users, the similarity between items, or the similarity between an item and a user.

Content-based recommendation systems try to analyze the characteristics of users and items to determine their relevance or similarity, and then recommend highly relevant items to the users. For example, knowing a user's preferred movie genre, a content-based recommendation system can recommend another movie of the same genre. The main problem with this type of recommendation system is that it is not always easy to infer the user's preference. Also, understanding the content of a movie is by no means a trivial task for machines.

Another popular approach is collaborative filtering (CF) [Resnick et al. 1994] [Sarwar et al. 2001] which exploits the idea that similar users will give similar items similar ratings. The assumption allows CF methods to obtain the ratings between all items and users given only a subset of them. CF methods have following potential drawbacks.

(1) Cold start. The prediction becomes unreliable for users with few rated items.
(2) It is not clear how extra relational information or knowledge can be incorporated into the CF model to improve its performance. For example, it is not clear how to tell a CF system that "these two movies have similar actors," but apparently this kind of information could be important for recommendation.
(3) In a sparse dataset, users with disjoint sets of ratings are not correlated. For example, user1 rates only item1 and item2, and user2 rates only item3 and item4. In a traditional CF framework the similarity of these two users would be zero. However, intuitively, this might not be the case if there is a third user who rated all four items, as the ratings provided by user3 can be exploited to relate user1 and user2 indirectly.

Other than CF-based methods, researchers have proposed using graph-based recommendations which transform ratings data into graphs. The graphical model considers the long-distance relationships between users and items. In practice, users not linked by common items directly, such as user1 and user2 in the previous example, can now be connected indirectly through another user on the graph. Another advantage of graph-based methods is that they naturally facilitate the usage of additional relationship information among users and items (e.g., the social relationship between users, or the same-authorship relation between items).

Among graph-based recommendation algorithms, random walk (RW) [Fouss 2007] is probably the most appealing and the most widely used. The algorithm simulates the user's random selection from some source nodes to other nodes. The rationale is that if an item node can be reached easily from a user node, then it is more likely to be relevant to the user. The graphs are usually bipartite, containing user nodes and item nodes. The probability of moving from a user node to an item node (or vice versa) is proportional to the rating of the item given by the user.

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	1	2	-1	-2		
u_2	2	2			-1	
u_3			-2	-2		1

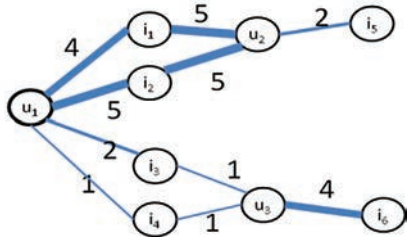

Fig. 1. A graph constructed by making all the ratings positive. The information flow to the bottom-part of the graph is smaller than that to the upper part of the graph, which causes the system to rank i_5 higher than i_6 and recommend it to u_1.

The main drawback of traditional RW-based approaches is that they cannot deal with negative ratings. Commonly used datasets, like Netflix and MovieLens, contain both positive and negative opinions (usually using rating scales range from 1 to 5 where 1 stands for 'dislike'). Generally, a negative rating indicates that the user is not impressed by an item. For recommendation systems, negative opinions are just as important as positive ones. Recommending something that users hate could damage the reputation of the system more than recommending something they may not like. Technically, negative ratings create another problem as traditional RW-based approaches require *positive* weights (or transition probabilities) on the graph to guarantee convergence to the stationary probabilities.

A popular strategy for dealing with negative impressions involves shifting the rating from negative to positive (e.g., from $[-2, -1, 0, 1, 2]$ to [1,2,3,4,5]), but it can cause other problems such as limiting the propagation of negative information. For example, in Figure 1, u_1 is similar to u_2 because they like the same items, and u_1 is also somewhat similar to u_3 because they dislike the same items. Now, item i_5 is disliked by u_2, but item i_6 is liked by u_3. Intuitively, i_6 should be a better candidate for recommendation to u_1 than i_5, which is disliked by a user similar to u_1. However, running a random walk algorithm on a weight-shifted graph (see the left part of Figure 1) shows that the upper part of the graph attracts most of the flow, which results in an unwanted higher stationary probability for i_5 than i_6. This is a serious problem because the system gives a higher preference rating to an item (i_5) that u_1 dislikes.

We attempt to solve the given problem in this article. Our contribution is twofold. First, we propose a modified random-walk method that can handle both positive and negative opinions. In our model, each node can store two types of values, one for positive information and another for negative information. During the random walk process, a node will propagate the same type of information to its neighbors if there is a positive weight between them, and the opposite information if there is a negative weight. We also provide a propagation mechanism to implement the process efficiently with a mathematical proof of its convergence.

Here we would like to emphasize that the main contribution for this article is to advance the state of the art in RW-based method, rather than designing a RW-based model to compete with other recommendation models such as the Matrix-Factorization (MF) based models or KNN-based models. In the recent years, thanks to several worldwide competitions such as NetFlix Prize and KDD Cup 2011, researchers have realized that the best strategy for recommendation requires a blending of diverse types of models [Toscher and Jahrer 2009; Chen et al. 2011; McKenzie et al. 2011]. The models that have been shown useful includes MF-based models [Koren et al. 2009]. Probabilistic latent semantic analysis models [Hofmann 2004], KNN-based models [Piotte and Chabbert 2009], supervised models, models using features created by

Restricted Boltzmann Machines (RBM) [Ackley et al. 1985], and random-walk-based models [McKenzie et al. 2011]. The advance of these models will eventually have an impact on the overall quality of recommendation results. Such "ensemble recommendation" idea changes the mindset of researchers in this area since different kinds of recommendation systems are not being viewed as "competitors" rather as "collaborators." The RW-based models are therefore useful due to the diversity they bring in to blend with other recommendation models, since their underlying mechanism is very different from that of the MF-based methods, RBM-based method, or even PLSA-based methods. Therefore, we believe our model, which significantly advances the state of the art in random-walk-based method, can be very useful for blending-based recommendation system. To verify such claim, we did additional experiments in section 5 to show that our model plays a critical role in the ensemble of different kinds of models.

Second, we believe that showing explanations of the system's ranking decisions improves users' acceptance of recommendations. Research in psychology suggests that people usually have difficulty accepting recommendations without knowing the reasoning behind the selections [Haynes 2001]. Therefore we propose a mechanism that automatically generates explanations for recommendations made by our RW-based system by identifying the dominant paths or subgraphs of information flow.

The remainder of this article is organized as follows. Section 2 contains a review of related works. We present our modified random walk approach in Section 3, and describe our explanation framework in Section 4. We report the results of experiments in Section 5, and summarize our conclusions in Section 6.

2. RELATED WORK

First, we define some notations used throughout this section.

2.1. Definitions of Notations

We denote a set of users as U, a set of items as I, and the extra knowledge related to the items as K, with $|U| = m, |I| = n, |K| = o$. The rating given by a user u to an item i is denoted by r_{ui}. Some works [Cheng et al. 2007; Zhang 2010] exploit extra knowledge such as the relatedness of item i to a knowledge item k is c_{ik}, 1 if they are related and 0 otherwise.

$R = \{r_{ui} | 1 \leq u \leq m, 1 \leq i \leq n\}$ and, $C = \{c_{ik} | 1 \leq i \leq n, 1 \leq k \leq o\}$. $\vec{r}_{u\bullet}$ and $\vec{r}_{\bullet i}$ represent the vectors comprised of user u's ratings for all items and ratings of all users for item i, respectively. In same way, $\vec{c}_{i\bullet}$ and $\vec{c}_{\bullet k}$ represent, the relations of all knowledge to an item i and the relations of all items to knowledge k, respectively. I_u is the set of items rated by user u, and U_i is the set of users who rated item i. Similarly, K_i is the set of all knowledge related to item i, and I_k is the set of all items related to knowledge k.

2.2. Random Walk

The concept of random walk bases on the PageRank [Brin and Page 1998] algorithm. The PageRank model computes the scores for web pages by simulating a user's surfing behavior. Similarly, random walk simulates the behavior of moving from a node to its neighbors on a graph by taking random steps. If a node X can be reached easily by a given source node Y, then X is deemed to be relevant to Y. As will be discussed in details later, several recommendation systems based on random walk have become popular in recent years [Craswell and Szummer 2007; Fouss 2007; Gori and Pucci 2007; Yildirim and Krishnamoorthy 2008; Liu and Yang 2009; Clements et al. 2009].

2.2.1. Random Walk on a Bipartite Graph with Rating Relations. Figure 2(a) shows a graph constructed based on the ratings that users give to items. $G = \{V, E\}$ $V = \{V_{user}, V_{item}\}$ $E = \{e_{ij} | i \in V_{user}, j \in V_{item}\}$. Fouss [2007] employs a binary selection relation which

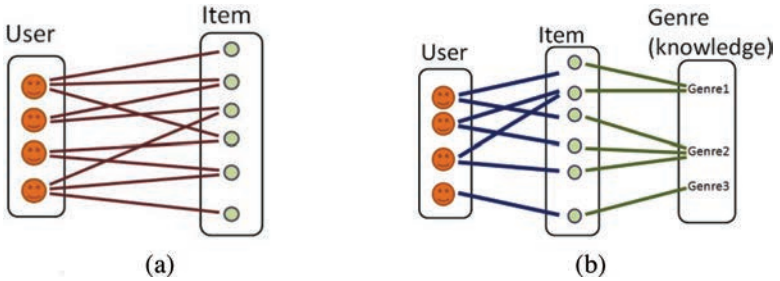

Fig. 2. (a) A graph constructed with rating information, (b) A tri-partite graph that incorporates extra genre knowledge [Cheng et al. 2007; Zhang et al. 2010].

assigns either 1 or 0 as the weight of each link. Despite this, in practice, the weights are usually assigned as integers or real values that better describe the ratings.

Like the PageRank algorithm, prior to processing, the random-walk algorithm with a certain starting point requires the weights, the transition probability between nodes, to be updated. The transition probability between node pairs is derived as follows:

$$P(i_j|u_i) = \frac{r_{ij}}{\sum\limits_{J} r_{ij}} = \frac{r_{ij}}{\sum\limits_{j\in I_i} r_{ij}} \text{ and } P(u_i|i_j) = \frac{r_{ij}}{\sum\limits_{I} r_{ij}} = \frac{r_{ij}}{\sum\limits_{i\in U_j} r_{ij}} \tag{2.1}$$

It has been shown that with nonnegative weights, the algorithm is guaranteed to converge [Athreya et al. 1996].

2.2.2. Random Walk with Extra Information. Cheng et al. [2007] posited that extra information about items could be exploited to improve the recommendation performance. The relationships can be seamlessly incorporated into graphical models by adding edges to the graph. Figure 2(b) shows an example of a tripartite graph: $G = \{V, E\}$, $V = \{V_{user}, V_{item}, V_{knowledge}\}$, $\mathrm{E} = \{e_{ij}|i \in V_{user}, j \in V_{item}, \text{ or } \mathrm{e}_{ij}|i \in V_{item}, j \in V_{knowledge}\}$.

Let $P(x)$ be the probability of staying at node x, then the initial probability distribution is $\mathrm{P}_0(\text{source user}) = 1$, $\mathrm{P}_0(\text{others}) = 0$. The transition probability then becomes:

$$P(i_j|u_i) = \frac{r_{ij}}{\sum\limits_{j\in I_i} r_{ij}}, \quad P(u_i|i_j) = \frac{r_{ij}}{\sum\limits_{i\in U_j} r_{ij} + \sum\limits_{m\in K_j} c_{jm}}$$

$$P(i_j|k_m) = \frac{c_{jm}}{\sum\limits_{j\in I_m} c_{jm}} \quad \text{and} \quad P(k_m|i_j) = \frac{c_{jm}}{\sum\limits_{i\in U_j} r_{ij} + \sum\limits_{m\in K_j} c_{jm}}. \tag{2.2}$$

Then, the update of the probability distribution will be:

$$P_{t+1}(i_j) = \sum_{i\in U_j} P(i_j|u_i)\cdot P_t(u_i) + \sum_{m\in K_i} P\left(i_j|k_m\right)\cdot P_t\left(k_m\right),$$

$$P_{t+1}(u_i) = \sum_{j\in I_i} P(u_i|i_j)\cdot P_t(i_j) \quad \text{and} \quad P_{t+1}(k_m) = \sum_{j\in I_m} P(k_m|i_j)\cdot P_t(i_j). \tag{2.3}$$

Let $\tilde{\mathrm{M}}$ be the transition matrix corresponding to the transition probability

$$P_{t+1} = \tilde{\mathrm{M}} P_t. \tag{2.4}$$

To give a penalty for longer path from the source and to let the process cope with personalized preferences, the parameter α is introduced.

$$P_{t+1} = \alpha\tilde{\mathrm{M}} P_t + (1-\alpha)\, P_0. \tag{2.5}$$

The value of α ranges from 0.15 to 0.2 will work for most cases [Craswell and Szummer 2007; Yildirim and Krishnamoorthy 2008; Liu and Yang 2009; Clements et al. 2009].

Cheng et al. demonstrated that the accuracy of recommendations improves with extra information.

2.3. Extensions of Random Walk

Several extensions of the traditional random walk approach have been proposed, such as Itemrank [Gori and Pucci 2007], similarity random walk [Yildirim and Krishnamoorthy 2008], Eigenrank [Liu and Yang 2009], and positive/negative relevance random walk [Clements et al. 2009]. We will describe them one by one in this section and compare them with our approach in Section 5.

2.3.1. Itemrank. Gori and Pucci [2007] propose this model to improve the efficiency when there are many more users than items. The graph is projected from the {user - item} space to {item-item} space. The weights of the edges between the items are proportional to the correlations of the items, while the correlation is modeled as the number of common users that have rated the items. If an item is highly correlated to the items that received high ratings from a user, then the item is a good candidate for recommendation. Given a correlation graph, it is possible to use random walk to identify items for recommendation. Mathematically, item correlations are decided by the set of common users of the items.

$$CU_{ij} = |U_i \cap U_j| \text{ and the transition probability is } P(i_j|i_k) = \frac{CU_{jk}}{\sum_J CU_{jk}}.$$

The initial probability distribution for a specific user u is defined as

$$P_0(i_k) \propto r_{uj} \quad \text{and} \quad \sum_k P_0(i_k) = 1,$$

and the probability update rule is defined as

$$P_{t+1}(i_k) = (1-\alpha) \sum_J P(i_k|i_j) \cdot P_t(i_j) + \alpha P_0(i_k).$$

The major concern with Itemrank is that information can be lost when summarizing a bipartite graph into a unipartite graph. Furthermore, the user's rating information is not exploited, as only the binary information about whether the user rated the item is used; therefore, Itemrank does not provide a good solution to negative feedbacks since the ratings will be treated equally as the positive ones. Moreover, the item-item correlation network is a nearly fully connected graph, which affects the algorithm's efficiency.

2.3.2. Similarity Random Walk. Yildirim and Krishnamoorthy [2008] propose a model similar to Itemrank which constructs a graph based on the similarity of items. If an item is similar to those rated highly by a specific user, it is considered suitable for recommendation**.** The model exploits the random walk method to capture transitive associations.

$$\text{Given an adjusted cosine similarity } S_{ij} = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_u)\left(r_{uj} - \bar{r}_u\right)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{u \in U_i \cap U_j} \left(r_{uj} - \bar{r}_u\right)^2}},$$

the transition probability becomes $P(i_j|i_k) = (1-\beta)\frac{S_{jk}}{\sum_J S_{jk}} + \beta\frac{1}{n}$.

The initial probability distribution for a specific user u is

$$P_0(i_k) \propto r_{uj} \quad \text{and} \quad \sum_k P_0(i_k) = 1.$$

The updated rule for the probability is $P_{t+1}(i_k) = (1-\alpha)\sum_J P(i_k|i_j)\cdot P_t(i_j) + \alpha P_0(i_k)$.

Similar to Itemrank, reducing a bipartite graph to a unipartite graph can cause information loss. Furthermore, the meaning of negative ratings cannot be fully exploited by this framework since the same problem stated in Figure 1 can still occur.

2.3.3. Eigenrank. Liu and Yang [2009] propose the Eigenrank model to generate the rankings of all item pairs. The rankings for unknown pairs are obtained from similar users. If an unseen item is ranked as high as or even higher than those that have already been given high ratings by a user, it can be a good candidate for recommendation to that user. To achieve this, a graph of asymmetric ranking order relations is constructed, with transition probability determined by the rankings, and the random walk approach is used to aggregate partial and incomplete rankings. The simulation of random walk algorithm on this graph would favor higher ranking neighbors.

We now list the key equations under this framework:

N_m : The set of similar users to user m

S_{mn} : The similarity between users

$$\psi_{\text{ij}} = \begin{cases} r_{mj} - r_{mi} \text{if} r_{mi}, r_{mj} \text{exists} \\ \dfrac{\sum_{n\in N_m\cap(I_i\cap I_j)} S_{mn}\cdot\left(r_{nj}-r_{ni}\right)}{\sum_{n\in N_m\cap(I_i\cap I_j)} S_{mn}} \end{cases} \quad \text{(preference of item j compared to item j)}$$

The transition probability : $P(i_j|i_i) = \dfrac{e^{\psi_{ij}}}{\sum_J e^{\psi_{ij}}}$.

The initial probability distribution for a specific user u is

$$P_0(i_k) \propto r_{uj} \quad \text{and} \quad \sum_k P_0(i_k) = 1.$$

The probability update rule is

$$P_{t+1}(i_k) = (1-\alpha)\sum_J P(i_k|i_j)\cdot P_t(i_j) + \alpha P_0(i_k).$$

One potential issue with this model is that the rankings are only obtained from similar users. The nondirect neighbor information in the graph is not exploited. Also negative ratings are used for only similarity measures, thus similar issues introduced in Figure 1 can also happen.

2.3.4. Positive and Negative Relevance Random Walks. Clements et al. [2009] observe that the traditional random walk method cannot deal with negative relevance assessments. To handle such problem, they propose to split the original graph into two graphs, one for positive-preference information and the other for negative-preference information. If a

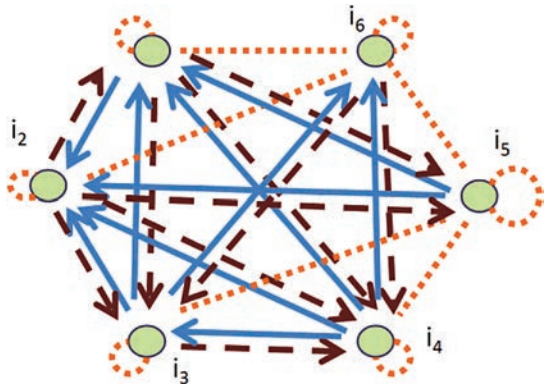

Fig. 3. The ranking relation graph for user u_1 in Figure 1. The solid line represents a "lower rating" relation; the dashed line represents a "higher rating" relation; and the dotted line means that a "rating has the same" relation. The strength of the line is roughly proportional to the exponentiation function value of the rating relation.

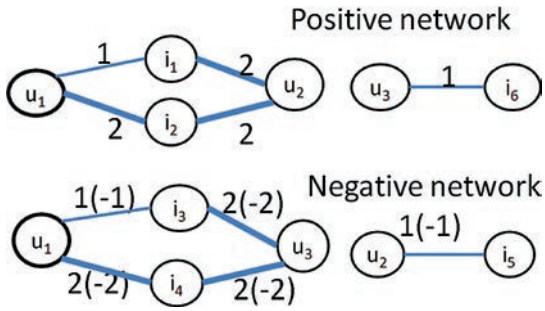

Fig. 4. The components of the graph in Figure 1 become disconnected when the original graph is split into two graphs.

user can reach an item in the positive graph easily, the item is deemed more relevant to the user. In contrast, if a user can reach an item in the negative graph easily, the user may not prefer it. Eventually, they combine the results of the two graphs to provide a single ranking value for recommendation.

In this model, positive and negative preference information is used separately. Therefore, nodes on the graph might not be fully connected, which would halt the information flow. For example, the connected graph shown in Figure 1 would be decomposed into disconnected components shown in Figure 4.

2.4. Other Models to Handle Negative Opinions

Guha et al. [2004] propose a way to measure the propagation of trust and distrust. They observe that including negative opinions will not improve the performance unless they are being propagated through positive relations. Our model mainly emphasizes on the like/dislike information, which is different from trust/distrust, as our experiment shows that our model allows dislike information to be propagated through negative links to enhance the quality of results.

Tong et al. [2008] try to measure proximity with side information. However, their model cannot handle multiple sets of could-be-conflicting side information, which happens a lot in a recommendation system.

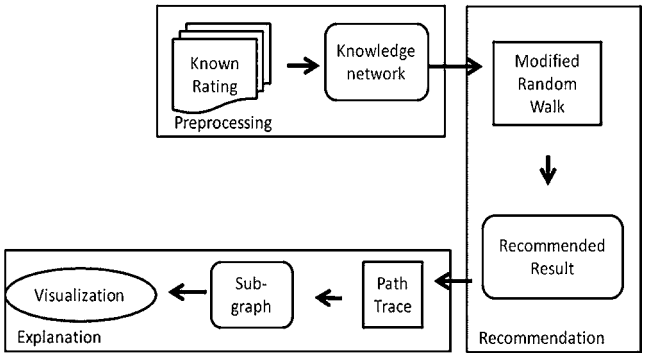

Fig. 5. The flowchart of the proposed method.

2.5. Retrieve Subgraph of Information

As will be discussed later, a subgraph is extracted as part of the explanation. Therefore we include some subgraph extraction algorithms as reference. Faloutsos et al. [2004] propose a way to quickly find a subgraph containing source vertex and target vertex with maximum flow from source to target. They later extend the algorithm to multiple sources and different logic conditions (AND, OR, SOFT-AND) [Tong and Faloutsas 2006].

3. DEALING WITH NEGATIVE OPINONS

In this article, we improve the random walk model in two ways.

(1) As mentioned before, existing RW-based methods do not pay special attention to negative feedbacks. They are usually treated as mild positive feedbacks. In contrast, we propose a method that handles bidirectional opinions with care. Our experiment results show that being able to deal with negative opinions can improve the recommendation performance.
(2) We propose a graph-based mechanism that generates explanations for the recommendations.

Figure 5 shows the flowchart of our framework. The center node of the system is a modified random walk that deals with bidirectional opinions. The input is a graph constructed based on the ratings that users give to the items. Finally, we trace the information flow path to produce visualized explanations for the recommended items.

3.1. Modified Random Walk

Without losing of generality, we assume that the ratings are discrete values distributed over the range $\{-2, +2\}$, where $+2$ means "like it very much," 0 stands for a neutral opinion, and -2 represents "strongly dislike." In the conventional random walk model, the strength of connections between users and items is proportional to the ratings, normalized to the positive side (See Figure 6(a)). However, as mentioned previously, the drawback of this model is that it tends to downplay the importance of negative ratings. Figure 6(b) shows the original graph in which both positive and negative ratings are weighted. Unfortunately, negative weights can cause convergence problem in random walk models. The transition probability $P(i/u) = r_{ij}/\sum r_{ij}$ will not remain bounded from [0,1] if some r_{ij} are negative. In the worse case, $\sum r_{ij}$ can be zero which results in infinite probability.

To deal with this problem, our modified RW model first assumes that each node stores two kinds of information (or probabilities). The first, denoted as $P(X^+)$, is the

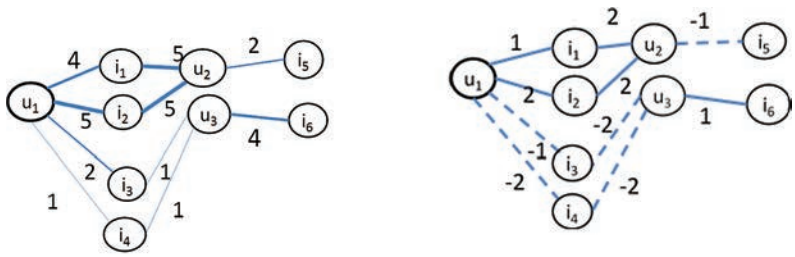

Fig. 6. (a) A graph constructed with rating values in the range [1,5]. The walks from u_1 tend to move toward i_5 than i_6. (b) A graph constructed with tuned rating values. The flow from u1 will be symmetrical in this case.

positive information and the second, denoted as $P(X^-)$, is the negative information of the node. The information from $P(X^+)$ and $P(X^-)$ propagate in different ways. Given a positive weight, $P(X^+)$ will propagate to its neighbors' $P(X^+)$ just like the normal RW would, while $P(X^-)$ will propagate to its neighbors' $P(X^-)$ similar to multiplication, negative*positive=negative. Given a negative weight between nodes, $P(X^-)$ should be propagated to its neighbors' $P(X^+)$ (i.e., negative*negative=positive), while $P(X^+)$ should be propagated to the neighbors' $P(X^-)$. The rationale is that if two people have similar preferences for items, they like or dislike similar things, their ratings should be propagated to each other. On the other hand, if two people have very different tastes, then their ratings should also be propagated to each other, but in a negative manner.

These conditions control the sign and direction of the propagation. Thus, we can focus on the amount of information being propagated by using the absolute strength as follows:

$$P(i_j|u_i) = \frac{|r_{ij'}|}{\sum_{j\in I_i}|r'_{ij}|} \quad \text{and} \quad P(u_i|i_j) = \frac{|r'_{ij}|}{\sum_{i\in U_j}|r'_{ij}|}. \tag{3.1}$$

To sum up, the model splits a node into two nodes, one containing positive information, *P(X+),* and the other containing negative information, *P(X−)*. Whether there is a link between the positive or negative node to the neighbors' positive/negative nodes will depend on the sign of the weight:

$N(X_i)$: neighbors of node X_i.

$$\text{When } r_{ik} \geq 0 \begin{cases} P(X_k^+|X_i^+) = \dfrac{|r_{ik}|}{\sum_{j\in N(X_i)}|r_{ij}|} \\ P(X_k^-|X_i^+) = 0 \\ P(X_k^-|X_i^-) = \dfrac{|r_{ik}|}{\sum_{j\in N(X_i)}|r_{ij}|} \\ P(X_k^+|X_i^-) = 0 \end{cases} \text{and} \quad \text{When } r_{ik} < 0 \begin{cases} P(X_k^+|X_i^+) = 0 \\ P(X_k^-|X_i^+) = \dfrac{|r_{ik}|}{\sum_{j\in N(X_i)}|r_{ij}|} \\ P(X_k^-|X_i^-) = 0 \\ P(X_k^+|X_i^-) = \dfrac{|r_{ik}|}{\sum_{j\in N(X_i)}|r_{ij}|}. \end{cases} \tag{3.2}$$

Because we are interested in the items that a user likes, we set the initial probability that P will propagate as P(source user$^+$)=1 and P(others)=0.

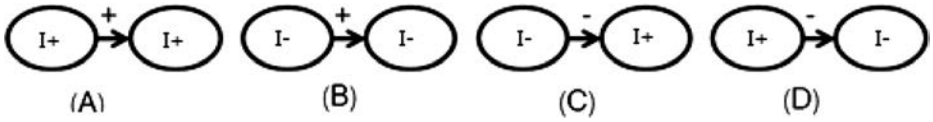

Fig. 7. The propagation of information in the proposed framework.

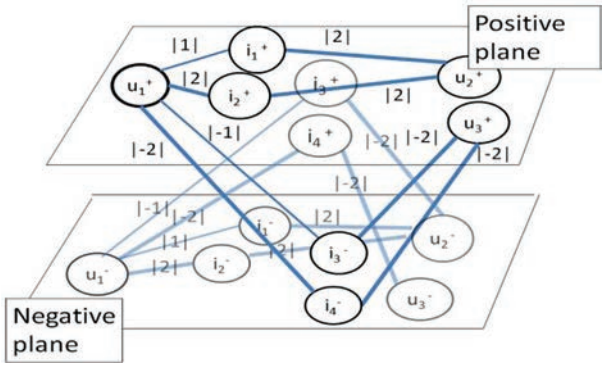

Fig. 8. The previous example in Figure 1 under our model. Every node is replicated in the negative plane to contain negative information. And the information can be transferred between positive and negative containers.

In addition, the updated probability in each iteration of RW is

$$P\left(X_k^+\right) = \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} \geq 0} P\left(X_i^+\right) P\left(X_k | X_i\right)\right] - \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} < 0} P\left(X_i^-\right) P\left(X_k | X_i\right)\right]$$

$$P\left(X_k^-\right) = \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} \geq 0} P\left(X_i^-\right) P\left(X_k | X_i\right)\right] - \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} < 0} P\left(X_i^+\right) P\left(X_k | X_i\right)\right]. \quad (3.3)$$

In (3.3), the positive value of a node is determined by the positive values of its neighbor nodes when the weight is poistive, and the negative values of its neighbors when the weight is negative (see parts (A) and (C) in Figure 7). The negative value of a node is determined by the negative values of its positive neighbors, and the positive values of its negative neighbors.

Finally, for each item node, the value $R(X_k) = P(X_k^+) - P(X_k^-)$ can be used as the final score to determine its ranking for recommendation. That is, an item that has more chance of being liked and less chance of being disliked by a user will be recommended.

In our modified random walk model, all values in the transition matrix are nonnegative; therefore, the model is guaranteed to converge. Figure 8 shows how the example in Figure 1 can be modeled under the modified random walk framework. Note that the main difference between our model and Clements et al.'s model is that we do not separate the positive and negative information. Figure 8 shows that the positive and negative planes of our model actually interact with each other to exchange information.

3.2. Equivalent Model

One limitation of the proposed model is that the number of nodes and edges in the graph doubles, which can significantly affect the efficiency. To address this problem, we propose an equivalent model that has the same complexity as the original RW model.

In fact, for recommendation systems, eventually we only need to derive the $R(X) = P(X^+) - P(X^-)$ values of the nodes. That is, the individual values of $P(X^+)$ and $P(X^-)$ are not important as long as we know the difference between them. As result, we find that, for each node, we only need to store its R values and propagate it based on a certain strategy. Thus, it is possible to apply the RW model to the original graph without replicating the nodes and edges. The updated rule for $R(X)$ is formulated in Equation (3.4), where $sign(W_{ik})$ denotes the sign of the given weight.

$$R(X_k)' = \sum_{x_i \in \mathrm{N(x_k)}} sign(w_{ik})\, R(X_i)\, P(X_k|X_i) \tag{3.4}$$

PROOF. $R(X_k) = P(X_k^+) - P(X_k^-) = R'(X_k)$

$$R(X_k) = P\left(X_k^+\right) - P\left(X_k^-\right)$$

$$= \left\{ \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} \geq 0} P\left(X_i^+\right) P(X_k|X_i) \right] - \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} < 0} P\left(X_i^-\right) P(X_k|X_i) \right] \right\}$$

$$- \left\{ \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} \geq 0} P\left(X_i^-\right) P(X_k|X_i) \right] - \left[\sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} < 0} P\left(X_i^+\right) P(X_k|X_i) \right] \right\}$$

$$= \sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} > 0} \left(P\left(X_i^+\right) - P\left(X_i^-\right)\right) P(X_k|X_i) + \sum_{x_j \in \mathrm{N(x_k)} \& r_{jk} < 0} \left(P\left(X_j^+\right) - P\left(X_j^-\right)\right) P\left(X_k|X_j\right)$$

$$= \sum_{x_i \in \mathrm{N(x_k)} \& r_{ik} > 0} R(X_i)\, P(X_k|X_i) + \sum_{x_j \in \mathrm{N(x_k)} \& r_{jk} < 0} R\left(X_j\right) P\left(X_k|X_j\right)$$

$$= \sum_{x_i \in \mathrm{N(x_k)}} sign(w_{ik})\, R(X_i)\, P(X_k|X_i) \Rightarrow R(X_k)'.$$

This proof shows that, instead of running RW on a larger graph *G(2n,2e)*, our equalivent model implemented on *R'* can be executed on the original graph *G(n,e)* without incurring any additional computational burden.

4. GENERATING EXPLANATIONS

We believe that providing the reasons behind recommendations will improve the probability of users accepting the recommended items. In this section, we describe our explanation mechanism on a graph-based RW model.

4.1. Information Tracing

The explanation for RW-based recommendations can be obtained by tracing the information flow of the graph. Based on this idea, we try to identify the paths that have the most influential flows (or the highest probability of being selected) from the source (the user) to the target (the recommended item) during the random walk process.

In our model, the influence of a node B on its neighbor A can be defined as

$$Influence_{B \to A} = P(A|B)P(B).$$

Starting from the item being recommended, it is possible to apply a greedy method to trace back and find the neighbor nodes X that contributed the most to the final score. Then, from X, we can keep tracing to another node Y with the most influence on

Algorithm: modified random walk

Input: G: Graph consists of ratings {user x item},u_{source}: the user who is recommended, α : decay parameter, λ : balance parameter

Goal. Given the relation graph, parameters, and the target, predict the ranking for all items

1. Mapping ratings to positive and negative opinions related to users (λ =0.65 in our experiment)

$$neutral_u = \lambda \bar{r}_u + (1-\lambda)\bar{r}$$

$$r_{ij}^{'} = r_{ij} - neutral_i$$

2. Computing transition probability between nodes pairs with respect to their ratings

When $r_{ik} \geq 0$

$$\begin{cases} P(X_k^+ \mid X_i^+) = \dfrac{|r_{ik}|}{\sum_{j \in N(X_i)} |r_{ij}|} \\ P(X_k^- \mid X_i^+) = 0 \\ P(X_k^- \mid X_i^-) = \dfrac{|r_{ik}|}{\sum_{j \in N(X_i)} |r_{ij}|} \\ P(X_k^+ \mid X_i^-) = 0 \end{cases}$$

and

When $r_{ik} < 0$

$$\begin{cases} P(X_k^+ \mid X_i^+) = 0 \\ P(X_k^- \mid X_i^+) = \dfrac{|r_{ik}|}{\sum_{j \in N(X_i)} |r_{ij}|} \\ P(X_k^- \mid X_i^-) = 0 \\ P(X_k^+ \mid X_i^-) = \dfrac{|r_{ik}|}{\sum_{j \in N(X_i)} |r_{ij}|} \end{cases}$$

3. Evaluate random with restart

$$R(X_k) = (1-\alpha)\left[\sum_{x_i \in N(x_k)} sign(w_{ik}) R(X_i) P(X_k \mid X_i)\right] + \alpha[R_0(X_k)]$$

Output: Rank: rank score of items R={R_1,R_2,R_3,…}

Fig. 9. The pseudocode of the modified random walk algorithm.

X in a greedy manner. The process continues until the original user node is reached. Figure 10 shows an example of such a path, which can be described as: An item, *B,* is recommended to the user who likes item A because another person, who also likes item *A,* likes *B*.

Generally, it is possible to search the top-k influencial paths using the greedy method. Eventually, we can obtain a set of paths that describe the dominant routes for passing the information from the source user to the recommended item. The paths can sometimes be merged. For example, Figure 11 shows three dominant paths from the user to the item. Based on his information, we can assume that because other users who like movie *A* also like movie *B*, the system will recommend movie *B*.

We also found that with additional knowledge in the network, it is possible to create more reasonable and inetresting explanations. The dominant paths in Figure 12 reveal that movie *B* is recommended to the user because it has similar actors and writers, and it is of the same genre as movie A, which is one of the user's favorite movies.

It is sometimes better to use a subgraph instead of a path or a set of paths to explain the recommendation. Figure 13 exemplifies the method we propose for this purpose.

Simiarily, we first generate the influence scores of each node on its neighbors as $Influence_{B \to A} = P(A|B)P(B)$. Again, a greedy method is used for backtracing until the

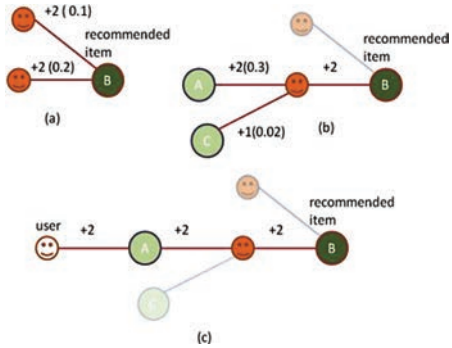

Fig. 10. An explanation path under rating information. The number in the bracket indicated the flow of influence.

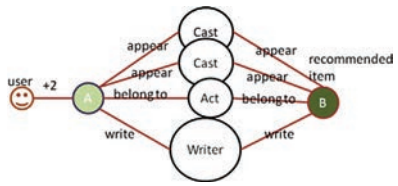

Fig. 11. The main cause is for recommending movie B to the user is that movie B is also liked by those users liked movie A.

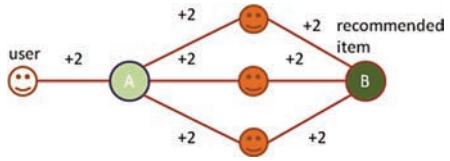

Fig. 12. The main cause for recommending movie B to the user is that movie B is highly related to movie A.

starting user node is found. However, this time the greedy method keeps several of the top nodes that, as a whole, account for more than k% of the influence score on the node. For example, in Figure 13(c), assuming k=51%, nodes E and F are kept in the subgraph because they sum up to more than 51% of the contribution to node B. In the end, given k, it is possible to obtain a subgraph that contributes most significantly to the recommendation of an item.

5. EXPERIMENT

5.1. Dataset

We use two popular datasets, the MovieLens and the Netflix Prize datasets, for evaluation. The MovieLens dataset consists of 943 users and 1682 movies, with a total of 100,000 ratings. We randomly sample data for five-fold cross validation and repeat the process three times. The reported results are the average of the outcomes of three trials. Two similar experiments were performed on the sized-down Netflix prize dataset. We selected the 2,000 DVDs rated by most users and randomly selected 1,000 users who rated those DVDs at least 40 times.

We also tested on a more sparse dataset where we select the same DVDs but a different set of 1,000 users who rate those DVDs only 20 to 100 times. We did not make it even sparser because by doing so the graph will not be connected (there will be several small islands).

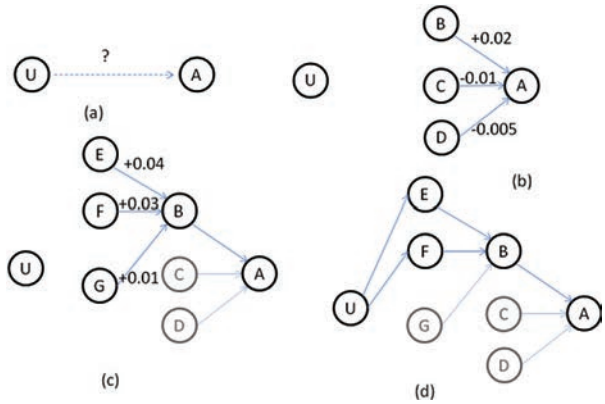

Fig. 13. (a) We want to trace a path from U to A. (b) Starting from the recommended item A, we first calculate the influence score for each neighbor, as shown above each edge. Then, we leave only the top nodes that account for 51% of the influence for backtracking. Here, the total absolute influence is $0.02 + 0.01 + 0.005 = 0.035$, while the influence from B to A accounts for $0.02/0.035 > 51\%$ of the contribution; therefore, only B is left in the subgraph for explanation. (c) Similar steps are performed from node B. This time we find that E does not contribute more than 51% , so only F is kept in the graph. (d) The same steps are performed until none of the neighbors are active.

5.2. Measurement

Following other random-walk recommendation methods, we use NDCG (Normalized Discounted Cumulative Gain) to measure the performance since RW-based systems produce a rank for each item. The NDCG is used because, comparing with other measurements such as RMSE, it is more suitable for ranking-based alrogithm such as RW. This is also a very common measure that has been widely used by other models such as CF-based and classifiction-based ones [Abbassi el al. 2009, Parra and Brusilovsky 2009, Zheng et al. 2010, Chapelle and Keerthi 2009].

NDCG measures how close the ranked results are to a perfect ranking.

$$NDCG_p = \frac{DCG_p}{IDCG_p} \quad \text{and} \quad 0 \le NDCG_p \le 1, \quad \text{where } DCG_p = \sum_{i=1}^{p} \frac{2^{rate_i} - 1}{\log_2(1+i)},$$

i is the ranking order, $rate_i$ is the corresponding rate of the item ranked at i position,
p stands for the top p ranking items,
and IDCG_p is the discounted cumulative gain with a perfectly ranked list.
The closer the NDCG is to 1, the closer the ranking is to the gold standard.

5.3. Competing Methods

We compare our models mainly with other state-of-the-art RW-based models, and we also include the CF-based model and the supervised model as reference. Note that the parameters for the competitive models are chosen according to the suggestions in the corresponding papers.

(1) *Collaborative filtering (CF(user), CF(item)).* This method predicts ratings based on user similarity [RESNICK, P. et al. 1994] or item similarity [Sarwar et al. 2001].
(2) *Original Random walk method (RW).* We compare our approach with the original random walk method, which simply shifts the ranking from $[-2, 2]$ to all positive scores [1,5] [Cheng et al. 2007] . The transition penalty$\alpha$is set to 0.2 and max iterations is set to 10.

Table I. Result of Experiments

NDCGs	CF(user)	CF(item)	SVM	KW	DW-PN	IR	SR	ER	MW
Movielen_{100k}	0.7798	0.7479	0.7901	0.7186	0.763	0.6599	0.7151	0.7423	0.7826
Netflix	0.7433	0.7074	0.7569	0.633	0.7273	0.5997	0.6602	0.7211	0.762
Netflix_{sparse}	0.5928	0.6235	0.6140	0.5843	0.6248	0.5608	0.5981	0.6202	0.6253

(3) *Positive and negative relevance random walks* [Clements et al. 2009]. We also compared the method proposed by Clements, et al., which splits the original graph into two graphs, one consisting of ratings in the range 3–5 to propagate positive information, and the other consisting of ratings in the range 1–2 to propagate negative information. Eventually, we merge the results from the two graphs to form a ranking denoted as RW-PN. The transition penalty α is set to 0.2 and max iterations is set to 10.
(4) *ItemRank (IR)* [Gori and Pucci 2007]. Use random walk model to capture the item correlation relations with user's preferred items. It is described in Section 2.3.1. The transition penalty$\alpha$is set to 0.15 and max iterations is set to 10.
(5) *Similarity Random Walk (SR)* [Yildirim and Krishnamoorthy 2008]. Use random walk model to capture the item similarity relations with user's preferred items. It is described in Section 2.3.2. The transition penalty α is set to 0.15 and max iterations is set to 20.
(6) *EigenRank(ER)* [Liu and Yang 2007]. Use random walk to aggregate the ranking relations of items. It is described in Section 2.3.3. The transition penalty α is set to 0.1, max iterations is set to 20 and the number of similar users is set to 100.
(7) *Supervised classifier.* Using the ratings of the targets as labels, and the ratings of other users as features, it is possible to train a classification model for recommendation. Here, we use SVM with an RBF kernel function to train a model for comparison. [Bomhardt 2004] The SVM parameter c is set to 10.0 and γ is set to 1/15072 for MovieLens dataset, and for Netflix, c is set to 10.0 and γ is set to 1/40000 .
(8) *Modified Random Walk (MW).* For our methods, we also set the transition penalty $\alpha = 0.1$ and max iterations $= 20$.

5.4. Results and Discussion

Table I shows the NDCG results of the given algorithms. We show CF-based methods first, and then supervised methods. The rest are all RW-based methods. The results reveal the following interesting phenomena.

Being able to handle negative information does improve the results significantly (see RW vs. MW, we obtain >10% improvement in NDCG for Netflix dataset). It basically confirms several things. First, treating negative ratings as truly negative (rather than less-positive) is critical. Second, our hypotheses to treat people who dislike similar movies as similar, and to use negative opinions of dissimilar persons as positive reinforcement are proper. Eventually it confirms that our overall model to handle negative examples is effective.

Our model also outperforms the state-of-the-art RW-based models by a significant margin, which confirms our advance of the RW-based methods in recommendation.

We also compare the best RW-model to the other types of methods. The results show that our model obtains similar quality of results comparing to CF-based method and supervised based method.

However, we would like to point out that the comparison among different types of models is only for reference, and we do not regard outperforming different types of algorithms as a research goal in this article. As it has been shown that in order to build a

Table II. Results of Weighted Voting of Three Different Algorithms on Movielens Dataset

weights	[SVM=0.7, MW=0.3]	[SVM=0.5, MW=0.5]	[SVM=0.4, CF=0.3, MW=0.3]	[SVM=0.1, CF=0.7, MW=0.2]
improvement	0.82%	0.44%	0.98%	1.05%

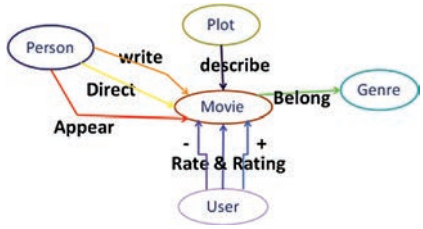

Fig. 14. The schema of relations for the movie dataset.

high-quality recommendation system, a blending method that combines different types of models is required. Therefore, we believe the goal for designing a recommendation system has to evolve from "outperforming other types of models" to "advancing state of the art in one type of model," as an inferior model with diversity might be even more helpful than a superior model with less diversity in improving the final outcome of blending [Toscher and Jahrer 2009; Chen P.-L. et al. 2011; McKenzie et al. 2011].

To verify such claim, we did a simple ensemble experiment to combine three diverse models (CF, SVM, and our MR model) to show the effectiveness of our model. We chose the MovieLens dataset. in which our model ranks 2nd in performance, better than CF but worse than the SVM model. Table II shows the results. First, we simply combine the predictions from the two better models (SVM and MR), and realize that even with equal weight, we can achieve 0.44% inprovement on performance. With slightly higher weight for SVM, the improvement can reach 0.82%. After adding the worse model (CF), the performance can further improve 0.23% to reach 1.05% total. Note that 1% difference is considered significant as the top teams in Netflix or KDD Cup 2011 are only seperated by one tenth of such scale. This experiment shows that even a relatively small portion of our model can contribute significantly to the overall results. Even better results can be expected with more sophisticated ensemble methods.

5.5. Demonstration of Our Explanation Generation Mechanism

To provide more readable explanations and to demonstrate that our recommendation framework can easily incorporate external information or knowledge, we provide an explanation on a heterogeneous graph whose relational schema is shown in Figure 14.

We collect extra information about movies from IMDB, including the directors, writers, genres, plots and the actors of each movie. Our model treats all of these relations as positive relations.

An online demomonstration system is available.[1] Given the user's id, the system will display a list of recommended movies to the user. Users then can check the explanations and visualization graph by clicking on the names of the movies.

Figures 15 to 17 are snapshots of the system. They are generated using the "subgraph-based" explanation technique discussed at the end of Section 4.1. The different colored edges and nodes represent different relations and node types. The two nodes with red outline and text color are the source user and the item to be recommended. Figure 14 displays the meanings of each kind of relation (in different colors). Figure 15 captures the major information flow from user 1 to the movie *Star Wars: VI*.

[1]http://mslab.csie.ntu.edu.tw:8080/recomex/ (userid: 1 ∼ 943 password: lab302).

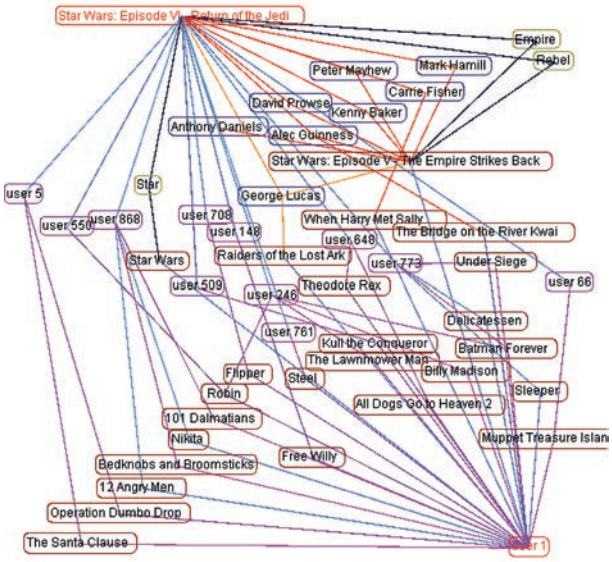

Fig. 15. The subgraph of the major information flow from user 1 to "Star Wars: VI" given k = 0.18.

Fig. 16. The observed first reason for recommending Star Wars: VI to user 1 by cut out sub graph related to high degree node, it can be easily understand that it's because Star Wars:V.

Focsuing on the high-degree nodes and their neighbors, we can see that the node with the highest degree is *Star Wars: V* (isolated in Figure 16). Thus, the user can conclude that "The system believes that *Star Wars: Episode V* is worth recommending because one of the user's favorite movies, *Start Wars VI*, is highly correlated with *Star Wars: Episode V* (i.e., they have the same director, producer and actor)". Figure 17 illustrates another example where *Apollo 13* is recommended to user 505. The graph shows that the system recommends this movie for three reasons. 1) People who like the movies that are rated favorably by the user also like *Apollo 13* (note that CF can also capture this); 2) *Apollo 13*'s director, Ron Howard, also directed the movie *Ransom*, which is liked by the user; and 3) Several actors in *Apollo 13,* such as Tom Hanks, Bill Paxton, and Gray Sinise, also performed in some of the user's other favorite movies.

5.6. Human Study of Explanation Effectiveness

We conducted a human study to evaluate the effectiveness of our explanation system. We first identify 100 popular movies (top 50 movies in moviemeter and top 50 movies

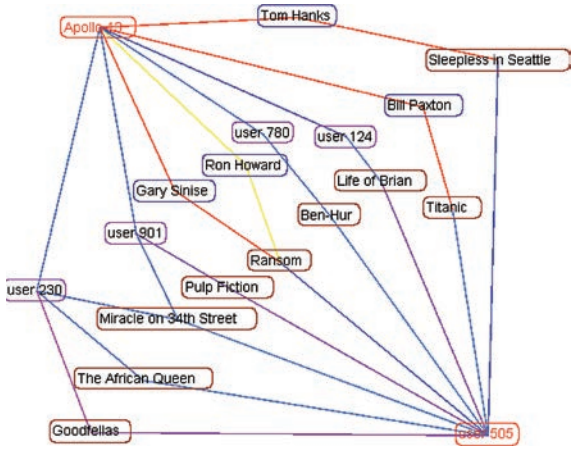

Fig. 17. The subgraph of the major information flow from user 505 to "Apollo 13".

in US box office) each year from 2005 to 2009 and 10 subjects. Out of these 500 movies, for each subject, we randomly select 50 movies and ask them to rate (the label "unseen" shall be given for unseen movies). We then construct a graph of all movies from 2005 to 2009. Then for each subjec t, we connect him/her to the rated movies to perform our RW model. For evaluation, we first display the top 10 recommended movies by the system to the subjects and ask them to check whether they want to watch those movies. The results show an average of 5.7, which means 57% of the recommended movies are accepted by the subjects. Then we show again the same movies recommended by our system with explanations to the same set of users, and ask them to mark again whether they want to watch the movies after viewing the explanations. The accepted rate then raised to 66%, which means the user changed their mind for one out of the four recommendations that were not accepted in the first place.

6. CONCLUSION

In this article, we have proposed a modified random walk model that allows the propagation of both positive and negative information simultaneously. We show that our model is guaranteed to converge, and there is no additional computation overhead. Furthermore, we demonstrate the advantages of the RW-based model by presenting a simple, efficient, yet intuitive explanation framework. We believe the major contribution of this article does not necessarily lie in the improvement of the recommendation results; rather, it highlights two issues that have not received much attention thus far: handling bidirectional opinions and providing explanations for the ranking of recommendations. Furthermore, we have demonstrated that by combining our system with other recommendation engines, it is possible to achieve even better performance.

Future work will include the investigation of how to deal with cold start and sparse data issues, as well as how explanations for other types of recommendation systems (such as CF and MF) can be produced.

REFERENCES

Abbassi, Z., Amer-Yahia, S., Lakshmanan, L. V. S., Vassilvitskii, S., and Yu, C. 2009. Getting recommender systems to think outside the box, in *Proceedings of the Recommender Systems Conference*. 285–288.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. 1985. A learning algorithm for Boltzmann Machines. *Cognitive Sci. 9*, 1, 147–169.

ADOMAVICIUS, G. AND TUZHILIN, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng. 17*, 6, 734–749.

ATHREYA, K. B., DOSS, H., AND SETHURAMAN, J. 1996. On the convergence of the Markov chain simulation method. *Ann. Statist. 24*, 1, 69–100.

BOMHARDT, C. 2004. Newsrec, a svm-driven personal recommendation system for news websites. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*.

BRIN, S. AND PAGE, L. 1998. Anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*.

CHAPELLE, O. AND KEERTHI, S. S. 2009. Efficient algorithms for ranking with svms. *Inf. Retrieval J*.

CHEN, P.-L., TSAI, C.-T., ET AL. 2011. A linear ensemble of individual and blended models for music rating prediction. In *Proceedings of the ACM SIGKDD KDD-Cup Workshop*.

CHENG, H., TAN, P.-N., STICKLEN, J., AND PUNCH, W. F. 2007. Recommendation via query centered random walk on k-partite graph. In *Proceedings of the IEEE International Conference on Data Mining*. 457–462.

CLEMENTS, M., DE VRIES, A., AND REINDERS, M. J. T. 2009a. Exploiting positive and negative graded relevance assessments for content recommendation. In *Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph*. Springer, 155–166.

CRASWELL N. AND SZUMMER, M. 2007. Random walks on the click graph, In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

FALOUTSOS, C., MCCURLEY, K. S., AND TOMKINS, A. 2004. Fast discovery of connection subgraphs. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. 118–127.

FOUSS, F.,PIROTTE, A.,RENDERS, J.-M., AND SAERENS, M. 2007. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.,* 355–369.

GALLAGHER, B., TONG, H. ELIASSI-RAD, T., AND FALOUSOS, C. 2008. Using ghost edges for classification in sparsely labeled networks, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, New York.

GORI, M. AND PUCCI, A. 2007. Itemrank: a random-walk based scoring algorithm for recommender engines. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 2766–2771.

GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. Propagation of trust and distrust. In *Proceedings of the 13th International Conference on World Wide Web*.

HAYNES, S. R. 2001. Explanation in information systems: A design rationale approach. PhD thesis, Department of Information Systems and Department of Social Psychology, London School of Economics and Political Science.

HERLOCKER, J., KONSTAN, J., AND RIEDL, J. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*. 241–250.

HOFMANN, T. Latent semantic models for collaborative filtering. 2004. *ACM Trans. Inf. Syst., 22*, 1, 89–115.

JAMALI, M. AND ESTER, M. 2009. TrustWalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, Eds., ACM, 397–406.

KOREN, Y., BELL, R. M., AND VOLINSKY, C. 2009. Matrix factorization techniques for recommender systems. *Comput. 42*, 8.

LIBSVM. http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html.

LIU, N. N. AND YANG, Q. 2008. Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 83–90.

LIU, N. N., ZHAO, M., AND YANG, Q. 2009. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the International Conference on Information and Knowledge Management*. 759–766.

MCKENZIE, T.G., FERNG, C. S., CHEN, Y. N., LI, C. S., TSAI, C. H., WU, K. W., CHANG, Y. H., AND LI, C. Y. 2011. Novel models and ensemble techniques to discriminate favorite items from unrated ones for personalized music recommendation. in *Proceedings of the ACM SIGKDD KDD-CUP Workshop*.

MIRZA, B. J., KELLER, B. J., AND RAMAKRISHNAN, N. 2003. Studying recommendation algorithms by graph analysis. *J. Intell. Inf. Syst. 20*, 2, 131–160.

MOVIELENS, http://www.grouplens.org/node/73.

NETFLIX PRIZE. http://www.netflixprize.com/.

PARRA, D. AND P. BRUSILOVSKY, P. 2009. Collaborative filtering for social tagging systems: an experiment with CiteULike. In *Proceedings of the 3rd ACM Conference on Recommender Systems*.

PIOTTE, M. AND M. CHABBERT, M. 2009. The pragmatic theory solution to the Netix grand prize. Tech. rep.

RESNICK, P., IACOVOU,N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the Conference on Computer Supported Collaborative Work*. R. Furuta and C. Neuwirth, Eds., ACM, New York. 175–186.

SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th InternationalWorldWideWeb Conference*.

TAKCS, G., PILSZY, I., NEMETH, B., AND TIKK, D. 2007. Matrix factorization and neighbor based algorithms for the Netflix prize problem, In *Proceedings of the ACM Conference on Recommender Systems*.

TONG, H. AND FALOUTSOS, C. 2006. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 404–413.

TONG, H., HUIMING H., AND JAMJOOM, H. 2008. Measuring proximity on graphs with side information. In *Proceedings of the 8th IEEE International Conference on Data Mining*. 598–607.

TOSCHER, A. AND JAHRER, M. 2009. The BigChaos solution to the Netix grand prize. Tech. rep.

YILDIRIM, H. AND KRISHNAMOORTHY, M. S. 2008. A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, New York, NY, 131–138.

ZHANG, J., TANG, J., LIANG, B., YANG, Z., WANG, S., ZUO, J., AND LI, J. 2008. Recommendation over a heterogeneous social network. In *Proceedings of the 9th International Conference on Web-Age Information Management.*

ZHANG, Z.-K., ZHOU, T., AND ZHANG Y.-C. 2010. Personalized recommendation via integrated diffusion on user-item-tag tripartite graphs. *Physica A 389,* 179–186

ZHENG, B., ZHENG, X., AND YANG, Q. 2010. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proceedings of the National Conference on Artificial Intelligence*.

Received March 2011; revised August 2011; accepted October 2011