Knowledge-Based Systems 214 (2021) 106720

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Hybrid ecommerce recommendation model incorporating product taxonomy and folksonomy

Mingsong Mao, Sihua Chen *, Fuguo Zhang, Jialin Han, Quan Xiao

School of Information Management, Jiangxi University of Finance and Economics, Nanchang, China

ARTICLE INFO

Article history:
Received 9 August 2020
Received in revised form 25 November 2020
Accepted 19 December 2020
Available online 7 January 2021

Keywords:
E-commerce
Recommender system
Random walk
Taxonomy
Folksonomy

ABSTRACT

In modern ecommerce platforms, product content information may have two origins: one is tree-structured taxonomy attributes, and the other is free-form folksonomy tags. This paper proposes a hybrid model to incorporate taxonomy and folksonomy information to enhance ecommerce recommendations. It first develops a tree matching algorithm to establish the overall similarity between items, where tag information is integrated for semantic analysis for taxonomy attributes. Next, it proposes a unique random walk model on a heterogeneous graph constructed by user nodes and item nodes and different types of relations — user–item preference and item–item similarity relations. The random walk model is designed to be effective to identify the nearest item nodes for a particular user node, which are seen as the best-fit items for recommendations. Empirical experiments demonstrate that the proposed model improves performance in terms of both recommendation coverage and accuracy, especially for sparse data.

1. Introduction

Recommender systems have been widely deployed as shopping assistants in large-scale ecommerce sites like Amazon, Taobao, eBay, etc., where individual users are hard to navigate all candidate items [1–3]. It has been reported that recommender systems are able to enhance ecommerce by encouraging potential buyers, increasing cross-selling and building customer loyalty [ibid]. Recommendation techniques are generally classified to three strategies: (1) Content-based (CB) approaches that search items similar to the ones previously chosen by users [4]; (2) Collaborative filtering (CF) that poll items preferred by other users with similar preferences [5], and (3) hybrid models that make combination of both CB and CF ideas [6,7]. Hybrid models are thought to have advantages in incorporating different sources to alleviate the sparseness problem of CF or CB models that rely only on single-source data. In the field of ecommerce, product content information is with different forms and is usually integral for recommendation making. The most common content information of ecommerce products are the standard taxonomy attributes. For example, an ecommerce seller will indicate the attributes of a standard taxonomy for a laptop, like CPU, memory, hard-drive, etc., in a tree-structure. Despite standard taxonomy attributes, user-generated-content (UGC) like tags also constitutes item content information with the development of Web 2.0 applications. A tag like "gaming laptop", for example, may indicate that the laptop is with high graphical capacity. Compared to taxonomy, tags are called as a type of "folksonomy" information. Although there have been some models utilizing taxonomy [8,9] and folksonomy [10,11] separately for recommendations, we notice that there are few integrating both [12]. Thus, this study develops a hybrid ecommerce recommendation model by incorporating both product taxonomy and folksonomy information.

Three common information resources in ecommerce are utilized in this study as input: explicit user preference (ratings), item taxonomy attributes and item folksonomy tags. We firstly perform random walks on a heterogeneous graph by considering users and items as two groups of nodes, and two types of relations as edges — user–item preference relations and item–item similarity correlations. The user–item preference relations are estimated from the ratings given by users to items, and the item–item correlations are built by comparing item content information, for which we develop a tree matching algorithm integrating both taxonomy and folksonomy information.

The contributions of this study are that it provides a tree matching method as a fusion scheme to handle different types of content information — tree-structured taxonomy attributes and free-form folksonomy tags; and it designs a unique random walk model on a heterogeneous graph with different types of nodes and edges. Like other hybrid recommendation models, this study aims to utilize heterogeneous information to improve

* Corresponding author.
E-mail addresses: maomingsong@jxufe.edu.cn (M. Mao), doriancsh@foxmail.com (S. Chen), redbird_mail@163.com (F. Zhang), jialinhan@jxufe.edu.cn (J. Han), xiaoquan@foxmail.com (Q. Xiao).

https://doi.org/10.1016/j.knosys.2020.106720
0950-7051/

recommendation coverage while maintaining high recommendation accuracy. By incorporating taxonomy and folksonomy information besides ratings, this study establish various correlations between users and items, thus improve recommendation coverage. Empirical experiments also demonstrate that more precise ranking results can be derived by the proposed random walk model. Hence, the proposed tree match and random walk model methods effective in improving recommendation performance in terms of both coverage and accuracy.

The remainder of this paper is as follows. Section 2 includes related works of our recommendation models. In Section 3, we conduct a comprehensive content analysis for ecommerce products with both taxonomy and folksonomy information, where a taxonomy tree matching algorithm is proposed. In Section 4, we propose a random walk model on a user–item graph with different types of nodes and edges. Recommendations can be made by searching the nearest item nodes for a particular user node. Section 5 conducts experimental analyses for the performance of our model. Conclusion of this study and future research directions are given in the last section.

2. Related works

Ecommerce recommender systems can be seen as search engines to find items that may be interested by users. As aforementioned, CB, CF and hybrid models are the three main recommendation strategies. The key idea of CB is to find items with similar content attributes of those previously chosen by users [4,13]. CF takes a different approach by extracting user profile from historical preferences [5,14,15]. CF can be further divided into memory-based and model-based. Memory-based CF predicts user ratings to unknown items by aggregating the preferences of neighbor users who share similar preferences [16]. Model-based CF approaches, on the other hand, are based on prediction models in which some parameters have been trained with previous data as input. Examples of model-based CF include matrix factorization models [17], probabilistic topic models [18], fuzzy models [19] , neural networks [20], or other optimization models like game theory [21], etc.

This study can be seen as a hybrid model integrating both CF and CB ideas. In general, pure CB and CF models suffer data sparseness problem as they resort only on single source. Hybrid models are expected to improve recommendation coverage while maintaining high accuracy. Existing hybridization strategies can be categorized into three ways [22]. One is to combine the results of different models after implementing them separately. In [23], for example, a trust-enhanced collaborative filtering model and a semantic content matching model are conducted separately to predict missing ratings, and the average scores are treated as final predictions. This type of post-hoc combinations are generally able to overcome the rating sparseness problem but are hard to increase prediction accuracy. The second type of hybridization is to incorporate the CF characteristics to enhance CB models [24,25]. The third way is to incorporate CB characteristics to enhance CF, for example, to derive user preferences based on item content information [26,27], which is similar to our study. We conduct content analysis to establish similarity correlations between items and import the result as input for an enhanced CF model based on random walks.

In ecommerce applications, system managers or sellers describe their products based on standard tree-structured taxonomies and some tree-matching algorithms have been successfully developed for recommendations. Existing models include tree similarity measure, tree isomorphism, and sub-tree comparison, etc., aiming at semantic analysis for various taxonomy attributes. In the food recommender system for diabetes patients [28], items (food menus in this case) are represented as hierarchical food ontology and users (patients) are represented as weighted nutrition demanding trees. Two patients are then comparable using a weighted tree matching method so that personal food menus can be generated for new patients, according to the diet plans of existing similar patients. Wu et al. [8] propose a fuzzy tree matching algorithm, in which fuzzy taxonomy trees and fuzzy profile trees are trained for items and users in advance and the best-matched items are selected for recommendations. In [9], Zheng et al. extract user hierarchical interest based on the analysis of item content ontology, and with such comparable tree-structured preferences, neighbor users can be detected for CF-like recommendations.

This study is also related to early recommendation models based on user–item heterogeneous graph random walk models. In these models, the active user is treated as the starting point to perform random walks, and recommendations are generated by estimating the ranks of stationary visiting probabilities of items. More extended graphs between users and items are proposed to incorporate different input resources. For example, authors in [29] considers the rating data as relations between user nodes and item nodes and propose a bipartite graph ranking method to address the recommendation problems. Furthermore, authors of [30] consider the possible multiple relations between two users and proposes a multigraph ranking model for multirelational social recommendations. Correlations between items have also been imported and random walks can be performed on item–item graphs, like the ItemRank model [31]. Subsequent studies also consider item content information as additive special nodes to build extended graphs with multiple groups of vertices and edges [32,33]. In general, a multi-partite graph is represented as a graph $G = (V_{\text{user}}, V_{\text{item}}, V_{\text{Attribute1}}, V_{\text{Attribute2}}, \ldots, E)$ containing user nodes, item nodes, and different aspects of taxonomy attributes such as movie genres, actors, directors, etc. Random walks will be performed by jumping between different types of vertices following different types of edges.

3. Tag-integrated taxonomy tree matching

As aforementioned, ecommerce products are with two different forms of content information: tree-structured taxonomy attributes and free-form folksonomy tags. In this section, a tree matching algorithm is developed to infer the overall content similarity between items.

3.1. Representation of taxonomy and folksonomy

3.1.1. Taxonomy tree

Taxonomy information of products in a specific domain usually consists of descriptions from multiple aspects with multi-level tree structure. For example, the descriptions of a book in Amazon [1] contain various attributes for the “category”, “author”, “language”, etc., and a restaurant in Yelp[2] is associated with attributes of the aspects of “category”, “trading hours”, “cuisine”, etc. We consider these aspects as the top-branches of the taxonomy tree and denote a virtual node for each branch. These aspect nodes are the special *aspect layer* of taxonomy trees in a specific domain, like books, restaurants, etc. Every aspect node “dominates” a subtree with single or multiple levels of sub-attributes, which provide deeper level details of item content in this aspect. We call other attributes except the virtual aspect nodes as *attribute layer*, which is further identified as *level 1 attribute layer*, *level 2 attribute layer*, etc., according to the depth.

[1] https://www.amazon.com/dp/1449361323?ref=emc_b_5_i.

[2] http://www.yelp.com.au/c/sydney/restaurants.

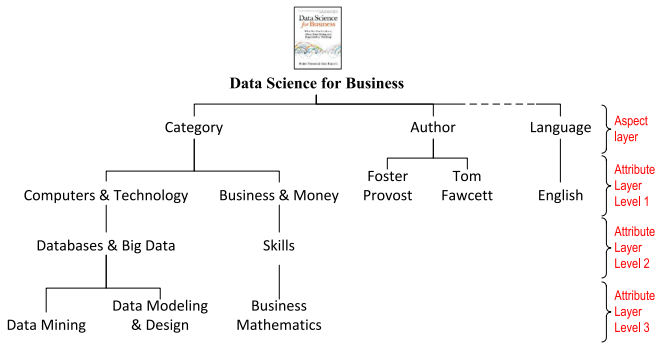


Fig. 1. Example taxonomy tree of a book in Amazon.

Fig. 1 is the multi-level taxonomy tree of a book in Amazon, showing that subtrees under different aspect nodes do not have to be of equal depth. For example, the branch of *category* has three levels of attributes, while the branch of *author* has only one level attributes.

The definition of taxonomy trees is given below.

Definition 1 (*Domain Taxonomy Tree*)**.** A taxonomy tree of a specific domain of items is defined as a directed graph $\Gamma = \{A, \rightarrow\}$ with no cycles, where $A = \{a_1, a_2, \ldots\}$ is a finite node set of taxonomy attributes, and $\rightarrow$ is a "parent–child" relation. For two nodes $a_i, a_j \in A$, if $a_i \rightarrow a_j$, then a_j is a child attribute of a_i, and we say $a_j \in child(a_i)$ and equivalently $a_i = parent(a_j)$. Note that a node can have multiple (or zero) child nodes, but must have one unique parent node. The root node $root(\Gamma)$ is a virtual node representing the topmost parent node, and for any node $a_i \in A$, there is one and only one path from $root(\Gamma)$ to a_i.

The above definition is the general taxonomy tree for all items in a domain. For an individual item $m \in M$, where M denotes the whole item set, it has its own taxonomy tree $\Gamma_m = \{A_m, \rightarrow\}$, where $A_m \subseteq A$ is a subset of A, and Γ_m is a subtree of Γ.

Let us suppose there are in total K aspect nodes collected from the taxonomy trees of all items, denoted as $C = \{c_1, c_2, \ldots, c_K\}$. Hence, the domain taxonomy tree Γ and every single tree Γ_m for $m \in M$ will have K branches indicating attributes from different aspects.

3.1.2. Folksonomy tags

In modern ecommerce, customers are also able to create descriptions in terms of tags as an extra facet of item content information. Despite creating new tags every time, users can also select existing tags to describe items. Thus, an item can be assigned with a same tag repeatedly, and the count of their co-occurrence indicates the strength of how much this item is relevant to this tag. In detail, we use the tf–idf metric (*term frequency–inverse document frequency*) to represent the relevance between items and tags. Let us denote the whole tag set as $T = \{t_1, t_2, \ldots\}$, and the tf–idf measurement for a single item $m \in M$ and a particular tag $t \in T$ is calculated by:

$$\text{tf-idf}(m, t) = \text{tf}(m, t) \times \log \frac{|M|}{\#\text{items containing } t} \tag{1}$$

where $\text{tf}(m, t)$ is the times of that tag t being assigned to item m, and $T_m \subset T$ is the set of tags assigned to item m. With this metric, the tags appearing in majority of items are penalized as they are not able to discriminate items well.

Different with tree-structured taxonomy information, the folksonomy information of an item m is thus represented with a vector-form with $|T|$ elements, in which the ith element is the value of $\text{tf-idf}(m, t_i)$.

In general, taxonomy information is relatively complete information, but folksonomy information could be absent or incomplete if no or few tags are assigned to a particular item. We conduct an empirical analysis of the tag distributions with a public dataset of Movielens.[3] In the dataset, about thirty percent of items have no tags. For the rest items, we present the item-tag frequency distributions in Fig. 2(a). We find that a large number of items contain only a few tags, i.e., less than five. We also present the item frequency for tags in Fig. 2(b), and it shows that there are also many tags only appearing in a small number of items.

3.2. Semantic analysis for taxonomy attributes

To incorporate the advantages of both taxonomy and folksonomy, we establish the correlations between taxonomy attributes and tags. In detail, a pair of an attribute $a \in A$ and a tag $t \in T$ can be linked via the common items that contain both of them:

$$f(a, t) = \sum_{m: a \in \Gamma_m} \text{tf-idf}(m, t) \tag{2}$$

Furthermore, we consider that tags are also issued with respect to the aspects of taxonomy attributes. Referring to the virtual aspect nodes defined in taxonomy tree, we call the correlation relationship between a tag and an aspect node $c \in C$ as "domination" relationship, that is, if a tag t is dominated by an aspect node $c \in C$, it means the tag is used as description with regarding to the aspect c. For example, the most issued tags of the movie "Avatar" include "sci-fi", "James Cameron" and "too long". Intuitively, these tags are descriptions from the aspects of *genre*, *director* and *running time*, respectively, which are virtual aspect nodes for movie taxonomy trees. In our study, the domination relationships are detected by the following definition.

Definition 2 (*Tag Domination*)**.** For each tag t, it is dominated by one and only one aspect node $c \in C$. The domination aspect is detected by the following:

$$t \prec \operatorname*{argmax}_{c \in C} \max_{a \in child(c)} f(a, t). \tag{3}$$

The above definition indicates that a tag is dominated by the aspect with the most relevant attribute. It can be explained by a simple example: if a tag "adventure" is found to be highly correlated to an attribute "sci-fi" under the aspect node "genre" in the movie taxonomy tree, then we consider that this tag is also a description from the aspect "genre", though it is not included as standard taxonomy attribute for movie genre.

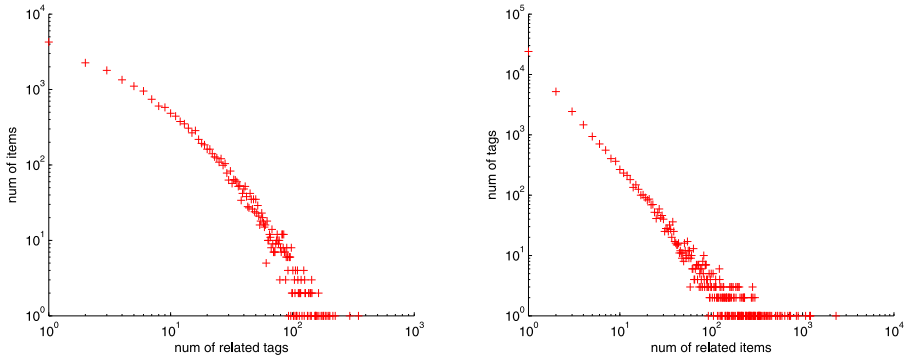
Figuring out the dominance aspect of tags is also helpful to understand the characteristics of items, and to conduct more precise comparison between them, which is discussed later in the next section.

Semantic similarities between different attributes are essential for matching item taxonomy trees and human expects are usually needed in previous studies. By integrating folksonomy information, however, we can infer semantic similarities between taxonomy attributes automatically. The following proposes the tag-derived semantic similarity between attributes.

Definition 3 (*Tag-Derived Semantic Similarity of Taxonomy Attributes*)**.** For two taxonomy attributes except the aspect nodes, $a_1, a_2 \in A$ satisfying $a_1, a_2 \notin C$, the tag-derived semantic similarity between them is defined as:

$$ss(a_1, a_2) = \frac{\sum_{t \in T} f(a_1, t) f(a_2, t)}{\sqrt{\sum_{t \in T} f(a_1, t)} \sqrt{\sum_{t \in T} f(a_2, t)}} \in [0, 1], \tag{4}$$

[3] http://grouplens.org/datasets/movielens/.



(a) Tag frequency distribution (b) Item frequency distribution

Fig. 2. The item-tag frequency distributions (Movielens-latest dataset).

In the above definition, *Cosine* correlation is used to produce a decimal value as the semantic similarity. It is worth mentioning that (4) is able to uncover implicit correlations between taxonomy attributes even at different levels to prevent vague human evaluation.

3.3. Tree matching algorithm

As aforementioned, user generated folksonomy is not always available while standard taxonomy is relatively needed for ecommerce products, therefore we compare taxonomy trees to infer the overall content similarity between two items.

With our definitions, item taxonomy tree contains K top-level subtrees corresponding to the aspect nodes $c_1, c_2, \ldots, c_K$. We first compare each pair of subtrees under the same aspect node separately, and aggregate the results as the overall content similarity. For each aspect node, we use a top-down matching manner to compare the two subtrees under this aspect.

Given two items m_1 and m_2 with taxonomy trees Γ_1 and Γ_2, respectively, the overall taxonomy similarity between them is aggregated by the matching results of the subtrees under all K aspect nodes:

$$sim(m_1, m_2) = \sum_{i=1}^{K} w_i Match(\Gamma_1[c_i], \Gamma_2[c_i]), \tag{5}$$

where $\Gamma[c]$ denotes the subtree of a taxonomy tree Γ under an aspect node $c \in C$, and $w_i > 0$ is the weighting of each aspect for aggregation satisfying $\sum w_i = 1$.

3.3.1. Weightings

Here we import the tag domination information to automatically generate the weightings in (5). For all items in a specific domain, by gathering the tag domination of all tags, we can discover which aspect for this domain is most discussed or concerned by users. For example, if most tags for books are, as defined in this study, dominated by the aspect "author", then we say the "author" of books is most concerned by users and should be of higher priority when comparing book taxonomy trees. The following equation is used to build the weightings of each aspect node $c_i \in C$:

$$w_i = \frac{\sum_{t \prec c_i} \sum_{m \in M} \mathrm{tf}(m, t)}{\sum_{t \in T} \sum_{m \in M} \mathrm{tf}(m, t)}, i = 1, \ldots, K. \tag{6}$$

With (6), we can establish a "global" weightings for the aspect nodes in the taxonomy tree of a specific domain of items, such as books, movies, etc.

3.3.2. Subtree matching

A top-down matching process for each pair of subtrees under a same aspect node is proposed below. At each level, two tasks are undertaken to match the attribute nodes:

1. Connect the common nodes. This step pairs those nodes exactly shared in both sides. Next-level comparison will be undertaken for their children nodes in both sides of the subtrees.
2. Match the non-common nodes. For the non-common nodes that appear only in one side, this step detects which are best matched. Next-level comparison will not be undertaken for their children nodes.

When no more nodes can be matched, the difference at the current level l is calculated by:

$$\delta_l = \frac{\sum(1 - ss(\text{paired non-common nodes}))}{|\text{paired non-common nodes}| + |\text{paired common nodes}|}. \tag{7}$$

After all levels comparison being completed, the matching result of two subtrees under an aspect node c is obtained by the following:

$$Match(\Gamma_1[c], \Gamma_2[c]) = \prod_l (1 - \lambda_l \delta_l) \tag{8}$$

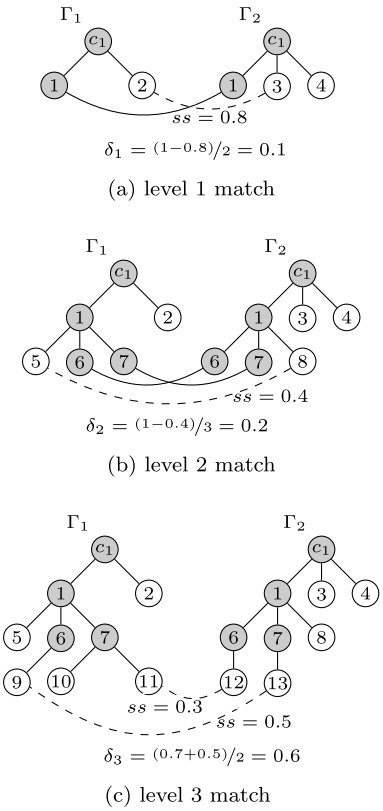
Here a positive parameter $\lambda \in [0, 1]$ is set to decrease along with increasing depth of matching. Simply, for example, we let $\lambda_l = 1/l$ in this study. The Algorithm 1 shows the main steps of our top-down subtree matching algorithm.

In addition, Fig. 3 shows a small example, where the subtrees under an aspect c_1 of the taxonomy trees Γ_1 and Γ_2 of two items are given. With the proposed matching algorithm, we first match the topper level nodes in the attribute layer. In level 1 as shown in Fig. 3(a), we find a common node 1 appearing in both sides. For the rest three nodes 2, 3 and 4, assuming the semantic similarity between node 2 and node 3 is higher than the similarity between node 2 and node 4, i.e., $ss(2, 3) > ss(2, 4)$, then we pair node 2 and node 3 in this level matching. At level 1, we obtain $\delta_1 = {}^{(1-0.8)}/_2 = 0.1$, according to (7). Next, we conduct level 2 comparison for the child nodes of the common node 1 and omit the child node information of other nodes (2, 3 and 4). In Fig. 3(b), we determine two common nodes 6 and 7, and match node 5 with node 8, and this level difference is $\delta_2 = {}^{(1-0.4)}/_3 = 0.2$. Similarly, in level 3 comparison of Fig. 3(c), two pairs of nodes (9, 12) and (10, 13) are matched and the difference at this level is $\delta_3 = {}^{(0.7+0.5)}/_2 = 0.6$. After all levels matching completed, the similarity of two subtrees under the aspect c_1 is:

$$Match(\Gamma_1[c_1], \Gamma_2[c_1]) = 0.9 \times 0.9 \times 0.8 = 0.648.$$

Data: Two subtrees $\Gamma_1[c]$ and $\Gamma_2[c]$ under an aspect node c
Result: subtree similarity $sim(\Gamma_1[c], \Gamma_2[c])$
Initialization:
  initial matching level $l = 1$;
  candidate nodes D_1: child nodes of c in $\Gamma_{1[c]}$;
  candidate nodes D_2: child nodes of c in $\Gamma_2[c]$.
while $D_1 \neq \emptyset$ *and* $D_2 \neq \emptyset$ **do**
  initialize $sum = 0, n = 0$
  get common candidates $CO \leftarrow D_1 \cap D_2$
  get candidates only in Γ_1: $X_1 \leftarrow D_1 - D_2$
  get candidates only in Γ_2: $X_2 \leftarrow D_2 - D_1$
  while $X_1 \neq \emptyset$ *and* $X_2 \neq \emptyset$ **do**
    pair $(x_1, x_2) \leftarrow \arg \max_{a \in X_1, b \in X_2} ss(a, b)$
    update $sum \leftarrow sum + (1 - ss(x_1, x_2))$
    update $n \leftarrow n + 1$
    delete x_1 from X_1
    delete x_2 from X_2
  obtain this level difference $\delta_l \leftarrow \frac{sum}{n+|CO|}$
  update matching level $l \leftarrow l + 1$
  clear all nodes in D_1 and D_2
  for *each common node d in CO* **do**
    add child nodes of d in $\Gamma_1[c]$ to D_1
    add child nodes of d in $\Gamma_2[c]$ to D_2
return comparison result $sim(\Gamma_1[c], \Gamma_2[c]) = \prod_l (1 - \lambda_l \delta_l)$

Algorithm 1: Subtree match algorithm



(a) level 1 match

(b) level 2 match

(c) level 3 match

Fig. 3. Top-down subtree matching example.

4. Hybrid recommendation framework

Let us consider a common scenario of ecommerce applications with such given information: users can evaluate items with scaled ratings; and items are described with taxonomy attributes (provided by sellers) and folksonomy tags (created by users). In this section, we propose a hybrid recommendation model based on random walks for a heterogeneous graph of users and items with different types of relations.

4.1. Heterogeneous graph construction

Connections for users and items can be derived from different resources. First, we consider the user ratings as user–item correlation relations. Second, with the above proposed tree matching algorithm, we can build similarity correlations between items from the tree-structured taxonomy attributes and free-form folksonomy tags. Consequently, we consider a unique graph with two groups of nodes, users and items, and two types of relations, user–item rating relations and item–item similarity relations. Fig. 4 demonstrates the structure of the user–item graph, where a pair of user and item (like user 1 and item 3) can be reached via multiple paths with different types of edges. The recommendation problem can therefore be transferred to a random walking model on the unique graph structure — to search the item nodes with the highest probabilities to be reached by a particular user node.

It is worth noticing that we cannot compare different types of edges directly, like rating relation and similarity relation that with different meanings or even different scales. This becomes a difficulty and highlight of this study, and we develop a unique random walking manner below.

Let us denote $U = \{u\}$ as the user set, $M = \{m\}$ the item set, $r(u, m)$ the rating value given by a user u to an item m, and $sim(m_1, m_2)$ the content similarity of two items m_1 and m_2, estimated by the above proposed taxonomy tree matching algorithm. We build a special graph $G = (V, E)$, where the vertex set $V = U \cup M$ is the union set of users and items, and the edge set $E = \{r(u \in U, m \in M)\} \cup \{sim(m_1 \in M, m_2 \in M)\}$ consists of two groups of relations: user–item ratings and item–item similarities. Given a particular user node u, the recommendation problem is defined as to determine an optimal ranking function $f : V \rightarrow \mathbb{R}$ to rank all vertices in the user–item graph. Ultimately, the top-ranked items are selected returned to the active user as recommendations.

4.2. Random walk manner

For the constructed graph with heterogeneous information of rating information and item similarities that are not comparable directly, we propose a unique random walk strategy as follows.

Given an active user u_0 for whom we need to make recommendations, we perform random walks on the user–item graph with this user node as the starter point. The idea is that after a long-term walking to convergence, the most visited items are considered as highly relevant to the active user. The random walk with restarting theory [31] is applied in our study. Because there are two different types of edges in the user–item graph, we elaborate the possible options of next move w.r.t the current visited vertex type.

First, if the current visited node is a user node $u \in U$, the options of next move include:

1. with probability α, it randomly moves to an item node linked to the current user u.
2. with probability $1 - \alpha$, it jumps back to the starting node (restarting the walk).

Here the parameter $\alpha \in (0, 1)$ is a decay factor that usually ranges from 0.8 to 0.85 for best performance as in previous studies.

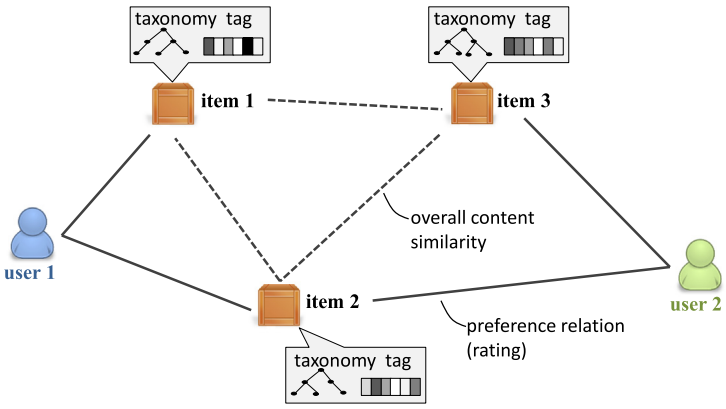


Fig. 4. Structure of the user–item graph constructed from heterogeneous information.

If option 1 is selected, the probability of moving to a particular item node $m \in M$ is simply:

$$P(m|u) = \frac{r(u,m)}{\sum_{i=1}^{|M|} r(u,m_i)}$$

Second, if the current visited node is an item node $m \in M$, the options of next move will be:

1. with probability α, it randomly moves to an adjacent node. Because item node m may be connected by two types of edges, i.e., linked to user nodes via rating relations or linked to item nodes via similarity relations, a Bernoulli switch variable s is introduced to control which type of edges is followed in the next move. So we have further two options in this case:
   (a) if $s = 0$, it randomly moves to a linked user node via rating relations. In this case, the probability of moving to a user node u is
   $$P(u|m) = \frac{r(u,m)}{\sum_{i=1}^{|U|} r(u_i,m)}$$
   (b) if $s = 1$, it randomly moves to a linked item node via similarity relations. In this case, the probability of moving to another item node n is
   $$P(n|m) = \frac{sim(m,n)}{\sum_{i=1}^{|M|} sim(m,m_i)}$$
2. with probability $1-\alpha$, it jumps back to the starting node.

Summing up the above walking strategy, if not jumping back, the transition probability between user and item nodes are as follows, where $P^{(t)}(u)$, $P^{(t)}(m)$ denotes the probability of being visited at time t for user node u and item node m, respectively.

For a user node u:

$$P^{(t+1)}(u) = \sum_{m\in M} P(u|m, s=0)P^{(t)}(m)P(s=0) \tag{9}$$

For an item node m:

$$P^{(t+1)}(m) = \sum_{n\in M} P(m|n, s=1)P^{(t)}(n)P(s=1) + \sum_{u\in U} P(m|u)P^{(t)}(u) \tag{10}$$

4.3. Recommendation making

The stationary visiting probabilities of all nodes in the user–item graph is considered as ranking them w.r.t to starting user node. Since our goal is to reveal the best fit items, we abstract the ranking results for only item nodes. Combining (9) and (10), we get the following update equation for items:

$$P^{(t+1)}(m) = \sum_{n\in M} P(m|n, s=1)P^{(t)}(n)P(s=1) + \sum_{u\in U} P(m|u) \sum_{n\in M} P(u|n, s=0)P^{(t-1)}(n)P(s=0) \tag{11}$$

The model can be written with matrix–vector notations. We denote the user–item rating matrix as R with $R_{ij} = r(u_i, m_j)$, and the item–item similarity matrix S with $S_{ij} = sim(m_i, m_j)$, and let the diagonal elements $S_{ii} = 0$. We also denote two diagonal matrices Π_u with size $|U| \times |U|$ and $\Pi_m$with size $|M| \times |M|$ representing the vertex degrees of users and items respectively in the rating matrix R, i.e., $(\Pi_u)_{ii} = \sum_{j=1}^{|M|} R_{ij}$ and $(\Pi_m)_{ii} = \sum_{j=1}^{|U|} R_{ji}$. Similarly, the degree matrix of items in the similarity matrix S is defined as a diagonal matrix D_m with size $|M| \times |M|$ and $(D_m)_{ii} = \sum_{j=1}^{|U|} S_{ij}$.

We suppose $P(s=1) = \beta$ and thus $P(s=0) = 1-\beta$, and denote a column vector $\mathbf{p}$ with length $|M|$ where each element $p_i = P(m_i)$, and let $\mathbf{p}^{(t)}$ represent the visiting probability distribution at time t.

Based on (11) and the restarting model, we obtain the update equation of $\mathbf{p}$ as follows.

$$\mathbf{p}^{(t+1)} = \alpha\left(\beta A\mathbf{p}^{(t)} + (1-\beta)B\mathbf{p}^{(t-1)}\right) + (1-\alpha)\mathbf{q} \tag{12}$$

Here, two matrices $A = S^T D_m^{-1}$ and $B = R^T \Pi_u^{-1} R \Pi_m^{-1}$ are defined to simplify expressions and $\mathbf{q}$ is the starting distribution.

Because the designed random walks start from a single active user such that no items are visited at beginning, we let $\mathbf{q} = \mathbf{p}^{(1)}$ as the starting distribution. That is, each element $q_i = r(u, m_i)/\sum_{j=1}^{|M|} r(u, m_j)$, where u is the active user requesting recommendations.

The stationary distribution $\mathbf{p}^*$ will gained when random walks reach convergence as proven in [34]. Let $\mathbf{p}^* = \mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} = \mathbf{p}^{(t-1)}$ and solve (12) with simple algebraic operations and we obtain

$$\begin{aligned}\mathbf{p}^* &= (1-\alpha)\left(I - \alpha\left(\beta A - (1-\beta)B\right)\right)^{-1}\mathbf{q} \\ &\triangleq \left(I - \alpha\left(\beta A - (1-\beta)B\right)\right)^{-1}\mathbf{q}\end{aligned} \tag{13}$$

Here the positive constant $(1-\alpha)$ is omitted because it does not affect ranking results. The top ranked unseen items for the active user will be suggested and recommendations are then completed.

5. Experiments

Experiments are conducted to compare our model with others with a public dataset of Movielens [35]. It is an extension of

the standard Movielens dataset with richer content information of items, therefore has been widely used for evaluating hybrid recommendation models.

5.1. Experiment setup

The dataset has 10109 movies with taxonomy information from five aspects – *genres*, *directors*, *actors*, *countries* and *locations*, and 46720 tags after necessary cleaning. The rating data is rich, in total 855k rating records, giving an average per user of 405 ratings. To evaluate the performances of recommendation models for sparse data, we extract different groups from the original rating data with different sparseness levels. The original dataset is denoted as Group 1, and then we randomly select half of ratings in it to build the Group 2 dataset, and then select half of Group 2 to build Group 3, and so on. We conduct this procedure five times and generate 6 data groups. The rating sparsity of these data groups are 96%, 98%, 99%, 99.5%, 99.7% and 99.9%, respectively. For the first data group, we split it to training set (80%) and test set (20%) randomly. For other data groups, we select 80% as training set, but still import the test set of group 1 for testing to prevent too small-size test sets. The items already being rated in the training set will be removed from the test set for each user. We conduct separate experiments on each of the six groups and apply five-folder validations (20% for test and 80% for training) for each.

The following related models are included for comparison. First, the memory-based CF is included. A model-based CF based on random walk is selected, which is called *ItemRank* [31]. Inspired by the works of [29,33], a multi-partite graph random walk model incorporating item taxonomy is implemented, and is named as *MultiWalk* for short. Another content-based model that integrates item taxonomy and tag information for recommendations [12] is included, named as *TagTax*. The hybrid model based on fuzzy tree matching techniques in [8] is imported, denoted as *FuzzTree*. A latest hybrid recommendation model based on user hierarchical preferences and social relations in [9] is also included, marked as *TreePref*. Notice that not explicit social relations are provided in the dataset, so we extract implicit social connections based on the tag data as input, like in previous tag-based recommendation studies [10,30]. The last, our hybrid model based on taxonomy and folksonomy is marked as *TFhybrid* for short. We can summarize these models into two groups — prediction-based models that aim to predict the ratings to each item separately (CF, TagTax, FuzzTree, TreePref) and ranking-based model that aim to derive a ranking order for all items (ItemRank, Multi Walk, TFhybrid).

5.2. Metrics

If rating data is insufficient, recommendation models may fail to make predictions, therefore it is needed to evaluate the ability of compared models in alleviating the rating sparseness problem. In our experiments, the task for each model is to predict the rating between each pair of users and items in the test set. The coverage metric is selected for evaluation, which is defined as the number of successfully predicted tasks divided by the total number of prediction tasks in the test set, as follows:

$$coverage = \frac{\#\text{ successful prediction tasks}}{\#\text{ total prediction tasks}} \tag{14}$$

Because the ranking based models do not predict rating scores directly, it is not able to calculate the prediction errors like MAE (Mean Absolute Error) and RMSE (Root Mean Square Error). The NDCG metric (*Normalized Discounted Cumulative Gain*) is hence selected to evaluate ranking accuracy for compared models. For each model, the ranking order of items according to predicted scores is compared with the ideal ranking order according to the actual ratings. The metric DCG (*Discounted Cumulative Gain*) is defined as follows. Given a ranking order where the actual ranking score at position i is rel_i, the DCG value at a specific position p is calculated as:

$$\text{DCG@}p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \tag{15}$$

The normalized DCG value (NDCG) is then defined as the prediction result of a model divided by the ideal (actual) result, calculated as: $\text{NDCG@}p = \text{DCG@}p/\text{IDCG@}p$.

One limitation of the NDCG metric for recommendation problems is that we cannot obtain the actual ratings of each user to all items. Most existing studies [36] commonly set the ranking score as 1 for the items in the test set and 0 for others. This will result in too small NDCG values because the test set is usually a very small portion of the whole item set. Again, it is unfair for the prediction-based models that are not optimized for list-wise ranking for all items. In our experiment, we therefore evaluate the NDCG metrics only for the items in the test set. The ideal ranking score is set to be the rating divided by 5 (the maximum rating). Equivalently, we evaluate whether a model can rank the known items in the test set correctly.

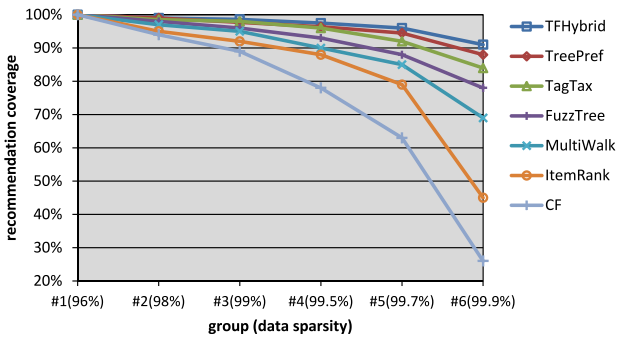


Fig. 5. Recommendation coverage comparison under different sparsity levels.

5.3. Results

We test each model with the generated data groups. For the random walk models (ItemRank, MultiWalk and TFhybrid) sharing a similar parameter α, we let $\alpha = 0.8$ initially. In addition, the parameter β for our model is set to be 0.5. Fig. 5 shows the result of recommendation coverage for each model on each data group.

It indicates that recommendation coverage is highly impacted by the sparseness level of rating data. For the initial dataset with the richest rating data (averagely 405 ratings per user), every model achieves the highest recommendation coverage — almost 100%. For other data groups with sparser ratings, however, these models fail to complete all prediction tasks, i.e., they are not able to generate recommendations for the "cold-start" users with insufficient ratings. In detail, we can find that the scores of single-source based models (CF and ItemRank) decrease sharply, while hybrid models can alleviate this problem to some extent. Our model maintains the highest scores on all data groups. This demonstrates the success of integrating other resources like tree-structured content and tag information in alleviating the rating sparseness problem.

Table 1 shows the performance comparisons in terms of ranking accuracy . Each model is tuned with its best parameter settings, and we calculate the NDCG for all items in the test set.

Table 1
Comparison on ranking accuracy (NDCG).

	#1 96%	#2 98%	#3 99.0%	#4 99.5%	#5 99.7%	#6 99.9%
CF	**0.737**	0.647	0.463	0.505	0.356	0.128
ItemRank	0.633	0.565	0.467	0.419	0.286	0.118
MultiWalk	0.659	0.555	0.507	0.519	0.376	0.277
FuzzTree	0.727	0.597	0.491	0.479	0.434	0.326
TagTax	0.711	0.577	0.487	0.459	0.406	0.316
TreePref	0.730	0.645	0.556	0.576	0.504	0.407
TFhybrid	0.731	**0.660**	**0.581**	**0.615**	**0.543**	**0.439**
improv.	−0.81%	2.01%	4.50%	6.77%	7.74%	7.86%

From the results, we can find that the proposed model TFhybrid achieves the best ranking performance for almost all data groups, especially in sparse environment, e.g., the last group with the most sparse rating data. For other models, the pure CF has high accuracy when the ratings are rich, as in the first group, but it loses the superiority quickly when ratings became increasingly sparse. As another single-source based model, ItemRank suffers the same problem and performs the worst on sparse datasets. We can find that hybrid models incorporate multiple sources of information generally outperform single source-based models when the data are sparse, especially for the models incorporating both taxonomy and folksonomy information (TagTax, TreePref, and TFhybrid).

With the above comparisons, we can conclude that the proposed hybrid model is able to improve recommendation performance in terms of both recommendation coverage and recommendation accuracy. The incorporation manner for item taxonomy and folksonomy in this study is believed to be effective in alleviating the rating sparseness problem.

6. Conclusion and future study

This paper proposed a hybrid ecommerce recommendation approach based on a random walk model of users and products and various relations between them. Heterogeneous information is utilized to build the user–item graph as input, including item taxonomy information and folksonomy information and user rating data on items. A novel tree matching algorithm is developed to compare the overall similarity between item taxonomy. Tag information is integrated for semantic analysis for taxonomy attributes. With the established item–item similarity relations and user–item rating relations, a heterogeneous graph for users and items can be constructed, and a random walk strategy for the unique graph data is developed to determine nearest item nodes for given user nodes as recommendations, which can be seen as a hybrid recommendation model that integrates both CB and CF ideas. The empirical experiments demonstrate that our model is able to improve recommendation performance in terms of coverage and accuracy, especially in the case of sparse data. Comparing with related models, the improvements also indicate that the proposed tree matching algorithm is effective to incorporate item taxonomy and folksonomy information to estimate precise correlations between items. One of the limitations in this study is that context information is not included. In the future study, we would like to incorporate context information like social networks, location and temporal information to enhance our hybrid model.

CRediT authorship contribution statement

Mingsong Mao: Conceptualization, Formal analysis, Writing - original draft. **Sihua Chen:** Data curation, Project administration, Funding acquisition. **Fuguo Zhang:** Investigation, Methodology. **Jialin Han:** Methodology, Writing - review & editing. **Quan Xiao:** Resources, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is partially supported by National Nature Science Foundation of China (No. 61802156, 71861013, 61772245, 71764006).

References

[1] S. Sivapalan, A. Sadeghian, H. Rahnama, A.M. Madni, Recommender systems in e-commerce, in: 2014 World Automation Congress (WAC), 2014, pp. 179–184.
[2] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: A survey, Decis. Support Syst. 74 (2015) 12–32.
[3] M. Zhou, Z. Ding, J. Tang, D. Yin, Micro Behaviors: A New Perspective in E-commerce Recommender Systems, in: The Eleventh ACM International Conference, 2018, pp. 727–735.
[4] C.C. Aggarwal, Content-based recommender systems, in: Recommender Systems: The Textbook, Springer International Publishing, Cham, 2016, pp. 139–166.
[5] C.C. Aggarwal, Neighborhood-based collaborative filtering, in: Recommender Systems: The Textbook, Springer International Publishing, Cham, 2016, pp. 29–70.
[6] J.K. Tarus, Z. Niu, D. Kalui, A hybrid recommender system for e-learning based on context awareness and sequential pattern mining, Soft Comput. 22 (8) (2018) 2449–2461.
[7] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, L. Getoor, Personalized explanations for hybrid recommender systems, in: Proceedings of the 24th International Conference on Intelligent User Interfaces, 2019, pp. 379–390.
[8] D. Wu, G. Zhang, J. Lu, A fuzzy preference tree-based recommender system for personalized business-to-business e-services, IEEE Trans. Fuzzy Syst. 22 (99) (2014) 29–43.
[9] J. Zheng, S. Wang, D. Li, B. Zhang, Personalized recommendation based on hierarchical interest overlapping community, Inform. Sci. 479 (2019) 55–75.
[10] T. Bogers, Tag-based recommendation, in: P. Brusilovsky, D. He (Eds.), Social Information Access: Systems and Technologies, Springer International Publishing, Cham, 2018, pp. 441–479.
[11] H. Xue, B. Qin, T. Liu, S. Liu, Tag recommendation based on topic hierarchy of folksonomy, Int. J. Comput. ence Eng. 20 (1) (2019).
[12] H. Liang, Y. Xu, Y. Li, R. Nayak, Personalized recommender system based on item taxonomy and folksonomy, in: The 19th ACM International Conference on Information and Knowledge Management, ACM, New York, NY, USA, 2010, pp. 1641–1644.
[13] D. Wang, Y. Liang, D. Xu, X. Feng, R. Guan, A content-based recommender system for computer science publications, Knowl.-Based Syst. 157 (2018) 1–9.
[14] Y. Shi, M. Larson, A. Hanjalic, Collaborative filtering beyond the user-item matrix: a survey of the state of the art and future challenges, ACM Comput. Surv. 47 (1) (2014) 3:1–3:45.
[15] Q. Zhang, J. Lu, D. Wu, G. Zhang, A cross-domain recommender system with kernel-induced knowledge transfer for overlapping entities, IEEE Trans. Neural Netw. Learn. Syst. 30 (7) (2019) 1998–2012.
[16] Y. Wang, J. Deng, J. Gao, P. Zhang, A hybrid user similarity model for collaborative filtering, Inform. Sci. 418–419 (2017) 102–118.
[17] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, N. Xiong, Deep matrix factorization with implicit feedback embedding for recommendation system, IEEE Trans. Ind. Inf. 15 (8) (2019) 4591–4601.
[18] M. Al-Ghossein, P.-A. Murena, T. Abdessalem, A. Barré, A. Cornuéjols, Adaptive collaborative topic modeling for online recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, New York, NY, USA, 2018, pp. 338–346.
[19] F. Shang, Y. Liu, J. Cheng, D. Yan, Fuzzy double trace norm minimization for recommendation systems, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 2039–2049.
[20] W. Chen, F. Cai, H. Chen, M.D. Rijke, Joint neural collaborative filtering for recommender systems, ACM Trans. Inf. Syst. 37 (4) (2019) 1–30.
[21] F. Xiong, X. Wang, S. Pan, H. Yang, H. Wang, C. Zhang, Social recommendation with evolutionary opinion dynamics, IEEE Trans. Syst. Man Cybern. (2018) 1–13.
[22] G. Adomavicius, A. Tuzhilin, 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) (2005) 734–749.

[23] Q. Shambour, J. Lu, A trust-semantic fusion-based recommendation approach for e-business applications, Decis. Support Syst. 54 (1) (2012) 768–780.
[24] E. Aslanian, M. Radmanesh, M. Jalili, Hybrid recommender systems based on content feature relationship, IEEE Trans. Ind. Inf. (2016) 1–1.
[25] M. Volkovs, G.W. Yu, T. Poutanen, Content-based neighbor models for cold start in recommender systems, in: Proceedings of the Recommender Systems Challenge 2017, in: RecSys Challenge '17, Association for Computing Machinery, New York, NY, USA, 2017.
[26] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, L. Getoor, User preferences for hybrid explanations, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, in: RecSys '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 84–88.
[27] D. Lian, Y. Ge, F. Zhang, N.J. Yuan, X. Xie, T. Zhou, Y. Rui, Scalable content-aware collaborative filtering for location recommendation, IEEE Trans. Knowl. Data Eng. 30 (6) (2018) 1122–1135.
[28] A. Arwan, B. Priyambadha, R. Sarno, M. Sidiq, H. Kristianto, Ontology and semantic matching for diabetic food recommendations, in: The 5th International Conference on Information Technology and Electrical Engineering, 2013, pp. 170–175.
[29] X. He, M. Gao, M.Y. Kan, D. Wang, Birank: Towards ranking on bipartite graphs, IEEE Trans. Knowl. Data Eng. 29 (1) (2016) 57–71.
[30] M. Mao, J. Lu, G. Zhang, J. Zhang, Multirelational social recommendations via multigraph ranking, IEEE Trans. Cybern. 47 (12) (2017) 4049–4061.
[31] M. Gori, A. Pucci, Itemrank: A random-walk based scoring algorithm for recommender engines, in: The 20th International Joint Conference on Artifical Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007, pp. 2766–2771.
[32] S. Lee, S.-i. Song, M. Kahng, D. Lee, S.-g. Lee, Random walk based entity ranking on graph for multidimensional recommendation, in: The 5th ACM Conference on Recommender Systems, ACM, Chicago, IL, USA, 2011, pp. 93–100.
[33] M. Mao, J. Lu, J. Han, G. Zhang, Multiobjective e-commerce recommendations based on hypergraph ranking, Inform. Sci. 471 (2019) 269–287.
[34] K.B. Athreya, H. Doss, J. Sethuraman, On the convergence of the Markov chain simulation method, Ann. Statist. 24 (1) (1996) 69–100.
[35] I. Cantador, P. Brusilovsky, T. Kuflik, Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011), in: The 5th ACM Conference on Recommender Systems, Chicago, IL, USA, 2011, pp. 387–388.
[36] Q. Cui, S. Wu, Y. Huang, L. Wang, A hierarchical contextual attention-based network for sequential recommendation, Neurocomputing 358 (2019) 141–149.