A Fuzzy Preference Tree-Based Recommender System for Personalized Business-to-Business E-Services

Dianshuang Wu, Guangquan Zhang, and Jie Lu, *Senior Member, IEEE*

***Abstract*—The Web creates excellent opportunities for businesses to provide personalized online services to their customers. Recommender systems aim to automatically generate personalized suggestions of products/services to customers (businesses or individuals). Although recommender systems have been well studied, there are still two challenges in the development of a recommender system, particularly in real-world B2B e-services: 1) items or user profiles often present complicated tree structures in business applications, which cannot be handled by normal item similarity measures and 2) online users' preferences are often vague and fuzzy, and cannot be dealt with by existing recommendation methods. To handle both these challenges, this study first proposes a method for modeling fuzzy tree-structured user preferences, in which fuzzy set techniques are used to express user preferences. A recommendation approach to recommending tree-structured items is then developed. The key technique in this study is a comprehensive tree matching method, which can match two tree-structured data and identify their corresponding parts by considering all the information on tree structures, node attributes, and weights. Importantly, the proposed fuzzy preference tree-based recommendation approach is tested and validated using an Australian business dataset and the MovieLens dataset. Experimental results show that the proposed fuzzy tree-structured user preference profile reflects user preferences effectively and the recommendation approach demonstrates excellent performance for tree-structured items, especially in e-business applications. This study also applies the proposed recommendation approach to the development of a web-based business partner recommender system.**

***Index Terms*—E-business, fuzzy preferences, recommender systems, tree matching, web-based support system.**

I. Introduction

With the use of recommendation methods, recommender systems [1], [2], which are web-based support systems, actively suggest a set of limited and ranked items from all available items without the direct input of users. These systems are widely used to overcome the problems created by the so-called "information explosion" in a variety of web-based applications in e-commerce [3], e-learning [4], and e-tourism, as well as in such areas as the recommendation of news, movies, books, videos, resources [5], and real estate [6]. Prior to making a recommendation, recommender systems use background data, such as historical data consisting of ratings from users, and input data, such as features of items or user ratings, to initiate a recommendation; models and algorithms combine the two and generate a recommendation [7], [8].

In real situations, the features of items and user behavior are often subjective, vague, and imprecise [8], and users' item preferences are frequently subjective and uncertain. It is difficult for a user to express his/her interest in an item with exact numbers. Fuzzy set theory and techniques lend themselves well to handling the fuzziness and uncertain issues in recommendation problems. User preferences and item features have been represented as fuzzy sets in previous research [8]–[11], and recommendations to customers for the selection of the most suitable items are made with incomplete and uncertain information [12], [13]. Current research and recommender system applications focus mainly on making recommendations to personal users. Fuzzy user preference and item representations focus on vector representations accordingly.

The abundance of information created and delivered via the Web provides excellent opportunities for the development of business-to-business (B2B) e-services, such as finding a business partner online [14]. Excessive amounts of information on the Web create a severe information overload problem. An effective solution for this problem is the development of personalized recommender systems; however, recommendation techniques have been rarely used in the B2B environment. The main reason is that items or user profiles in a B2B environment are so complex that they can only be presented as complicated structures, such as tree structures. For example, a business in a B2B application environment may supply several product categories, each of which may contain a number of subcategories, under which there may be multiple specific products, which together form a tree structure. Therefore, tree-structured data modeling and tree matching methods are needed. However, an item is normally described as a single value or a vector in current research, and tree-structured items or user profiles have not been considered to date. The fuzzy preference models mentioned previously, which are represented as vectors, are not suitable to dealing with the tree-structured data in a Web-based B2B environment.

To solve these challenges—namely, tree-structured items (products/services), tree-structured user preferences,

Manuscript received August 27, 2013; revised January 6, 2014; accepted March 11, 2014. Date of publication April 4, 2014; date of current version January 30, 2015. This work was supported in part by the Australian Research Council under discovery Grant DP110103733.

The authors are with Decision Systems and e-Service Intelligence Laboratory, Centre for Quantum Computation & Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia (e-mail: Dianshuang.Wu@student.uts.edu.au; Guangquan.Zhang@uts.edu.au; Jie.Lu@uts.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2014.2315655

vague values of user preferences, and personalization of recommendations—in B2B e-service recommendation problems, this study proposes a method for modeling fuzzy tree-structured user preferences, presents a tree matching method, and, based on the previous methods, develops an innovative fuzzy preference tree-based recommendation approach. The developed new approach has been implemented and applied in a business partner recommender system.

This paper has three main contributions. From the theoretical aspect, a tree matching method, which comprehensively considers tree structures, node attributes, and weights, is developed. From the technical aspect, a fuzzy tree-structured user preference modeling method is developed, as well as a fuzzy preference tree-based recommendation approach for tree-structured items. From the practical aspect, the proposed methods/approaches are used to develop a Web-based B2B recommender system software known as Smart BizSeeker, with effective results.

The remainder of the paper is organized as follows. In Section II, the related works in recommender systems, tree matching methods, and fuzzy set techniques are expatiated. Section III presents the fuzzy tree-structured preference model. Section IV proposes a comprehensive tree matching algorithm to identify the corresponding parts between two trees. The fuzzy preference tree construction algorithm is proposed in Section V. A fuzzy preference tree-based recommendation approach for tree-structured items is presented in Section VI. The approach has been tested using the Australian business dataset and MovieLens dataset. The experimental evaluations and results are given in Section VII. In Section VIII, the proposed recommendation approach is implemented in a recommender system software-Smart BizSeeker-to help businesses find partners (suppliers or buyers). Finally, the conclusions and future study are given in Section IX.

II. Related Works

This section will review the literature on recommender systems, tree matching methods, and fuzzy set techniques in recommender systems.

A. Recommender Systems

Recommendation techniques have attracted much attention and many recommendation approaches have been proposed. The three main recommendation techniques are collaborative filtering (CF), content-based (CB) and knowledge-based (KB) techniques [7]. The CF technique is currently the most successful and widely used technique for recommender systems [15], [16]. It helps people to make choices based on the opinions of other people who share similar interests [17]. The CF technique can be further divided into user-based and item-based CF approaches [18]. The major limitations of CF methods are data sparsity and cold-start (CS) problems [1], [16], [18]. The data sparsity problem occurs when the number of available items increases and the number of ratings in the rating matrix is insufficient to generate accurate predictions. When the ratings obtained are very small compared with the number of ratings that need to be predicted, a recommender system becomes incapable of locating similar neighbors and produces poor recommendations. The CS problem consists of the CS user problem and the CS item problem. The CS user problem, also known as the new user problem, affects users who have a small number of ratings or none. When the number of rated items for the CS user is small, the CF-based approach cannot accurately find user neighbors using rating similarity; therefore, it fails to generate accurate recommendations. The CS item problem, also known as the new item problem, affects items that have only a small number of ratings or none. With only a few ratings for CS items, CF-based approaches cannot appropriately locate similar neighbors using rating similarity and will be unlikely to recommend them [1], [19]. CB recommendation techniques recommend items that are similar to those previously preferred by a specific user [20]. The major limitations of CB approaches are the item content dependence problem, overspecialization problem, and new user problem [1], [20]. The KB recommender systems offer items to users based on knowledge about the users and items [21]. In contrast with the CF and CB approaches, KB approaches have no CS problems because a new user can obtain recommendations based on the knowledge of his/her interests. KB approaches are, in the majority of cases, applied to recommend complex products and services such as consumer goods, technical equipment, or financial services [21]. The KB approach has some limitations, however, for instance, the KB approach needs to retain some information about items and users, as well as functional knowledge, to make recommendations. It also suffers from the scalability problem because it requires more time and effort to calculate the similarities in a large case base than other recommendation techniques. Each recommendation technique has its own merits and drawbacks, thus hybrid recommendation techniques have been proposed to achieve higher performance while avoiding the drawbacks of typical recommendation techniques [22]. The most common practice in existing hybrid recommendation techniques is to combine CF with other recommendation techniques in an attempt to avoid the problems of the CF approaches [1], [23].

The underlying semantic properties and attributes associated with users and items have been exploited to generate recommendations in certain types of recommender systems called semantic-based recommender systems [24]. The semantic information about items consists of the attributes of the items, the relation between items, and the relation between items and meta-information [25]. Taxonomies and ontologies as the major source of semantic information can be used to advantage in recommender systems, since they provide a means of discovering and classifying new information about the items to recommend, user profiles and even context [24]. For example, product taxonomies and ontologies have been presented in several recommender systems to utilize relevant semantic information to improve recommendation quality [26]–[28]. In a business environment, product categories are used to evaluate the semantic similarity between businesses [2], [14], [29]. The usage of the semantic information can provide additional explanation as to why particular items have or have not been recommended, and

provide better recommendation effectiveness than current CF techniques, particularly in cases where little or no rating information is available [29]. In this study, the attribute information of tree-structured business data should be fully considered to make accurate recommendations.

B. Tree Matching

Tree-structured data are widely used in many application fields [30]–[34]. The tree similarity measure [35], tree isomorphism [36], and subtree matching problems [34], [37] have been researched during the applications of tree-structured data. The tree edit distance model [37] is the most widely used method to compare the structures of ordered or unordered labeled trees. The model compares two trees with the minimum cost of the edit operation sequences that convert one tree into another. The edit operations give rise to an edit distance mapping which is a graphical specification of which edit operations apply to each node in the two labeled trees [38]. The tree edit distance model is also used in tree isomorphism and subtree matching problems [37]. In the aforementioned researches, only tree structures and node labels are considered, which is insufficient for use in the application of the B2B e-service recommendation. In our previous research, an edit distance mapping between two tree-structured data was constructed that considered tree structures, node attributes, and weights [39], [40]. A number of tree matching methods have been used in recommender systems. In [41], users' behavior patterns were represented as tree structures, and tree matching was used to find the correlation between different behavior patterns. In the telecom industry, products/services for business users usually have complex hierarchical structures; therefore, tree structured data have been used to represent items and user profiles [40], [42]. Tree matching methods have been developed to evaluate the similarity between tree-structured items or users. However, these methods are unable to construct fuzzy tree-structured user preferences and match items to user preferences. In this study, to match tree-structured items and user preferences in a B2B e-service environment, a new tree matching method is developed based on tree edit distance mapping.

C. Fuzzy Set Techniques in Recommender Systems

Fuzzy set theory offers a rich spectrum of methods for the management of nonstochastic uncertainty. It is well suited to handling imprecise information, the unsharpness of classes of objects or situations, and the gradualness of preference profiles [8], [43]. In [10], an item is represented as a fuzzy set over an assertion set. The value of a feature or attribute for an item is a fuzzy set over the subset of the assertions relevant to the feature. The user's intentional preferences are represented as a basic preference module, which is the ordered weighted averaging of components that can evaluate items. The user's extensional preferences are expressed as a fuzzy set over the user's experienced items whose membership degrees are the ratings. Based on the representation, the preference for an item by a user can be inferred. Zenebe *et al.* [8], [9] defined a feature set for items and a set of values for each feature. The items are represented as the fuzzy subset over the values, denoted by a feature vector. Four kinds of fuzzy-set-based similarity measures: fuzzy set theoretic, cosine, proximity, and correlation-like, are introduced. Cao and Li [44] used linguistic terms for domain experts to evaluate the features of consumer electronic products and allow users to use linguistic terms to express their needs for item features. In [12], the user preferences are represented as two fuzzy relations, positive and negative feelings, from user set to item set. The item similarity is computed by integrating CB similarity, which is a fuzzy relation within an item set, and item-based CF similarity, which is computed on the basis of user preferences. The user similarity is generated by fuzzy relational calculus from the preferences and item similarity relations. The final recommendations, which are the positive and negative preferences, are generated by composing the aforementioned fuzzy relations. Porcel *et al.* [13] developed a fuzzy linguistic-based recommender system based on both CB filtering and the multigranular fuzzy linguistic modeling technique, which is useful to assess different qualitative concepts. Fuzzy similarity measures and fuzzy matching methods have been used in previous research, but to the best of our knowledge, no research has focused on solving fuzziness problems in tree-structured data.

III. Fuzzy Tree-Structured Preference Model

This section describes the representation of fuzzy tree-structured user preferences. Fuzzy set techniques are used to model user preferences; a formal tree-structured data model is given, and a fuzzy tree-structured user preference model is then presented.

A. Users' Fuzzy Preferences

To make a recommendation to a user, the information about the user's preferences must be known. The modeling method for user's preferences is presented in this section.

Information about user preferences can essentially be obtained in two different ways: extensionally and intentionally [10]. The extensionally expressed preference information refers to information that is based on the actions or past experiences of the user with respect to specific items. The intentionally expressed preference information refers to specifications by the user of what they desire in the items under consideration. In this paper, the user preference model covers both kinds of information.

In the practice of recommender systems, a business user's preferences are usually complex and vague. It might be difficult to require a business user to express a crisp preference for an item or a feature of an item, and it is therefore difficult to represent the user's preferences with crisp numbers. In this study, fuzzy set techniques are used to describe users' complex and vague preferences.

It is assumed that a totally ordered set $R = \{1, 2, \ldots, r\}$ is predefined to represent the crisp values of ratings. A user u's preference for an item (or feature) j is represented as a fuzzy set over $R, \tilde{p}_{uj} = \{f_{1,uj}/1, f_{2,uj}/2, \ldots, f_{r,uj}/r\}$, where each $f_{i,uj} \in [0, 1]$ represents the membership degree of the rating i.

$\tilde{p}_{uj}$ will be expressed as $\{f_{1,uj}, f_{2,uj}, \ldots, f_{r,uj}\}$, if there is no confusion. For example, supposing that the crisp ratings are on a scale of 1 to 5, with 1 being the lowest rating and 5 the highest rating, a user's preference for an item is represented as a fuzzy subset on $\{1, 2, 3, 4, 5\}$ by membership degree [0, [1]. The preference value (0/1, 0/2, 0/3, 0.9/4, 1/5) indicates that a user likes an item very much by the high membership degree (1) on rating value "5" as well as the very high membership degree (0.9) on "4," while the preference value (1/1, 0/2, 0/3, 0/4, 0/5) indicates that the user does not like the item at all by the high membership degree (1) on rating value "1" and the low membership degree (0) on the other rating values.

The items considered in this study are presented as tree structures, and the features of items form a hierarchical structure. A business user's preferences concern a set of products/features, and user preference is therefore described as a tree structure which has fuzzy preference values. To formally describe the tree-structured items and user preferences, a tree-structured data model is defined as follows.

B. Tree-Structured Data Model

A tree-structured data model is defined to represent tree-structured items or user preferences. It is based on the basic tree definition, which is given as follows.

Definition 1 [45]: A tree is defined as a directed graph $T = (V, E)$ where the underlying undirected graph has no cycles and there is a distinguished root node in V, denoted by $\mathrm{root}(T)$, so that for any node $v \in V$, there is a path in T from $\mathrm{root}(T)$ to node v.

In real applications, the definition is usually extended to represent practical objects. In this research, a tree-structured data model is proposed by adding the following features to the definition.

1) Node Attribute: Nodes in a tree are assigned semantic meanings. A domain attribute term set A, which is a set of symbols to specify the semantic meanings of nodes, is introduced. There exists an attribute assignment function $a : V \to A$ so that each node in the tree is assigned an attribute. The attribute terms can be divided into basic attributes and complex attributes. A complex attribute represents the semantic concept combined with several basic attributes. A basic attribute is a unary variable.

2) Attribute Conceptual Similarity: An attribute conceptual similarity measure within the domain attribute term set A is defined as a set of mappings $sc : A \times A \to [0, 1]$, in which each mapping denotes the conceptual similarity between two attributes [31]. For any $a_1, a_2 \in A$, we say a_1 and a_2 are similar if $sc(a_1, a_2) > \varepsilon$, where ε is the threshold of a similar relation. The larger $sc(a_1, a_2)$ is, the more similar are the two attributes. The conceptual similarity measures can be given by domain experts or inferred from the domain ontology that describes the relations between the attributes.

3) Node Value: Each node can be assigned a value to represent some kind of degree of the attribute relevant to the node. The value type is an abstract type in the definition, which can be instantiated in a specific application.

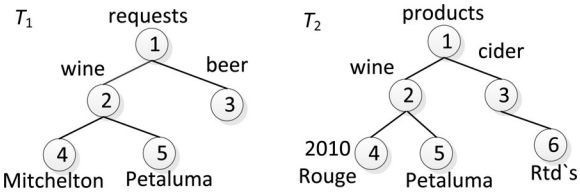


Fig. 1. Two tree-structured data examples.

4) Node Weight: A weight function $w : V \to [0, 1]$ assigns a weight to each node to represent its importance degree in relation to its siblings.

Two examples are given to explain these features.

Fig. 1 shows two tree-structured data in a business environment, which are called two trees for simplicity. Tree T_1 is a user's buying request. The user requires two kinds of product, wine and beer. For the wine request, two products are specified. Tree T_2 is a product tree of a wine business. It provides two product categories, wine and cider. For each category, specific products are given. The two examples contain the four features listed in the aforementioned tree-structured data definition. For Feature 1—Node attribute, a domain attribute term set which includes product category names, product names, product feature names, and so on exists. For Feature 2—Attribute conceptual similarity, conceptual relations between these attributes exist, such as attribute conceptual similarity measures [31], to express the similarity between the attributes. For Feature 3—Node value, the values are assigned to the trees. For example, the values of the product tree can be the quality or the quantity of the attributes. For Feature 4—Node weight, different nodes/attributes may have different importance degrees in real applications, which are reflected by the weights of the nodes. It can be seen from Fig. 1 that the two tree-structured data T_1 and T_2 have different structures and node attributes, which makes it difficult to identify the corresponding node pairs between them.

C. Fuzzy Tree-Structured User Preference

A user's preference is represented as the tree-structured data model defined previously. It is called a fuzzy preference tree and is defined as follows.

Definition 2: The fuzzy preference tree of a user is tree-structured data whose node values are the user's fuzzy preferences for the corresponding attributes.

Each subtree in a user's fuzzy preference tree represents the user's preference for one aspect of the features, and the subtrees of that aspect represent the user's preferences for the finer features. The leaf nodes represent the preferences for the finest features. The fuzzy preference tree has a similar structure to the item tree except for the node value definition. The value of a node in an item tree represents the numeric degree of the attribute associated with the node, while the value of a node in a fuzzy preference tree represents the user's preference for the attribute represented by the node. The node value of the fuzzy preference tree contains two components. One is the preference $\tilde{p}_u$, which is expressed with a fuzzy set; the other is a count number count, which indicates the number of preference

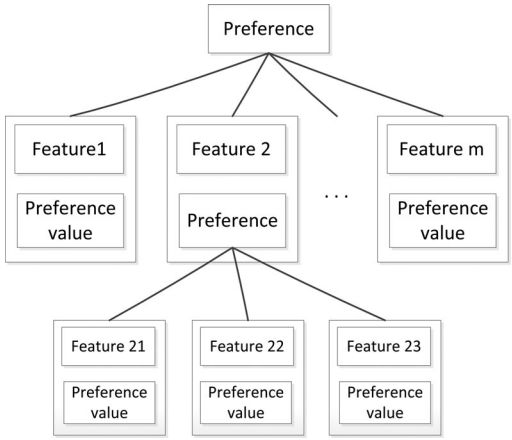


Fig. 2. Intentionally expressed user preference.

sources used to calculate the value. The count is used to incrementally update the user's fuzzy preference tree, as shown next.

1) Intentionally Expressed Preference: The intentionally expressed preference is acquired directly from users. This kind of information is especially important for new users to obtain recommendations.

Because the item features present tree structures, the preferences given by users are in tree structures, as shown in Fig. 2. To express preferences, a user selects several features. For example, *Feature* 1, *Feature* 2, . . . , *Feature m* are selected in Fig. 2. For each feature, there are two situations. First, the user can assign a preference value, such as *Feature* 1 in Fig. 2. Second, the user can drill down to detail and express preferences for finer features under the macro feature, as shown for *Feature* 2. Therefore, users' preference values, which are represented as fuzzy sets, can be expressed at different levels. For different features, the user can also specify various weights to express the different importance degrees of diverse features.

A user's fuzzy preference tree T_u is constructed based on user input preferences. The tree has the same structure as the user preferences shown in Fig. 2. The tree node attributes are the relevant features. If the user expresses the preference value for a feature, then the value will be assigned to the relevant node accordingly. The node weights are also set according to the user's input.

2) Extensionally Expressed Preference: The extensionally expressed preference of a user is constructed from the items experienced by the user. Let the items experienced by a user u be the set $EI_u = \{i_1, i_2, \ldots, i_m\}$. Each item i_j $(j = 1, 2, \ldots, m)$ corresponds to an item tree $T_{i,j}$ and a preference value given by the user $\tilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \ldots, f_{r,uj}\}$. Let the user's fuzzy preference tree be T_u. The construction process of T_u is presented as follows.

For each item i_j experienced by user u with preference value $\tilde{p}_{uj}$, add the item tree $T_{i,j}$ into the fuzzy preference tree T_u. The add operation integrates the user's preference for an item into T_u. When all the items experienced are added into T_u, the user's fuzzy preference tree T_u will be constructed. The fuzzy preference tree construction algorithm is described in detail in Section V. During the process, the conceptual corresponding parts between two trees considering tree structures, node attributes, and weights comprehensively must be identified. Therefore, a tree-structured data matching algorithm is presented first in the following section.

IV. Tree-Structured Data Matching Algorithm

Based on the tree-structured data model, this section proposes a tree matching algorithm to construct a map to identify the parts of two trees that most correspond. The proposed tree matching algorithm will be used to construct the user's fuzzy preference tree and match user preferences and items in the course of making recommendations.

Let two trees to be matched be denoted as T_u and T_i. A maximum conceptual similarity tree mapping [39], which is a kind of edit distance mapping, is constructed to identify the parts of the two trees that most correspond conceptually. When constructing the mapping, tree structures, node attributes, and node weights are all taken into consideration.

It should be noted that in contrasting application scenarios, the requirements to match two trees are different. For example, when matching two item trees, the weights of both trees should be considered. In another situation, when matching a target item tree to a user's fuzzy preference tree to make recommendations, the user's fuzzy preference tree should mainly be weighted. Therefore, the matching method should consider the two situations, respectively. In the former situation, the matching is called symmetric matching, while in the latter situation the matching is called asymmetric matching.

In the following section, the symbols in [46] are used to represent trees and nodes. Suppose that we have a numbering for each tree node in a tree. Let $t[j]$ be the jth node of the tree T in the given numbering. Let $T[j]$ be the subtree rooted at $t[j]$ and $F[j]$ be the unordered forest obtained by deleting $t[j]$ from $T[j]$. Let $t[j_1], t[j_2], \ldots, t[j_{n_j}]$ be the children of $t[j]$.

The maximum conceptual similarity tree mapping maps the most conceptually similar parts of two trees. This mapping can be constructed during the computation of the conceptual similarity between two trees. Let S_T be the set of trees to be compared. A conceptual similarity between two trees is defined as a set of mappings $sc_T : S_T \times S_T \rightarrow [0, 1]$, in which each mapping denotes the conceptual similarity between the corresponding two trees. The conceptual similarity also has two types, symmetric and asymmetric, depending on the matching types. They are denoted as $sc_{T_{\mathrm{sym}}}$ and $sc_{T_{\mathrm{asym}}}$ when the matching type needs to be specified.

The conceptual similarity between two trees is calculated as follows.

Given two trees $T_u[j]$ and $T_i[k]$ to be compared, according to the matching situations of their roots $t_u[j]$ and $t_i[k]$, three cases are considered: $t_u[j]$ and $t_i[k]$ are matched; $t_u[j]$ is matched to $t_i[k]$'s child; $t_i[k]$ is matched to $t_u[j]$'s child. The matching situation with the maximum conceptual similarity value is the best match and the relevant similarity value is taken as the conceptual similarity between the two trees.

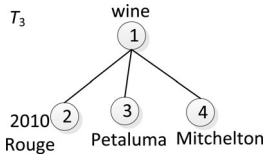


Fig. 3. Tree-structured data example.

In the case where $t_u[j]$ and $t_i[k]$ are matched, the conceptual similarity between $T_u[j]$ and $T_i[k]$ is calculated as (1), shown at the bottom of the page, where $a(t_u[j])$ and $a(t_i[k])$ represent the attributes of $t_u[j]$ and $t_i[k]$, respectively, w_{k_t} and w_{j_t} are the weights of $t_i[k_t]$ and $t_u[j_t]$, respectively, and α is the influence factor of the parent node. Four situations are listed in Formula (1), according to the condition of whether $t_u[j]$ and $t_i[k]$ are leaves.

In situation $S1$, $t_u[j]$ and $t_i[k]$ are both leaves, and their conceptual similarity is equivalent to the conceptual similarity of their attributes.

In situations $S2$ and $S3$, one node is a leaf and the other is an inner node. As the concept of a tree is dependent not only on its root's attribute but also on its children's attributes, the children of the inner node are also considered in the formula.

In situation $S4$, both $t_u[j]$ and $t_i[k]$ have children. Their children construct two forests $F_u[j]$ and $F_i[k]$. They are compared with the forest similarity measure $sc_F(F_u[j], F_i[k])$, which will be described in detail next.

Taking the two trees in Fig. 1 as examples, for node 4 in T_1 and node 5 in T_2, $sc_{T1}(T_1[4], T_2[5]) = sc$("*Mitchelton,*" "*Petaluma*"). For node 2 in T_1 and node 2 in T_2, $sc_{T_1}(T_1[2], T_2[2]) = \alpha \cdot sc$("*wine,*" "*wine*") $+ (1-\alpha)$ $sc_F(F_1[2], F_2[2])$.

In the case where $t_u[j]$ is matched to $t_i[k]$'s child, the concept level of $t_u[j]$ is lower than the concept level of $t_i[k]$. $T_u[j]$ is mapped to one subtree of $T_i[k]$ which has maximum conceptual similarity with $T_u[j]$. The conceptual similarity between $T_u[j]$ and $T_i[k]$ is represented as

$$sc_{T2}(T_u[j], T_i[k]) = \max_{1 \le t \le n_k} \{w_t \cdot sc_T(T_u[j], T_i[k_t])\} \quad (2)$$

where w_t is the weight of the matching node pair. If the matching is a symmetric matching, both of the corresponding nodes' weights should be considered, $w_t = (w(t_u[j]) + w(t_i[k_t]))/2$. If the measure is an asymmetric matching, then only the first node's weight is considered, $w_t = w(t_u[j])$.

For example, a business only supplies wine products. Its product tree is shown as T_3 in Fig. 3. When matching T_3 with T_1 in Fig. 1, the node 1 in T_3 is probably matched to the node 2 in T_1.

In the case where $t_i[k]$ is matched to $t_u[j]$'s child, the concept level of $t_i[k]$ is lower than the concept level of $t_u[j]$. $T_i[k]$ is mapped to one subtree of $T_u[j]$ which has maximum conceptual similarity with $T_i[k]$. The conceptual similarity between $T_u[j]$ and $T_i[k]$ is calculated as

$$sc_{T3}(T_u[j], T_i[k]) = \max_{1 \le t \le n_j} \{w_t \cdot sc_T(T_u[j_t], T_i[k])\} \quad (3)$$

where w_t is the weight of the matching node pair. If the matching is a symmetric matching, both of the corresponding nodes' weights should be considered, $w_t = (w(t_u[j_t]) + w(t_i[k]))/2$. If the measure is an asymmetric matching, then only the first node's weight is considered, $w_t = w(t_u[j_t])$.

Considering the three cases given previously, the case with the maximum conceptual similarity is selected, and the conceptual similarity calculated in that case is taken as the conceptual similarity between $T_u[j]$ and $T_i[k]$

$$sc_T(T_u[j], T_i[k]) = \max\{sc_{T_1}, sc_{T_2}, sc_{T_3}\} \quad (4)$$

In the fourth situation in Formula (1), both $t_u[j]$ and $t_i[k]$ have children which construct two forests, denoted as $F_u[j]$ and $F_i[k]$. To compute the conceptual similarity between $T_u[j]$ and $T_i[k]$, the conceptual similarity between $F_u[j]$ and $F_i[k]$ is required. In the following, the calculation method of $sc_F(F_u[j], F_i[k])$ is given.

The roots of $F_u[j]$ and $F_i[k]$ construct a bipartite graph, $G_{j,k} = (V_j \cup V_k, E)$, in which $V_j = \{t_u[j_1], t_u[j_2], ...t_u[j_{n_j}]\}, V_k = \{t_i[k_1], t_i[k_2], ...t_i[k_{n_k}]\}$, and $E = \{(s,t) : s \in V_j, t \in V_k\}$. For any edge (s,t), a weight is assigned to it as $w_{s,t} = sc_T(s,t)$. A maximum weighted bipartite mapping [47] of $G_{j,k}$, denoted as $\mathrm{MWBM}_{j,k}$, is constructed. The conceptual similarity between $F_u[j]$ and $F_i[k]$ is calculated as

$$sc_F(F_u[j], F_i[k]) = \sum_{(t_u[p], t_i[q]) \in \mathrm{MWBM}_{j,k}} w_{p,q} \cdot sc_T(T_u[p], T_i[q]) \quad (5)$$

where $w_{p,q}$ is the weight of the corresponding matching node pair $t_u[p]$ and $t_i[q]$. If the matching is a symmetric matching, $w_{p,q} = (w(t_u[p]) + w(t_i[q]))/2$. If the measure is an asymmetric measure, $w_{p,q} = w(t_u[p])$.

These maximum weighted bipartite mappings are recorded during the computation. The matching node pairs that maximize the conceptual similarity between two trees $T_u[j]$ and $T_i[k]$ are

$$sc_{T_1}(T_u[j], T_i[k]) = \begin{cases} sc(a(t_u[j]), a(t_i[k])), F_u[j] = \phi, F_i[k] = \phi, (S1) \\ \alpha \cdot sc(a(t_u[j]), a(t_i[k])) \\ +(1-\alpha) \cdot \sum_{t=1}^{n_k} w_{k_t} \cdot sc_T(T_u[j], T_i[k_t]), F_u[j] = \phi, F_i[k] \neq \phi, (S2) \\ \alpha \cdot sc(a(t_u[j]), a(t_i[k])) \\ +(1-\alpha) \cdot \sum_{t=1}^{n_j} w_{j_t} \cdot sc_T(T_u[j_t], T_i[k]), F_u[j] \neq \phi, F_i[k] = \phi, (S3) \\ \alpha \cdot sc(a(t_u[j]), a(t_i[k])) \\ +(1-\alpha) \cdot sc_F(F_u[j], F_i[k]), F_u[j] \neq \phi, F_i[k] \neq \phi, (S4) \end{cases} \quad (1)$$

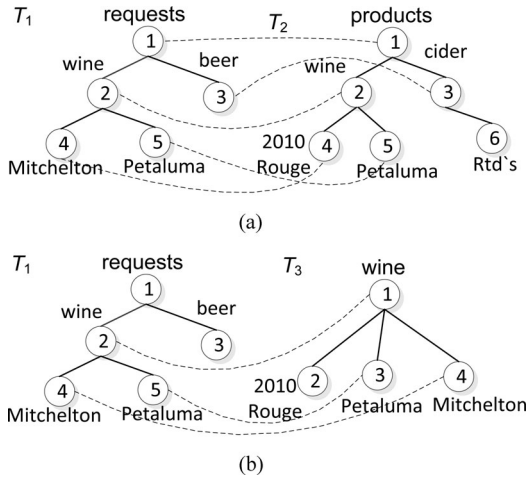


Fig. 4. Maximum conceptual similarity tree mappings between T_1 and T_2 (a) and between T_1 and T_3 (b).

finally taken as the maximum conceptual similarity tree mapping between $T_u[j]$ and $T_i[k]$.

Taking the three tree-structured data in Figs. 1 and 3 as examples, the maximum conceptual similarity mapping between T_1 and T_2, and that between T_1 and T_3 can be constructed. Given the conceptual similarity between attributes *sc*("*wine,*" "*beer*") = 0.6, *sc*("*beer,*" "*cider*") = 0.6, *sc*("*2010 Rouge,*" "*Petaluma*") = 0.8, *sc*("*2010 Rouge,*" "*Mitchelton*") = 0.8, let $\alpha = 0.5$, the conceptual similarity between T_1 and T_2, and that between T_1 and T_3 can be calculated as $sc_{T_{\text{asym}}}(T_1, T_2) = 0.88, sc_{T_{\text{asym}}}(T_1, T_3) = 0.5$. Their maximum conceptual similarity tree mappings are shown in Fig. 4, in which corresponding nodes are connected by dashes.

The computation process of the conceptual similarity between two trees is shown in Algorithm 1. The algorithm has three inputs: the two trees to be matched and the reference of a mapping set M which is used to record the maximum conceptual similarity tree mapping. The output of the algorithm is the conceptual similarity between the two trees.

In the algorithm, lines 1–18 deal with the case where the roots of two trees are matched. According to the condition of whether $t_u[j]$ and $t_i[k]$ are leaves, four situations are processed according to Formula (1). In the fourth situation, in line 15, *ComputeMatching*() [47] is used to calculate the maximum weighted bipartite matching. Lines 19-24 deal with the case where $t_u[j]$ is matched to a child of $t_i[k]$. Lines 25–30 deal with the case where $t_i[k]$ is matched to a child of $t_u[j]$. In lines 31–34, the maximum value of the three cases is taken as the final conceptual similarity value and the corresponding tree matching is added to the final maximum conceptual similarity tree mapping.

Based on the maximum conceptual similarity tree mapping, the nodes in two trees can be divided into three kinds: 1) conceptual corresponding nodes, which appear in the maximum conceptual similarity tree mapping; 2) semiconceptual corresponding nodes, which do not appear in the mapping but whose descendants appear in the mapping; 3) noncorresponding nodes, which neither appear in the mapping nor have descendants in the mapping. These three kinds of nodes play different roles in

Algorithm 1. Conceptual similarity computation algorithm $sc_T(T_u[j], T_i[k], M)$

input: two trees $T_u[j]$, $T_i[k]$ and the mapping set M

output: the conceptual similarity between $T_u[j]$ and $T_i[k]$

1 mapping set $M_1 \leftarrow \{(t_u[j], t_i[k])\}$

2 if $F_u[j] = \phi, F_i[k] = \phi$

3 $\quad sc_{T1} \leftarrow sc(a(t_u[j]), a(t_i[k]))$

4 else if $F_u[j] = \phi, F_i[k] \neq \phi$

5 $\quad sc_{T1} \leftarrow \alpha \cdot sc(a(t_u[j]), a(t_i[k])) + (1-\alpha) \cdot \sum_{t=1}^{n_k} w_{k_t} \cdot sc_T(T_u[j], T_i[k_t], \phi)$

6 else if $F_u[j] \neq \phi, F_i[k] = \phi$

7 $\quad sc_{T1} \leftarrow \alpha \cdot sc(a(t_u[j]), a(t_i[k])) + (1-\alpha) \cdot \sum_{t=1}^{n_j} w_{j_t} \cdot sc_T(T_u[j_t], T_i[k], \phi)$

8 else if $F_u[j] \neq \phi, F_i[k] \neq \phi$

9 $\quad V_j \leftarrow \{t_u[j_1], t_u[j_2], ... t_u[j_{n_j}]\}$

10 $\quad V_k \leftarrow \{t_i[k_1], t_i[k_2], ... t_i[k_{n_k}]\}$

11 for *s*=1 to n_j

12 $\quad$ for *t*=1 to n_k

13 $\quad\quad$ new mapping set $M_{s,t}$

14 $\quad\quad ew_{s,t} \leftarrow sc_T(T_u[j_s], T_i[k_t], M_{s,t})$

15 $m \leftarrow$ *ComputeMatching*($V_j \cup V_k$, *ew*)

16 for each $(t_u[j_s], t_i[j_t]) \in m$,

17 $\quad M_1 \leftarrow M_1 \cup M_{s,t}$

18 $\quad sc_{T1} \leftarrow \alpha \cdot sc(a(t_u[j]), a(t_i[k])) + (1-\alpha) \cdot \sum_{(t_u[j_s], t_i[j_t]) \in m} w_{s,t} \cdot ew_{s,t}$

19 $sc_{T2} \leftarrow 0$, mapping set $M_2 \leftarrow \phi$

20 for *t*=1 to n_k

21 $\quad$ new mapping set $M_{j,t}$

22 $\quad sc_t \leftarrow w_t \cdot sc_T(T_u[j], T_i[k_t], M_{j,t})$

23 $\quad$ if $sc_{T2} < sc_t$

24 $\quad\quad sc_{T2} \leftarrow sc_t$, $M_2 \leftarrow M_{j,t}$

25 $sc_{T3} \leftarrow 0$, mapping set $M_3 \leftarrow \phi$

26 for *t*=1 to n_j

27 $\quad$ new mapping set $M_{t,k}$

28 $\quad sc_t \leftarrow w_t \cdot sc_T(T_u[j_t], T_i[k], M_{t,k})$

29 $\quad$ if $sc_{T3} < sc_t$

30 $\quad\quad sc_{T3} \leftarrow sc_t$, $M_3 \leftarrow M_{t,k}$

31 for *p*=1,2,3

32 $\quad$ if $sc_{Tp} = \max\{sc_{T1}, sc_{T2}, sc_{T3}\}$

33 $\quad\quad M \leftarrow M \cup M_p$

34 return $\max\{sc_{T1}, sc_{T2}, sc_{T3}\}$

the mapping and are processed differently when dealing with the mapping.

V. Fuzzy Preference Tree Construction Algorithm

Since a user's preference is in a tree structure and has fuzzy values, a fuzzy preference tree is established to cover both the user's intentionally expressed preference and their extensionally expressed preference. As described in Section III, the intentionally expressed preference is acquired directly, while the extensionally expressed preference is constructed from the experience items of the user. The construction algorithm is presented in detail in this section.

A. Fuzzy Preference Tree Construction Algorithm

The construction process is an incremental process. The user's preferences for newly experienced items are integrated into the user's fuzzy preference tree.

The integration operation is described in detail as follows. It takes three components as input: the user's fuzzy preference tree T_u, the item tree T_i, and the user's preference value for the item $\tilde{p}_{ui}$. It contains two steps.

Step 1: Generate the maximum conceptual similarity tree mapping between T_u and T_i

A maximum conceptual similarity tree mapping between T_u and T_i, $M_{u,i}$, is constructed to identify the corresponding parts between two trees and to determine the positions in T_u into which the relevant nodes in T_i can be merged. The mapping should be symmetric, i.e., both trees' weights are treated equally. It is constructed by calculating $sc_{T_{\mathrm{sym}}}(T_u, T_i)$ using the proposed tree matching method.

Step 2: Merge T_i into T_u

Based on the maximum conceptual similarity tree mapping between T_u and T_i, $M_{u,i}$, all the features in T_i are merged into T_u. A merge operation is defined which takes the tree mapping $M_{u,i}$, the item tree node n_i, and the user's preference value for the item $\tilde{p}_{ui}$ as input.

According to the different mapping situations of n_i, the merge operation is processed in the following five cases.

In Case 1, $M_{u,i}$ is empty. This case emerges when T_u is initially empty or the subtree under n_i and T_u represent totally different features. In this case, a new root of T_u is created, and the original T_u is inserted as a subtree. The subtree under n_i is copied and inserted under the new root of T_u. Each leaf of the copied subtree is assigned a value whose preference is $\tilde{p}_{ui}$ and count is 1.

In Case 2, n_i is mapped to a node n_p in the mapping $M_{u,i}$, but the attributes of n_i and n_u are not identical. In this case, the subtree under n_i is copied and inserted under the parent node of n_p. Each leaf of the copied subtree is assigned a value whose preference is $\tilde{p}_{ui}$ and count is 1.

In Case 3, n_i is mapped to a node n_p in the mapping $M_{u,i}$, and their attributes are identical. According to the condition of whether or not n_i has children, the operation is processed in the following two cases. In the first case, n_i has no children, i.e., n_i represents the finest feature. The $\tilde{p}_{ui}$ is integrated into the preference value of node n_p which is denoted as $\tilde{p}_{un_p}$: $f_{k,un_p} = \left(f_{k,un_p} \cdot \mathrm{count} + f_{k,ui}\right) / (\mathrm{count}+1), k = 1, 2, \ldots, r$; $\mathrm{count} = \mathrm{count} + 1$. In the second case, n_i has child nodes. These child nodes are merged recursively.

In Case 4, n_i is not in the mapping $M_{u,i}$, but its parent node is mapped to node n_p. The subtree under n_i is copied and inserted under n_p. Each leaf of the copied subtree is assigned a value whose preference is $\tilde{p}_{ui}$ and count is 1.

In Case 5, neither n_i nor its ancestor nodes are in the mapping $M_{u,i}$, but n_i's descendant nodes are in the mapping. In this case, T_u represents part of the features of the subtree under n_i. The root of T_u must be mapped to a node n_t which is the descendant of n_i. The tree under n_i except for the subtree under n_t is copied and taken as tree T'_u. Each leaf of the copied subtree is assigned a value whose preference is $\tilde{p}_{ui}$ and count is 1. Let n_t's parent node be n_p. T_u is inserted into T'_u under the corresponding node of n_p. Replace T_u with T'_u, and then merge the node n_t recursively.

The process of the merge operation is shown in Algorithm 2, which takes the reference of the fuzzy preference tree as input, and obtains the merged fuzzy preference tree following the operation.

In Algorithm 2, lines 1–7, lines 11–14, lines 16–21, lines 24–27, and lines 29–35 deal with the five cases, respectively. The procedure *insert*(r, T_{n_i}) inserts T_{n_i} under the tree node r. *CopyTree*(n_i) copies the subtree under n_i. $SetLeafValues(T_{n_i}, \tilde{p}_{ui})$ sets the preference values of the leaves of T_{n_i} as $\tilde{p}_{ui}$. *GetMappedNode*$(n_i, M_{u,i})$ returns the mapped node of n_i in the mapping $M_{u,i}$. *SearchMappedDescendant*$(n_i, M_{u,i})$ returns the first node under n_i which is in the mapping $M_{u,i}$. *remove*(n_i, n_t) removes the subtree under n_t from the tree under n_i.

The merging operation is a recursive process. To merge T_i into T_u, the root of T_i is merged. Following the merging operation, the weights of the updated fuzzy preference tree T_u are normalized.

B. Example

An example to show the construction of a business user's preference tree is given as follows.

Let T_1 in Fig. 1 represent the structure of a user's intentionally expressed preference. The preference values of nodes 3, 4, 5 are: $\tilde{p}(t_1[3])$ = (0/1, 0/2, 0/3, 1/4, 0.8/5), $\tilde{p}(t_1[4])$ = (0/1, 0/2, 0/3, 0.8/4, 1/5), $\tilde{p}(t_1[5])$ = (0/1, 0/2, 0/3, 0.8/4, 1/5). Let T_2 in Fig. 1 be the product tree of a business experienced by the user. The preference of the user for T_2 is $\tilde{p}_i$ = (0/1, 0/2, 0.6/3, 1/4, 0.6/5). This information can be merged to construct the user's fuzzy preference tree, which is shown in Fig. 5.

In T_u, the preference values of the nodes are: $\tilde{p}(t_1[3])$ = (0/1, 0/2, 0/3, 1/4, 0.8/5), $\tilde{p}(t_1[4])$ = (0/1, 0/2, 0/3, 0.8/4, 1/5), $\tilde{p}(t_1[5])$ = (0/1, 0/2, 0.3/3, 0.9/4, 0.8/5), $\tilde{p}(t_1[6])$ = (0/1, 0/2, 0.6/3, 1/4, 0.6/5), $\tilde{p}(t_1[8])$ = (0/1, 0/2, 0.6/3, 1/4, 0.6/5).

VI. Fuzzy Preference Tree-Based Recommendation Approach

A fuzzy preference tree-based recommendation approach for tree-structured items is proposed in this section. The proposed

Algorithm 2. Fuzzy preference tree merging algorithm *merge*($T_u, n_i, \widetilde{p}_{ui}, M_{u,i}$)

input: fuzzy preference tree T_u, item tree node n_i, the user's preference value to the item $\widetilde{p}_{ui}$, the maximum conceptual similarity tree mapping between T_u and item tree $M_{u,i}$

```
1  if M_{u,i} = φ
2     create a tree node r
3     insert(r, T_u)
4     tree T_{n_i} ← CopyTree(n_i)
5     SetLeafValues(T_{n_i}, p̃_{ui})
6     insert(r, T_{n_i})
7     T_u ← r
8  else
9     n_p ← GetMappedNode(n_i, M_{u,i})
10    if n_p ≠ null
11       if a(n_p) ≠ a(n_i)
12          tree T_{n_i} ← CopyTree(n_i)
13          SetLeafValues(T_{n_i}, p̃_{ui})
14          insert(parent(n_p), T_{n_i})
15       else
16          if n_i has no children
            let n_p.p̃_u be (f_{1,un_p}, f_{2,un_p}, ..., f_{r,un_p}), n_p.count
            be c, p̃_{ui} be (f_{1,ui}, f_{2,ui}, ..., f_{r,ui})
17             f_{k,un_p} ← (f_{k,un_p} · c + f_{k,ui})/(c+1), k=1,2,..,r
18             n_p.count ← c+1
19          else
20             for each child node n_{ic} of n_i
21                merge(T_u, n_{ic}, p̃_{ui}, M_{u,i})
22    else
23       n_p ← GetMappedNode(parent(n_i), M_{u,i})
24       if n_p ≠ null
25          tree T_{n_i} ← CopyTree(n_i)
26          SetLeafValues(T_{n_i}, p̃_{ui})
27          insert(n_p, T_{n_i})
28       else
29          n_t ← SearchMappedDescendant(n_i, M_{u,i})
30          n_p ← parent(n_t)
31          remove(n_i, n_t)
32          tree T_{n_i} ← CopyTree(n_i)
33          insert(n_p, T_u)
34          T_u ← T_{n_i}
35          merge(T_u, n_t, p̃_{ui}, M_{u,i})
```

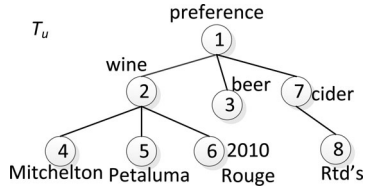


Fig. 5. Constructed fuzzy preference tree.

approach takes a user's fuzzy preference tree T_u and an item tree T_i as input, and calculates the predicted rating of the user to the target item.

The recommendation approach contains two steps. In the first step, the corresponding parts of T_u and T_i are matched. In the second step, a predicted rating of the user to the target item is calculated by aggregating the user preferences on the matched part of T_u.

A. Step 1: Identifying the Corresponding Parts of T_u and T_i

A maximum conceptual similarity tree mapping between T_u and T_i, $M_{u,i}$, is constructed to identify the corresponding parts between two trees. The mapping should mainly consider the weights of T_u, i.e., it is an asymmetric mapping. The mapping is constructed by calculating $sc_{T_{\text{asym}}}(T_u, T_i)$ using the proposed tree matching method.

B. Step 2: A Fuzzy Preference Tree-Based Recommendation Approach

Given a user's fuzzy preference tree T_u and an item tree T_i, the maximum conceptual similarity tree mapping between T_u and T_i, $M_{u,i}$ has been constructed. A function $pr()$, which takes the fuzzy preference tree node and the maximum conceptual similarity tree mapping as input, is developed to calculate the predicted rating of user u to item i.

Let $v(t_u[j])$ represent the value of node $t_u[j]$. $v(t_u[j]) = null$, if $t_u[j]$ is not assigned a value. Let $mc(t_u[j])$ represent the child nodes of $t_u[j]$ that are in the maximum conceptual similarity tree mapping. According to whether $v(t_u[j])$ and $mc(t_u[j])$ in different cases are $null$ or not, $pr(t_u[j], M_{u,i})$ is calculated in the following four cases.

Case 1: $v(t_u[j]) = null, mc(t_u[j]) = null$.

In this case, the subtree $T_u[j]$ makes no contribution to the predicted rating

$$pr(t_u[j], M_{u,i}) = 0. \tag{6}$$

Case 2: $v(t_u[j]) \neq null, mc(t_u[j]) = null$.

In this case, node $t_u[j]$ is assigned a preference value. Let it be $\tilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \ldots, f_{r,uj}\}$

$$pr(t_u[j], M_{u,i}) = \sum_{k=1}^{r} k \cdot f_{k,uj}. \tag{7}$$

Case 3: $v(t_u[j]) = null, mc(t_u[j]) \neq null$.

In this case, the predicted rating value of $t_u[j]$ is calculated by aggregating the predicted ratings of its mapped children

$$pr(t_u[j], M_{u,i}) = \sum_{t_u[j_x] \in mc(t_u[j])} w_x \cdot pr(t_u[j_x], M_{u,i}) \tag{8}$$

Algorithm 3. Rating prediction algorithm $pr(t_u[j], M_{u,i})$

input: fuzzy preference tree node, the maximum conceptual similarity tree mapping
output: the predicted rating

1 $mc \leftarrow MatchedChildren(t_u[j], M_{u,i})$
2 if $v(t_u[j]) = null$ and $mc = null$
3 return 0;
4 else if $v(t_u[j]) \neq null$ and $mc = null$
5 let the preference value be $\widetilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \ldots, f_{r,uj}\}$
return $\sum_{k=1}^{r} k \cdot f_{k,uj}$
6 else if $v(t_u[j]) = null$ and $mc(t_u[j]) \neq null$
7 return $\sum_{t_u[j_x] \in mc} w_x \cdot pr(t_u[j_x], M_{u,i})$
8 else if $v(t_u[j]) \neq null$ and $mc(t_u[j]) \neq null$
9 return $\beta_j \cdot \sum_{k=1}^{r} k \cdot f_{k,uj} + (1-\beta_j) \cdot \sum_{t_u[j_x] \in mc} w_x \cdot pr(t_u[j_x], M_{u,i})$

where w_x represents the aggregating weight of the node $t_u[j_x]$, $w_x = w(t_u[j_x]) / \sum_{t_u[j_s] \in mc(t_u[j])} w(t_u[j_s])$.

Case 4: $v(t_u[j]) \neq null, mc(t_u[j]) \neq null$.

This case is a combination of Cases 2 and 3. Both the values of node $t_u[j]$ and its children should be considered

$$pr(t_u[j], M_{u,i}) = \beta_j \cdot \sum_{k=1}^{r} k \cdot f_{k,uj} + (1-\beta_j) \times \sum_{t_u[j_x] \in mc(t_u[j])} w_x \cdot pr(t_u[j_x], M_{u,i}) \quad (9)$$

where $w_x = w(t_u[j_x]) / \sum_{t_u[j_s] \in mc(t_u[j])} w(t_u[j_s])$, and β_j is the influence factor of the parent node $t_u[j]$.

The calculation process of the predicted rating is shown in Algorithm 3.

Let the root node of T_u be $\mathrm{root}(T_u)$. The predicted rating of user u to item i is calculated as $pr(\mathrm{root}(T_u), M_{u,i})$.

The proposed recommendation approach overcomes the CS issue more efficiently than existing approaches because a new user's preferences take the form of a tree structure and are intentionally expressed with uncertain values. Our approach handles this issue using fuzzy preference tree and is therefore able to make more accurate recommendations to new users. Existing methods such as CB and CF cannot do this. Similarly, the proposed approach can also recommend new items more accurately by constructing new item trees, whereas the CF approach cannot. The proposed approach also overcomes the sparsity issue more effectively than existing methods, because it does not rely on the user-item rating matrix to calculate the user or item similarity. It therefore does not suffer from the sparsity problem which commonly exists in CF methods. Moreover, the proposed approach better overcomes the scalability issue than existing methods because the incremental construction of the fuzzy preference tree means that a user's fuzzy preference tree can be updated efficiently, which deals with the scalability problem to some extent.

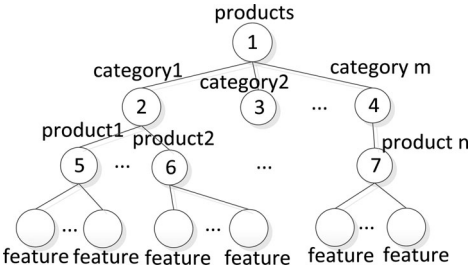


Fig. 6. Business product tree structure.

VII. Experimental Evaluation

This section evaluates the performance of the proposed fuzzy preference tree-based recommendation approach. It is evaluated by comparing it with existing recommendation approaches. The section includes the datasets used for evaluation, the evaluation metrics, and the evaluation results.

A. Evaluation DataSets

Two datasets are used to validate the performance of the proposed recommendation approach.

1) Australian Business DataSet: This dataset was collected from Australian businesses. It contains the ratings of business users on their business partners (buyers or suppliers). Basic business information, product information, and buying request information was collected for each business. Basic information includes a company's business type, scale, contact information, and so on. Product information contains the product categories and detailed features of the products supplied by the business, and there is a product category tree in the dataset which infers the semantic relations between products or product categories. The buying request information consists of the products or product categories required by the business. The products and buying requests of a business are both presented as hierarchical structures and described as tree-structured data. The structure of the product tree of a business is illustrated in Fig. 6. The buying requests of businesses have similar structures. There are 130 business users in the dataset, which includes 363 ratings on suppliers and 360 ratings on buyers. In the experiment, 20% of ratings from both types are randomly selected as the testing set. Because making buyer and supplier recommendations requires different information, the applications to recommend buyers and suppliers are described separately. When recommending suppliers, the potential suppliers' product trees are taken as the item trees, and the user's buying request is taken as the user's intentionally expressed preference. The fuzzy tree-structured user preference profile is constructed by merging the user's buying request and the product trees of businesses rated by the user in the training set. When recommending buyers, the potential buyers' request

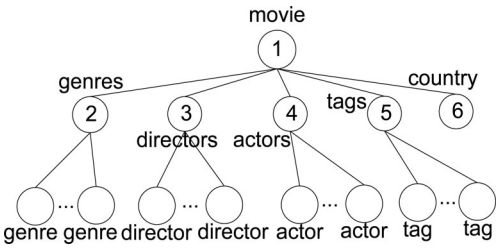


Fig. 7. Movie tree structure.

trees are taken as the item trees, and the user's product tree is taken as the user's intentionally expressed preference. The user's preference profile is constructed by merging the user's product tree and the buying requirement trees of businesses rated by the user in the training set.

2) The HetRec 2011 MovieLens Dataset (HetRec 2011, http://ir.ii.uam.es/hetrec2011): The HetRec 2011 MovieLens Dataset is extended from the MovieLens10M dataset (http://www.grouplens.org/node/73). The dataset includes the user ratings, tags, movie genres, directors, actors, and so on. Because the MovieLens dataset is commonly used in recommender system research and has been used to test many other recommendation approaches, this dataset is used to fairly compare the performance of our approach with other approaches. In our experiment, each movie is represented as a tree-structured object. The structure is shown in Fig. 7. There are 2113 users in the dataset. In our experiment, the ratings of each user are split into two parts, the training set and the testing set, and the ten most recent ratings of each user make up the testing set. The movies in a user's training set are used to construct that user's preference profile tree. The ratings of users on movies in the dataset are on a scale of 1 to 5. In our experiments, each rating is transformed into a fuzzy subset on $\{1, 2, 3, 4, 5\}$.

B. Evaluation Metrics

The following evaluation metrics have been used in this study.

1) Statistical Accuracy Metric: This measure evaluates the accuracy of a recommender system by comparing the numerical prediction values with the user ratings for the user–item pair in the test set. The mean absolute error (MAE) is the most widely used statistical accuracy metric in recommendation research [48]. MAE is computed as the average absolute difference between predictions and actual ratings. In particular, given the set of the actual/predicted rating pair (r_a, r_p) for all the n items in the test set, the MAE is computed by

$$\text{MAE} = \frac{\sum_{i=1}^{n} |r_a - r_p|}{n}. \tag{10}$$

2) Recall, Precision, and F1 Metrics: Recall is defined as the fraction of preferred items that are recommended. Precision is defined as the fraction of recommended items preferred by the user. The F1-measure, which combines precision and recall, is the harmonic mean of precision and recall.

In this experiment, a preferred rating threshold is predefined. The preferred movies are the movies in the test set whose actual ratings are greater than the preferred rating threshold. The recommended movies are the movies whose predicted ratings are greater than the preferred rating threshold. The recall, precision, and F1 are defined as follows:

$$\text{recall} = \frac{|\{\text{preferred}\} \cap \{\text{recommended}\}|}{|\{\text{preferred}\}|} \tag{11}$$

$$\text{precision} = \frac{|\{\text{preferred}\} \cap \{\text{recommended}\}|}{|\{\text{recommended}\}|} \tag{12}$$

$$F1 = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}. \tag{13}$$

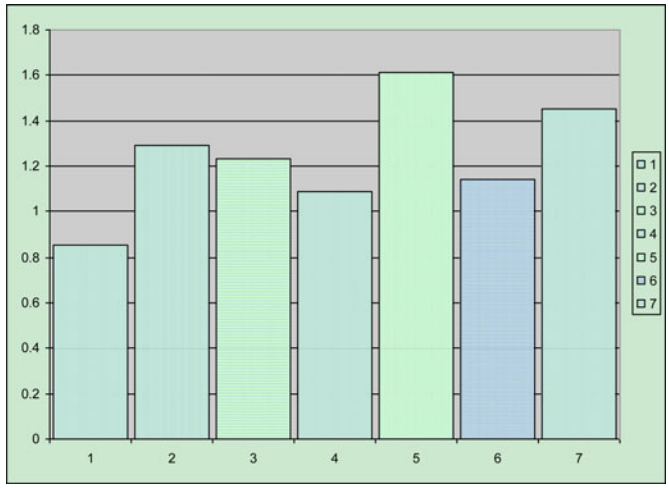


Fig. 8. MAE on the Australian business dataset.

C. Benchmark Recommendation Approaches

The application of fuzzy modeling for a CB recommender system was initially presented in [10]. A fuzzy set theoretic method (FTM) for recommender systems was proposed in [8]. Four kinds of fuzzy set-based similarity measures were introduced: fuzzy set theoretic, cosine, proximity, and correlation-like. The FTM with all four similarity measures was implemented in our experiment. A crisp set-based method was also implemented. A fuzzy user preference model was presented in [9] and was also implemented in our experiment for comparison.

D. Evaluation Results

Figs. 8–15 show the MAE, precision, recall, and F1 of each recommendation approach. In these figures, the first approach is our proposed fuzzy preference tree-based recommendation approach. Approaches two to five are the FTM methods with fuzzy set theoretic, cosine, proximity, and correlation-like similarity measures, respectively. The sixth approach is the crisp set-based method. The seventh approach is the fuzzy user preference model-based method [9].

1) Evaluation Results on the Australian Business Dataset: Figs. 8–11 show the evaluation results on the Australian business dataset. It can be seen that the proposed recommendation approach in this study has the lowest MAE, the highest precision, high recall, and highest F1 measure. The results indicate that the fuzzy tree-structured user preference profile effectively

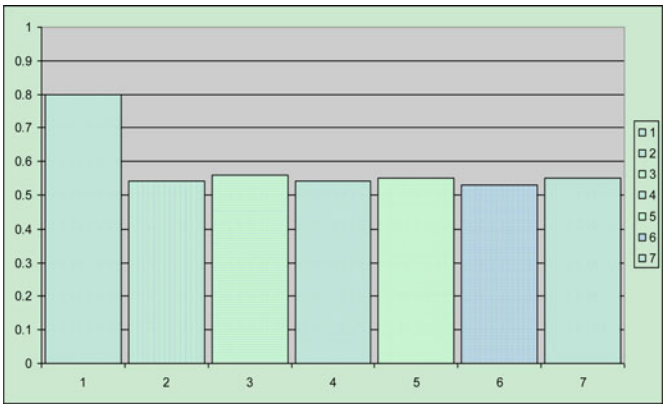


Fig. 9. Precision on the Australian business dataset.

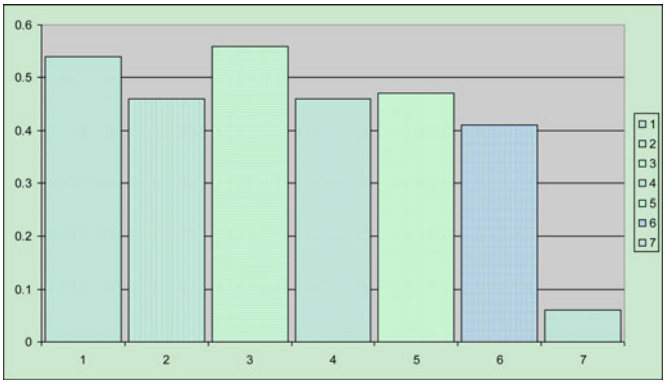


Fig. 10. Recall on the Australian business dataset.

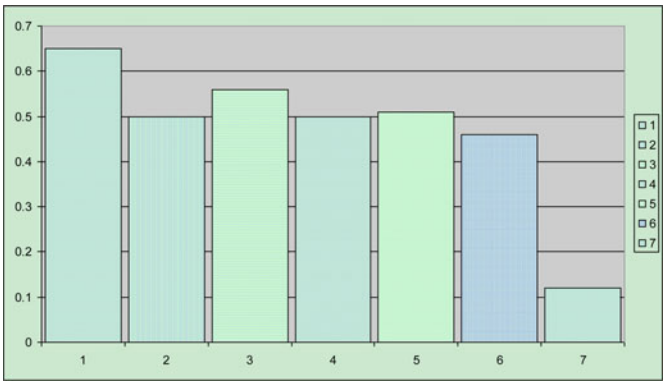


Fig. 11. F1 on the Australian business dataset.

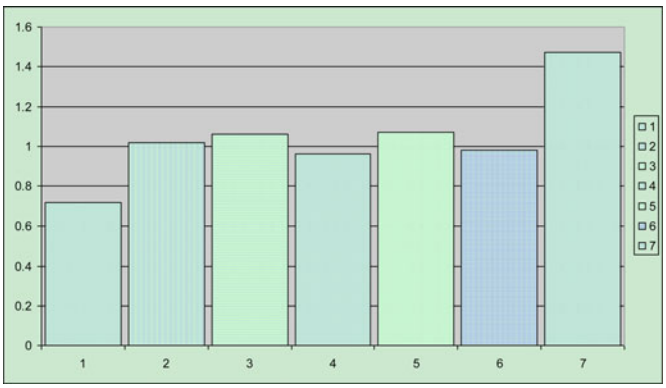


Fig. 12. MAE on the MovieLens dataset.

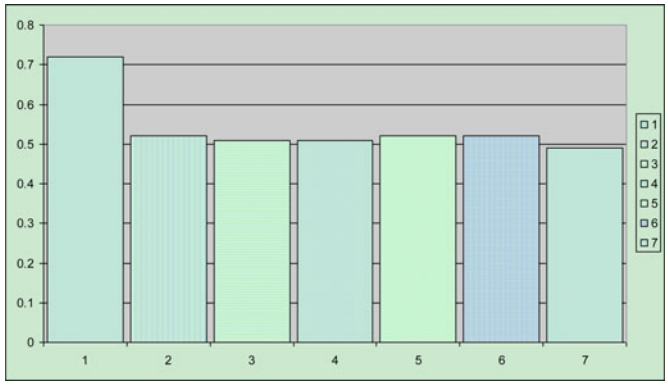


Fig. 13. Precision on the MovieLens dataset.

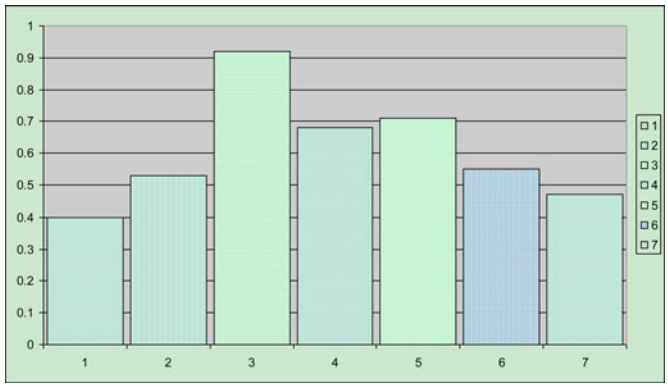


Fig. 14. Recall on the MovieLens dataset.

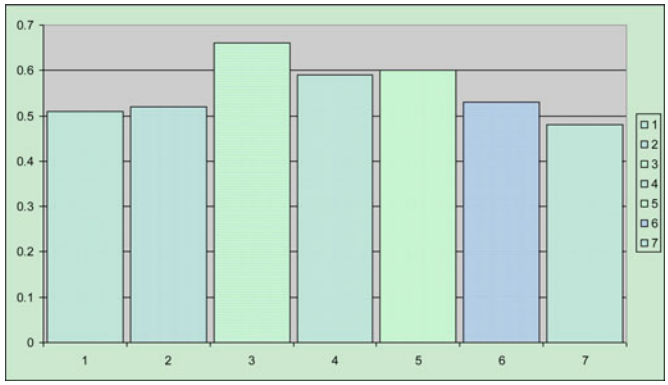


Fig. 15. F1 on the MovieLens dataset.

reflects business users' preferences, and the proposed approach is well-suited to the business application environment.

2) Evaluation Results on the MovieLens Dataset: Figs. 12–15 show the evaluation results on the MovieLens dataset. It can be seen that the proposed approach in this study has the lowest MAE and the highest precision, which indicates the accuracy of our approach. Even though the recall of the proposed approach is a little lower, the F1 measure is comparable with others.

Experimental results on both datasets show that high recommendation accuracy is obtained by representing the user preferences with our proposed fuzzy tree-structured preference model, especially on the Australian business dataset which has tree-structured features. This reflects the effectiveness of the

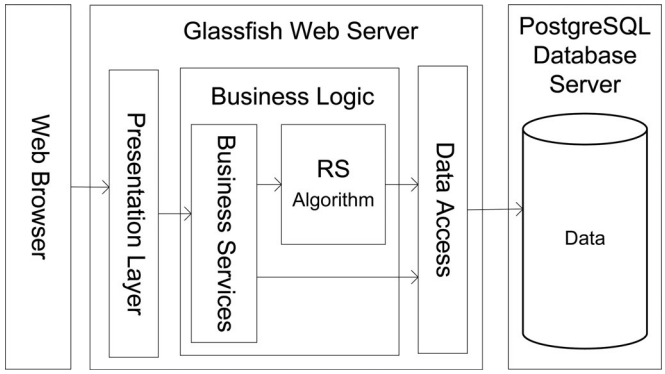


Fig. 16. System architecture of Smart BizSeeker.

fuzzy tree-structured user preference model and the proposed recommendation approach based on it.

VIII. Smart BizSeeker—A Web-Based Recommender System for Business Partner Recommendation

The proposed approach has been implemented in a recommender system software called Smart BizSeeker. The Smart BizSeeker is designed to recommend business partners (buyers and suppliers) to business users.

A. System Architecture

The architecture of Smart BizSeeker is illustrated in Fig. 16. The arrows in the figure represent the function call relations. As a Web-based online system, Smart BizSeeker has a standard multitier architecture, which includes web browser, web server, and database server. The main components of the system are as follows. A database which stores all the business data in the system is designed and implemented in the PostgreSQL database server. The application in the web server contains three layers: the presentation layer, business logic layer, and data access layer. The presentation layer is responsible to generate the requested web pages and handling the user interface logic and events. The business logic layer realizes the business services and the core recommendation algorithm. The proposed recommendation approach is applied here for both the buyer and supplier recommendations. The data access layer deals with the data operations of the database.

B. System Implementation

The system is developed using the Netbeans development platform. JSF, EJB, and JPA frameworks are used in the implementation of the presentation layer, business logic layer, and data access layer.

Fig. 17 shows the login page of the Smart BizSeeker software. Two examples of supplier recommendation results and buyer recommendation results are illustrated in Figs. 18 and 19, respectively.

Fig. 17. Login page of Smart BizSeeker.

Fig. 18. Supplier recommendation results.

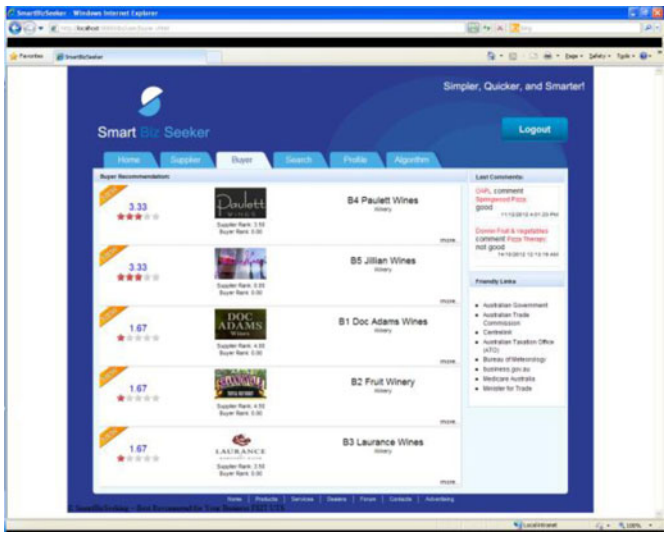


Fig. 19. Buyer recommendation results.

IX. Conclusion and Further Studies

This paper proposes a method for modeling tree-structured user preferences with uncertainty and develops a new recommendation approach, which can recommend tree-structured items. The fuzzy tree-structured user preference modeling method integrates both the user's extensionally and intentionally expressed preferences. During the construction process of the user's fuzzy preference tree and the matching process between the fuzzy preference tree and item trees, a comprehensive tree matching method for identifying the corresponding parts of two tree-structured data is presented, which comprehensively considers tree structures, node attributes, and node weights. Two experiments on an Australian business dataset and the MovieLens dataset, respectively, are conducted to evaluate the performance of the proposed recommendation approach. Both results show that our approach makes accurate recommendations and demonstrates that the fuzzy tree-structured user preference profile reflects user preferences effectively. The experiment on the Australian businessdata set shows that it is well-suited to the business application environment. The proposed recommendation approach is implemented in a business partner recommender system software.

At the current research stage, the inputs of the recommendation approach require tree-structured data and cannot deal with data in a network structure or a matrix. In addition, this approach requires that the tree-structured data must satisfy the requirement that the semantic meanings of the parent–child relations in different tree-structured data must be the same. However, this approach provides a new solution for improving recommender systems in general and it can therefore be used in e-Government, e-Business, e-learning, and so on when the data are described in a tree structure.

In the future, we will consider the features and characteristics of groups of similar businesses and will develop methods for identifying business groups and make group recommendations.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[2] X. Guo and J. Lu, "Intelligent e-government services with personalized recommendation techniques," in *Proc. Int. J. Intell. Syst.*, 2007, vol. 22, pp. 401–417.

[3] P. Markellou, I. Mousourouli, S. Sirmakessis, and A. Tsakalidis, "Personalized e-commerce recommendations," in *Proc. IEEE Int. Conf. e-Bus. Eng.*, Beijing, China, 2005, pp. 245–252.

[4] J. Lu, "Personalized e-learning material recommender system," in *Proc. 2nd Int. Conf. Inf. Technol. Appl.*, 2004, pp. 374–379.

[5] J. Gemmell, T. Schimoler, B. Mobasher, and R. Burke, "Resource recommendation in social annotation systems: A linear-weighted hybrid approach," *J. Comput. Syst. Sci.*, vol. 78, pp. 1160–1174, 2012.

[6] X. Yuan, J. H. Lee, S. J. Kim, and Y. H. Kim, "Toward a user-oriented recommendation system for real estate websites," *Inf. Syst.*, vol. 38, pp. 231–243, 2013.

[7] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling User-Adapted Interact.*, vol. 12, pp. 331–370, 2002.

[8] A. Zenebe and A. F. Norcio, "Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems," *Fuzzy Sets Syst.*, vol. 160, pp. 76–94, 2009.

[9] A. Zenebe, L. Zhou, and A. F. Norcio, "User preferences discovery using fuzzy models," *Fuzzy Sets Syst.*, vol. 161, pp. 3044–3063, 2010.

[10] R. R. Yager, "Fuzzy logic methods in recommender systems," *Fuzzy Sets Syst.*, vol. 136, pp. 133–149, 2003.

[11] C. M. Chen and L. J. Duh, "Personalized web-based tutoring system based on fuzzy item response theory," *Expert Syst. Appl.*, vol. 34, pp. 2298–2315, 2008.

[12] C. Cornelis, J. Lu, X. Guo, and G. Zhang, "One-and-only item recommendation with fuzzy logic techniques," *Inf. Sci.*, vol. 177, pp. 4906–4921, 2007.

[13] C. Porcel, A. G. López-Herrera, and E. Herrera-Viedma, "A recommender system for research resources based on fuzzy linguistic modeling," *Expert Syst. Appl.*, vol. 36, pp. 5173–5183, 2009.

[14] J. Lu, Q. Shambour, Y. Xu, Q. Lin, and G. Zhang, "BizSeeker: A hybrid semantic recommendation system for personalized government-to-business e-services," *Internet Res.*, vol. 20, pp. 342–365, 2010.

[15] Z. Huang, D. Zeng, and H. Chen, "A comparison of collaborative-filtering recommendation algorithms for e-commerce," *IEEE Intell. Syst.*, vol. 22, no. 5, pp. 68–78, Sep./Oct. 2007.

[16] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*, vol. 4321, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 291–324.

[17] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, pp. 143–177, 2004.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[19] R. M. Rodríguez, M. Espinilla, P. J. Sánchez, and L. Martínez-López, "Using linguistic incomplete preference relations to cold start recommendations," *Internet Res.*, vol. 20, pp. 296–315, 2010.

[20] M. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds.. vol. 4321, Berlin, Germany: Springer-Verlag, 2007, pp. 325–341.

[21] A. Felfernig, B. Gula, G. Leitner, M. Maier, R. Melcher, and E. Teppan, "Persuasion in knowledge-based recommendation," in *Persuasive Technology, vol. 5033*, H. Oinas-Kukkonen, P. Hasle, M. Harjumaa, K. Segerståhl, and P. Øhrstrøm, Eds. Berlin, Germany: Springer-Verlag, 2008, pp. 71–82.

[22] R. Burke, "Hybrid Web recommender systems," in *The Adaptive Web, vol. 4321*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Germany: Springer-Verlag, 2007, pp. 377–408.

[23] B. Kim, Q. Li, C. Park, S. Kim, and J. Kim, "A new approach for combining content-based and collaborative filters," *J. Intell. Inf. Syst.*, vol. 27, pp. 79–91, 2006.

[24] M. Ruiz-Montiel and J. Aldana-Montes, "Semantically enhanced recommender systems," in *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, R. Meersman, P. Herrero, and T. Dillon, Eds. Berlin, Germany: Springer-Verlag, 2009, vol. 5872, pp. 604–609.

[25] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proc. 14th Int. Joint Conf. Artif. Intell.*, Montreal, QC, Canada, 1995, pp. 448–453.

[26] A. Albadvi and M. Shahbazi, "A hybrid recommendation technique based on product category attributes," *Expert Syst. Appl.*, vol. 36, pp. 11480–11488, 2009.

[27] Y. H. Cho and J. K. Kim, "Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce," *Expert Syst. Appl.*, vol. 26, pp. 233–246, 2004.

[28] L. P. Hung, "A personalized recommendation system based on product taxonomy for one-to-one marketing online," *Expert Syst. Appl.*, vol. 29, pp. 383–392, 2005.

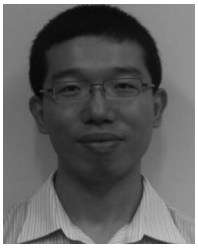
[29] Q. Shambour and J. Lu, "A trust-semantic fusion-based recommendation approach for e-business applications," *Decision Support Syst.*, vol. 54, pp. 768–780, 2012.

[30] A. Ouangraoua and P. Ferraro, "A constrained edit distance algorithm between semi-ordered trees," *Theor. Comput. Sci.*, vol. 410, pp. 837–846, 2009.

[31] Y. Xue, C. Wang, H. Ghenniwa, and W. Shen, "A tree similarity measuring method and its application to ontology comparison," *J. Universal Comput. Sci.*, vol. 15, pp. 1766–1781, 2009.

[32] Z. Lin, H. Wang, and S. McClean, "A multidimensional sequence approach to measuring tree similarity," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 2, pp. 197–208, Feb. 2012.

[33] V. C. Bhavsar, H. Boley, and L. Yang, "A weighted-tree similarity algorithm for multi-agent systems in e-business environments," *Comput. Intell.*, vol. 20, pp. 1–20, 2004.
[34] L. Yang, B. Sarker, V. Bhavsar, and H. Boley, "A weighted-tree simplicity algorithm for similarity matching of partial product descriptions," in *Proc. Int. Soc. Comput. Appl. 14th Int. Conf. Intell. Adaptive Syst. Softw. Eng.*, Toronto, ON, Canada, 2005, pp. 55–60.
[35] R. Connor, F. Simeoni, M. Iakovos, and R. Moss, "A bounded distance metric for comparing tree structure," *Inf. Syst.*, vol. 36, pp. 748–764, 2011.
[36] F. Ricci and L. Senter, "Structured cases, trees and efficient retrieval," *Adv. Case-Based Reason.*, vol. 1488, pp. 88–99, 1998.
[37] P. Bille, "A survey on tree edit distance and related problems," *Theor. Comput. Sci.*, vol. 337, pp. 217–239, 2005.
[38] K. Zhang, "A new editing based distance between unordered labeled trees," in *Combinatorial Pattern Matching*, A. Apostolico, M. Crochemore, Z. Galil, and U. Manber, Eds. Berlin, Germany: Springer-Verlag, 1993, pp. 254–265.
[39] D. Wu, J. Lu, and G. Zhang, "Similarity measure models and algorithms for hierarchical cases," *Expert Syst. Appl.*, vol. 38, pp. 15049–15056, 2011.
[40] D. Wu, J. Lu, and G. Zhang, "A hybrid recommendation approach for hierarchical items," in *Proc. Int. Conf. Intell. Syst. Knowl. Eng.*, 2010, pp. 492–497.
[41] N. Biadsy, L. Rokach, and A. Shmilovici, "Transfer learning for content-based recommender systems using tree matching," in *Availability, Reliability, and Security in Information Systems and HCI*, A. Cuzzocrea, C. Kittl, D. Simos, E. Weippl, and L. Xu, Eds. Berlin, Germany: Springer, 2013, pp. 387–399.
[42] D. Wu, G. Zhang, and J. Lu, "A fuzzy tree similarity based recommendation approach for telecom products," in *Proc. Joint IFSA World Congress NAFIPS Annu. Meeting (IFSA/)NAFIPS*, Edmonton, Canada, 2013, pp. 813–818.
[43] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.
[44] Y. Cao and Y. Li, "An intelligent fuzzy-based recommendation system for consumer electronic products," *Expert Syst. Appl.*, vol. 33, pp. 230–240, 2007.
[45] G. Valiente, *Algorithms on Trees and Graphs*. New York, NY, USA: Springer-Verlag, 2002.
[46] K. Zhang, "A constrained edit distance between unordered labeled trees," *Algorithmica*, vol. 15, pp. 205–222, 1996.
[47] D. Jungnickel, *Graphs, Networks, and Algorithms*, 3rd ed. Berlin, Germany: Springer-Verlag, 2008.
[48] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: An open architecture for collaborative filtering of netnews," in *Proc. ACM Conf. Comput. Supported Cooperative Work*, Chapel Hill, North Carolina, USA, 1994, pp. 175–186.



Dianshuang Wu is working toward the Ph.D. degree with the Faculty of Engineering and Information Technology, School of Software, University of Technology Sydney, Ultimo, NSW, Australia.

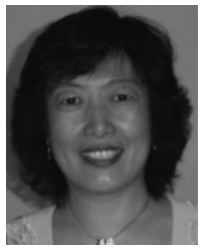
His research interests include tree similarity measure, recommender systems, and business intelligence.

He is a Member of the Decision Systems and e-Service Intelligence Research Lab, Centre for Quantum Computation & Intelligent Systems, University of Technology Sydney.

Guangquan Zhang received the Ph.D. degree in applied mathematics from Curtin University of Technology, Perth, Australia, in 2001.

He is an Associate Professor and Co-Director of the Decision Systems and e-Service Intelligence Lab, Centre for Quantum Computation and Intelligent Systems, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia. He has authored and coauthored 324 publications, including four monographs, five textbooks, 17 book chapters, and 154 refereed international journal papers.

Dr. Zhang has served as an Advisory Board Member or a Member of the editorial boards of several international journals and co-chaired several international conferences/workshops in the area of fuzzy decision-making and knowledge engineering, such as FLINS08, FLINS10, and ISKE09.



Jie Lu (SM'13) received the Ph.D. degree from the Curtin University of Technology, Bentley, WA, Australia, in 2000.

She is the Associate Dean, Research (Acting) of the Faculty of Engineering and Information Technology and the Director of the Decision Systems and e-Service Intelligence Research Laboratory, Centre for Quantum Computation & Intelligent Systems, University of Technology Sydney, Ultimo, NSW, Australia. Her main research interests include decision making modeling, decision support system tools, uncertain information processing, recommender systems, and e-Government and e-Service intelligence. She has published five research books and 270 papers in refereed journals and conference proceedings.

Dr. Lu has won five Australian Research Council discovery grants. She received the first UTS Research Excellent Medal for Teaching and Research Integration in 2010. She serves as the Editor-In-Chief of *Knowledge-Based Systems* (Elsevier) and Editor for a book series on (*intelligent transportation systems)* (World Scientific).