FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Tree Similarity Measure-based Recommender Systems

Dianshuang Wu

A thesis submitted for the Degree of

Doctor of Philosophy

University of Technology, Sydney
October, 2014

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

ACKNOWLEDGEMENTS

I would like to express my earnest thanks to my principal supervisor, Professor Guangquan Zhang, and my co-supervisor, Professor Jie Lu. Their comprehensive guidance has covered all aspects of my PhD study, including research methodology, research topic selection, experiments, academic writing skills and thesis writing, and even the sentence structure and formulas. Their critical comments and suggestions have strengthened my study significantly. Their strict academic attitude and respectful personality have benefited my PhD study and will be a great treasure throughout my life. Without their excellent supervision and continuous encouragement, this research could not have been finished on time. Thanks to you all for your kind help.

I am grateful to all members of the Decision Systems and e-Service Intelligent (DeSI) Lab in the centre for Quantum Computation and Intelligent Systems (QCIS) for their careful participation in my presentation and valuable comments for my research. I especially thank PhD students Mr Mingsong Mao, Mr Wei Wang, Master student Mr Yushi Zhou and other students for their contributions in the development of the recommender system softwares related to this study.

I would like to thank Ms. Barbara Munday and Ms. Sue Felix for helping me to correct English presentation problems in my publications and this thesis.

I am grateful to FEIT Travel fund, Vice-Chancellor's Postgraduate Conference Fund, and the International Postgraduate Research Scholarship (IPRS) and Australian Postgraduate Award (APA) scholarship.

Last but not least, I would like also to thank my family members. Thanks to my mother and father for their conscious encouragement and generous support.

ABSTRACT

The rapid growth of web information provides excellent opportunities for developing e-services in many applications but also caused increasingly severe information overload problems whereby users are not able to locate relevant information to exactly meet their needs efficiently by using the current Internet search functions. A personalised recommender system aims to handle this issue.

A big challenge in current recommender system research is: the items and user profiles in many recommender system applications nowadays, such as the e-business and e-learning recommender systems, are so complex that they can only be described in complicated tree structures. Therefore, the item or user similarity measure, as the core technique of the recommendation approach, becomes a tree similarity measure, which existing recommender systems cannot provide. Another challenge is that in many real life situations, online recommendations to customers in selecting the most suitable products/services are often made under incomplete and uncertain information, which needs fuzzy set theory and techniques to deal with. Thus, how to use fuzzy set techniques to handle data uncertainty issues in tree-structured items or user profiles needs to be investigated.

This research aims to handle these two challenges in both theoretical and practical aspects. It first defines a tree-structured data model, which can be used to model tree-structured items, user profiles and user preferences. A comprehensive similarity measure on tree-structured data considering all the information on tree structures, nodes' concepts, weights and values is then developed, which can be used to compute the semantic similarity between tree-structured items or users, and the matching degree of items to tree-structured user requests. Based on the tree-structured data model, the tree-structured

items and user requirements are modelled as item trees and user request trees respectively. An item tree and user request tree-based hybrid recommendation approach is then developed. To model users' fuzzy tree-structured preferences, a fuzzy preference tree model is proposed. A fuzzy preference tree-based recommendation approach is then developed. Experimental results on an Australian business dataset and the Movielens dataset show that the proposed recommendation approaches have good performance and are well-suited in dealing with tree-structured data in recommender systems. By use of the proposed tree similarity measure and recommendation approaches based on that, two real world applications, a business partner recommender system, Smart BizSeeker, and an e-learning recommender system, ELRS, are designed and implemented, which demonstrate the applicability and effectiveness of the proposed approaches.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

The rapidly increasing popularity of information and communication technologies in recent years strongly promotes the development of e-services in many application domains, such as e-commerce (Schafer, Konstan & Riedl 2001), e-business (Zhang & Wang 2005), e-learning (Drachsler, Hummel & Koper 2008a) and e-government (Guo & Lu 2007). In the meantime, the rapid growth of the web-based e-services has caused information overload whereby users are not able to effectively choose from the range of information they are exposed to. Difficulties in locating the right information for the right users will increasingly impact on the loyalty of users to use those websites, and the increases in the information overload will hinder the effectiveness of the e-services. A successful approach to solve this problem is the deployment of web/e-service personalization techniques (Adomavicius & Tuzhilin 2005a), which utilize information technology to provide personalized content and services to individuals based on their preferences and behaviours.

The recommender system is the most popular technique to implement web/e-service personalization, which has gained considerable attention and undergone rapid developments in recent years (Adomavicius & Tuzhilin 2005b; Bobadilla et al. 2013). A recommender system is a type of information system which attempts to recommend products/services (called items) to users (individuals or businesses) by predicting a user's interest in an item based on various sorts of information. The aim of recommender

systems is to provide the right information about products/services to the right customers that is relevant to their needs/interests at the right time. Recommender systems are achieving widespread success and have attracted researchers' attention in many fields, such as e-commerce/e-business (Wei, Huang & Fu 2007), e-learning (Shishehchi et al. 2011), e-government (Shambour 2012) and e-tourism (Batet et al. 2012). The most commonly used recommendation approaches include collaborative filtering (CF)-based, content-based (CB), and knowledge-based (KB) approaches, and their variations (Burke 2007). The basic idea of recommender systems is that similar users like similar items. Therefore, the similarity measure for users or items is vital in the applications of recommender systems. The CF method only uses users' ratings of items to calculate the similarity between items or users (Schafer et al. 2007), so it can deal with any kind of item (Adomavicius & Tuzhilin 2005b). However, it suffers from sparsity, scalability and cold-start problems (Adomavicius & Tuzhilin 2005b; Schafer et al. 2007). To improve the recommendation accuracy and interpretability, semantic information of users or items is used and semantic similarities are developed (Cantador 2008; Ruiz-Montiel & Aldana-Montes 2009). The usefulness of semantic similarities has been demonstrated in many recommender system applications (Lu et al. 2013; Shambour & Lu 2012).

In many recommender system applications, items or user profiles are so complex that they can only be represented as tree structures. The semantic similarity between items or users thus becomes tree similarity. For example, in telecommunication product/service recommender systems, the telecom product/service for a customer is usually presented as a package. The package contains several services, and each service contains several attributes, which construct a tree-structured data. In business partner recommender systems, a business may provide several product categories, and each product category contains several sub-categories and so on. Under most sub-categories, there are several specific products, which also form a tree structure. During the recommendation process in these applications, whether to find the similar users, or to find the similar products, or to search the most matched products to users' requirements, tree-structured data will be compared. However, to the best of available knowledge, the existing recommendation methods represent user profiles or item features just as vectors of semantic concepts

(Cantador 2008), and none of them considers tree-structured items or user profiles. To solve this problem, this study proposes and develops a set of recommendation approaches for the recommender system applications which have tree-structured items or user profiles. To evaluate the semantic similarity between tree-structured items or users effectively, a comprehensive similarity measure method and relevant algorithms on tree-structured data will be developed.

In many real situations, recommendations to users in selecting the most suitable products/services are often made under incomplete and uncertain information. In many cases, the features of items which are described by domain experts are often subjective and imprecise. The users' preferences to items 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 lends itself well to handle the fuzziness and uncertain issues in the recommendation problems. Recent research efforts have indicated that fuzzy sets, fuzzy logic and fuzzy relations are potentially within the domain of recommender systems (Leung, Chan & Chung 2006; Lu et al. 2013; Yager 2003; Zenebe, Zhou & Norcio 2010). However, an item is normally described as a single value or a vector in previous research, and tree-structured items or user profiles have not been considered to date. How to use fuzzy set techniques to handle these uncertainty issues in tree-structured items or user profiles in recommender systems, clearly, needs to be investigated.

1.2 RESEARCH QUESTIONS AND OBJECTIVES

This research aims to develop a set of recommendation approaches and systems for the applications which have tree-structured items or user profiles. To summarize, the following research questions will be answered by this research:

Question 1. How should the similarity between tree-structured data be effectively measured?

Question 2. How should the performance of a recommender system for tree-structured items be improved through utilizing the similarity measure on tree-structured data in the recommendation process?

Question 3. How should the uncertainty issues in a user's tree-structured preferences be dealt with through utilizing fuzzy set techniques, and how should the fuzzy tree-structured preferences be utilized to make recommendations?

Question 4. How should the tree similarity based recommendation approaches be applied into the real recommender system applications?

This research aims to achieve the following objectives, which are expected to answer the above research questions:

Objective 1. To develop a tree-structured data model.

This objective corresponds to research Question 1. To represent the tree-structured data emerging in recommender systems, a tree-structured data model which assigns tree nodes concepts, values and weights will be defined. The tree-structured data model can be used to model tree-structured items, user profiles and user preferences.

Objective 2. To develop a comprehensive similarity measure on tree-structured data and related computation algorithms.

This objective corresponds to research Question 1. Based on the tree-structured data model, a comprehensive similarity measure method considering all the information on tree structures, nodes concepts, weights and values will be developed. Because a tree-structured data represents specific concepts and values, the similarity between two tree-structured data will be evaluated from both the conceptual and value aspects. The conceptual similarity and value similarity between two tree-structured data will be assessed respectively, and the final similarity measure between them will be assessed as the weighted sum of their conceptual and value similarities. The similarity measure on tree-structured data can be used to compute the semantic similarity between tree-structured items or users, and the matching degree of items to tree-structured user requests.

Objective 3. To develop an item tree and user request tree-based hybrid recommendation approach.

This objective corresponds to research Question 2. To utilize fully the semantic information of tree-structured items and users and the requirement matching knowledge, an item tree and a user request tree will be defined to represent semantic features of items and users, and the proposed similarity measure on tree-structured data will be employed to evaluate the semantic similarity between item trees or user request trees. The developed hybrid recommendation approach mainly uses the features of items and the requirements of users, and also takes advantage of the merits of CF-based recommendation approaches.

Objective 4. To develop a fuzzy tree-structured preference model.

This objective corresponds to research Question 3. Fuzzy set techniques will be applied to represent users' uncertain preferences. A fuzzy preference tree will be defined to express users' tree-structured preferences. The construction and maintenance methods of fuzzy preference trees will also be developed.

Objective 5. To develop a fuzzy preference tree-based recommendation approach.

This objective corresponds to research Question 3. Based on the proposed fuzzy preference tree model and the similarity measure on tree-structured data, a fuzzy preference tree-based recommendation approach will be developed.

Objective 6. To develop a business partner recommender system.

This objective corresponds to research Question 4. A business partner recommender system called Smart BizSeeker will be developed by use of the proposed tree similarity measure and recommendation approaches based on that. The tree-structured products and buying requests of businesses will be modelled, and the framework of the system will be designed and implemented.

Objective 7. To develop an e-learning recommender system.

This objective corresponds to research Question 4. An e-learning recommender system will be developed, in which tree-structured learning activities and learner profiles

will be modelled as learning activity trees and learner profile trees. Fuzzy set techniques will be applied to handle the uncertain data in e-learning systems.

1.3 RESEARCH SIGNIFICANCE

The significance of this research work can be summarized from the theoretical, technical and practical aspects as follows:

Significance 1: theoretically, the research develops a comprehensive tree similarity measure model and relevant algorithms

The proposed similarity measure compares two tree-structured data comprehensively by considering all the information on tree structures, nodes concepts, weights and values. It can construct a maximum concept corresponding tree mapping which satisfies both structural and conceptual constraints to identify the corresponding node pairs between two trees. The comprehensive similarity measure model not only can be used in recommender systems but also can be used to compare complex tree-structured objects in other applications, such as case-based reasoning.

Significance 2: technically, the research develops an item tree and user request tree-based hybrid recommendation approach and a fuzzy preference tree-based recommendation approach for complex tree-structured items

The developed recommendation approaches utilize a tree-structured data model to express the features of items, user profiles and requirements, and evaluate the semantic similarity between items or users by use of the proposed tree similarity measure, which fully use the semantic information to make recommendations. These approaches can be used in any recommender system with tree-structured items and user profiles.

Significance 3: in practice, the research develops two recommender system softwares

The developed business partner recommender system will help businesses find a suitable business partner (buyers and suppliers), and promote the development of personalized e-business services. The e-learning recommender system will help learners

find appropriate learning activities, and enhance the development of e-learning systems. The two recommender systems also demonstrate the usability of the proposed recommendation approaches in different application domains.

1.4 RESEARCH METHODOLOGY AND PROCESS

Research methodology is the "collections of problem solving methods governed by a set of principles and a common philosophy for solving targeted problems" (Gallupe 2007). This research belongs to the information system domain. A number of research methodologies have been proposed and applied in the information system domain, such as case study, field study, design research, archival research, field experiment, laboratory experiment, survey and action research (Niu 2009; Shambour 2012; Vaishnavi & Kuechler Jr 2007).

1.4.1 RESEARCH METHODOLOGY

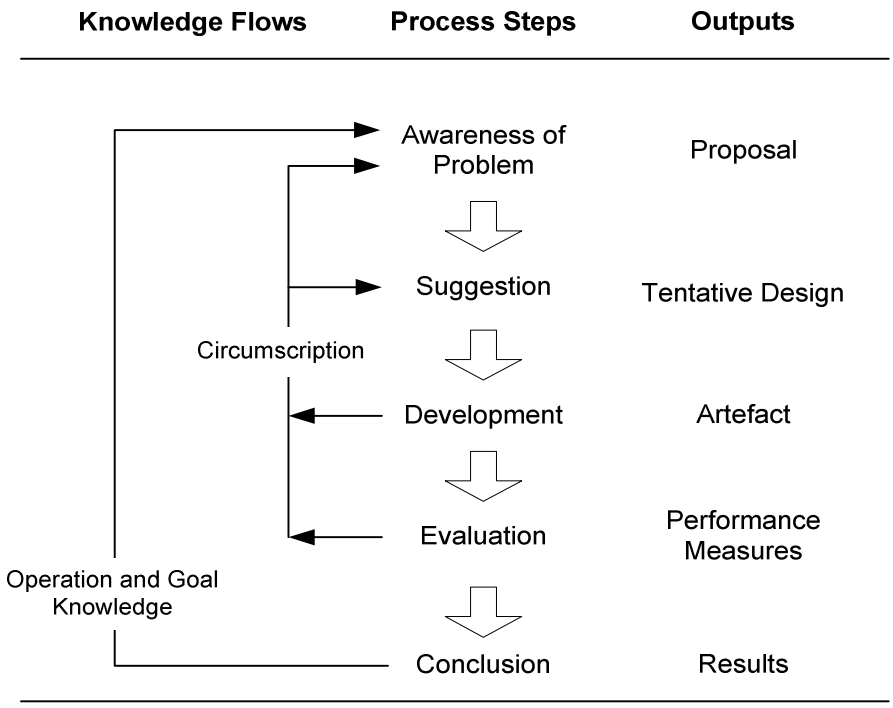


Figure 1-1. Reasoning in the general design cycle (Vaishnavi & Kuechler Jr 2007)

In this study, design research is utilized as the research methodology according to the analysis of the research questions and objectives. Design research focuses on crafting and analysing artefacts in order to gain insights into research problems. Examples of the artefacts include physical product prototypes, computer-based information systems and human-computer interfaces. The methodology of design research is illustrated in Figure 1-1. Generally speaking, a design research effort includes five basic steps (Vaishnavi & Kuechler Jr 2007).

1) Awareness of problem

This is the starting point of a design research, at which limitations of existing applications are examined and meaningful research problems are identified. The research problems reflect a gap between existing applications and the expected status. The awareness of problems can come from different sources: industry experience, observations on practical applications and literature review. The corresponding output of this step is a research proposal (Vaishnavi & Kuechler Jr 2007).

2) Suggestion

This step follows the identification of research problems, and a tentative design is suggested. The tentative design describes what the prospective artefacts will be and how they can be developed. Suggestion is a creative process during which new concepts, models and functions of artefacts are demonstrated. The resulting tentative design of this step is usually one part of the research proposal. Thus, the output of the suggestion step is fed back to the first step, so that the research proposal can be revised (Vaishnavi & Kuechler Jr 2007).

3) Development

In this step, artefacts are actually built based on the suggested design. The development of artefacts can testify to the reasonability and feasibility of the original design and improve the original design. As a result, the development of artefacts is often an iterative process in which an initial prototype is first built and this then evolves when the researcher gains a deeper understanding of the research problems. The knowledge

obtained in this step is fed back to the previous two steps, which helps researchers revise the design and the proposal (Vaishnavi & Kuechler Jr 2007).

4) Evaluation

This step considers the evaluation of the developed artefacts. The performance of artefacts can be evaluated according to criteria defined in the research proposal and the suggested design. The evaluation results, which might or might not meet the expectations, are fed back to the first two steps. Thus, the proposal and design might be revised and the artefacts might be improved (Vaishnavi & Kuechler Jr 2007).

5) Conclusion

This is the final step of a design research effort. A conclusion or end is reached as a result of satisfaction with the evaluation results of the developed artefacts. There might still be deviations between the suggested proposal and the artefacts that are actually developed. However, a design research effort concludes as long as the developed artefacts are considered as "good enough" (Vaishnavi & Kuechler Jr 2007).

1.4.2 RESEARCH PROCESS

This research was planed according to the methodology of design research. First, a subject was chosen as a very broad research topic of this research. A literature review of previous research in the topic area was conducted, and existing literature was retrieved and critically reviewed. The results of the literature review helped to identify specific research questions to be directly addressed in this research. As the research questions grew clearer and more definite, more literature closely related to the research questions was reviewed. Because the existing work in the literature lacks the ability to deal with tree-structured data in recommender systems, this research proposed tree-structured data model, presented tree similarity measures, and developed recommendation approaches based on the tree similarity measure. The proposed models and approaches were implemented and evaluated. According to the methodology of design research, this research is an iterative process. As indicated in Figure 1-1, the output of each research step might be fed back to its previous step when deviations between expectations and

evaluation results are found. Through the feedback, research outcomes are progressively improved until satisfying results are drawn from evaluations. The developed methods were then implemented in two recommender system softwares. Finally, writing up the PhD thesis is done at the end of the research.

1.5 THESIS STRUCTURE

This thesis contains eight chapters. Chapter 1 presents the research background, research questions, objectives, significance, research methodology and process, and the thesis structure. Chapter 2 presents the literature relevant to this study, including recommender system techniques and applications, and similarity measures on tree-structured data. Chapter 3 proposes a comprehensive similarity measure on tree-structured data. Chapters 4 and 5 are based on Chapter 2 and Chapter 3, and each chapter develops a recommendation approach. Chapter 4 develops an item tree and user request tree-based hybrid recommendation approach. Chapter 5 develops a fuzzy preference tree-based recommendation approach. Chapter 6 and 7 are based on the methods developed in Chapter 4 and 5, and each chapter presents an application. Chapter 6 presents the design and implementation of a business partner recommender system – Smart BizSeeker, and Chapter 7 presents the design and implementation of an e-learning recommender system (ELRS). Chapter 8 presents the conclusions and further study recommendations. The structure of the thesis is shown in Figure 1-2.

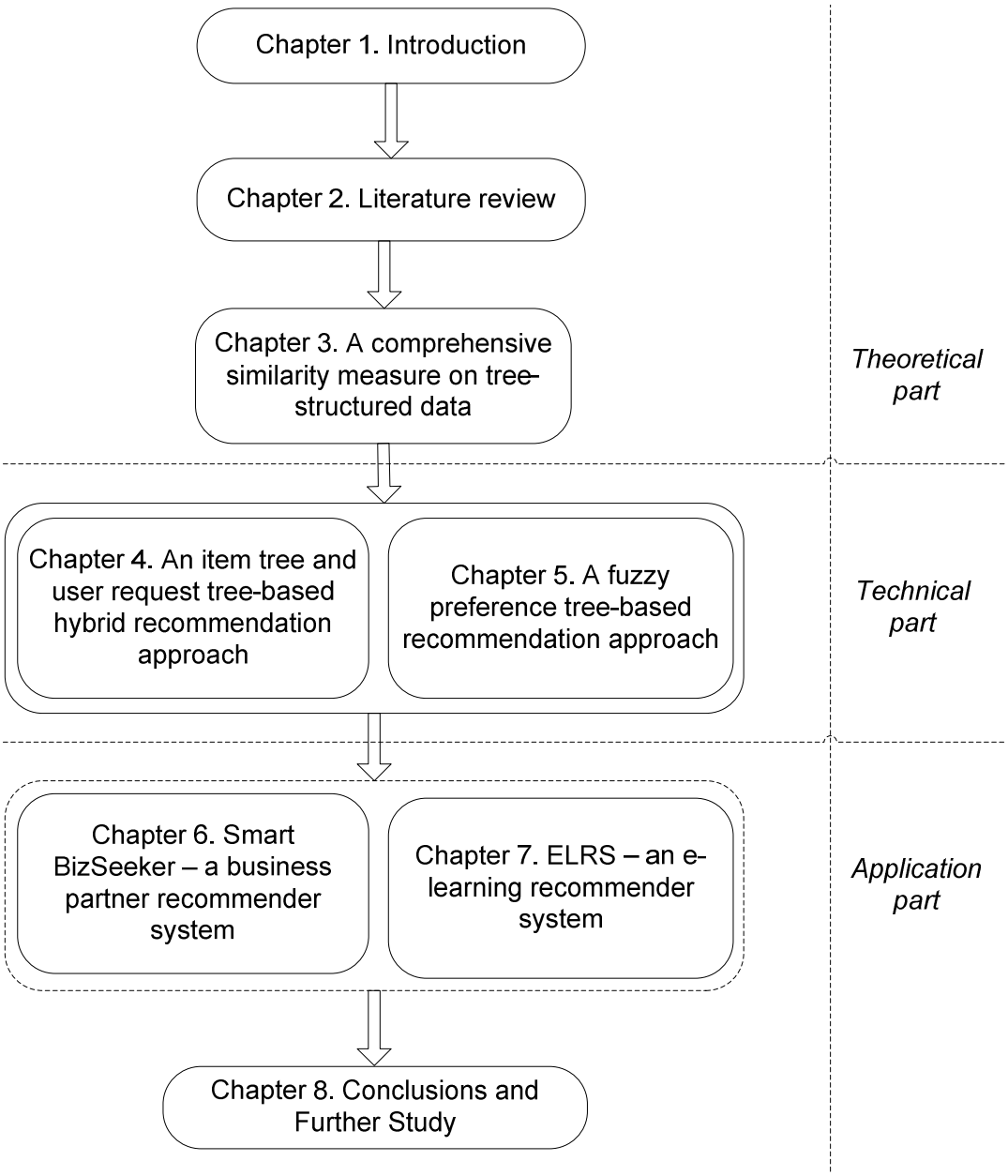


Figure 1-2. Thesis structure

1.6 PUBLICATIONS RELATED TO THIS THESIS

Below is a list of the refereed international journal and conference papers associated with my PhD research that have been submitted, accepted and published:

Refereed international journal publications:

- Papers published:

1) Wu, D., Zhang, G. & Lu, J. 2014, 'A fuzzy preference tree-based recommender system for personalized business-to-business e-services', *IEEE Transactions on Fuzzy Systems*, online on 04 April 2014, in press. (ERA Tier A*)

2) Wu, D., Lu, J. & Zhang, G. 2011, 'Similarity measure models and algorithms for hierarchical cases', *Expert Systems with Applications*, vol. 38, no. 12, pp. 15049-56.

3) Zhang, Z., Lin, H., Liu, K., Wu, D., Zhang, G. & Lu, J. 2013, 'A hybrid fuzzy-based personalized recommender system for telecom products/services', *Information Sciences*, vol. 235, no. 0, pp. 117-29.

4) Niu, L., Lu, J., Zhang, G. & Wu, D. 2013, 'FACETS: A cognitive business intelligence system', *Information Systems*, vol. 38, no. 6, pp. 835-62. (ERA Tier A*)

- Papers submitted:

5) Lu, J., Wu, D., Mao, M., Wang, W. & Zhang, G. 2014, 'Recommender system application developments: a survey', submitted to *Decision Support Systems*. (ERA Tier A*)

6) Wu, D., Lu, J. & Zhang, G. 2014, 'A fuzzy tree matching-based personalized e-learning recommender system', submitted to *IEEE Transactions on Fuzzy Systems*. (ERA Tier A*)

7) Wu, D., Zhang, G. & Lu, J. 2014, 'Product and request trees-based personalized business partner recommender system', submitted to *Information Systems*. (ERA Tier A*)

Refereed international conference publications:

8) Wu, D., Zhang, G. & Lu, J. 2014, 'A fuzzy tree matching-based personalised e-learning recommender system', *The 2014 IEEE World Congress on Computational Intelligence*, Beijing, 6-11 July. (ERA Tier A)

9) Wu, D., Zhang, G. & Lu, J. 2013, 'A fuzzy tree similarity measure and its application in telecom product recommendation', *The 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Manchester, pp. 3483-8.

10) Wu, D., Zhang, G. & Lu, J. 2013, 'A fuzzy tree similarity based recommendation approach for telecom products', *The 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, eds W. Pedrycz & M.Z. Reformat, Edmonton, Canada, pp. 813-8.

11) Wu, D., Zhang, G., Lu, J. & Halang, W.A. 2012, 'A similarity measure on tree structured business data', *ACIS 2012: Location, location, location: Proceedings of the 23rd Australasian Conference on Information Systems 2012*, ACIS, pp. 1-10. (ERA Tier A)

12) Wu, D. & Zhang, G. 2011, 'Fuzzy similarity measure model for trees with duplicated attributes', in S. Li, X. Wang, Y. Okazaki, J. Kawabe, T. Murofushi & L. Guan (eds), *Nonlinear Mathematics for Uncertainty and its Applications*, vol. 100, Springer Berlin / Heidelberg, pp. 333-40.

13) Wu, D., Lu, J. & Zhang, G. 2010, 'A hybrid recommendation approach for hierarchical items', *The 2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 492-7.

14) Wu, D., Lu, J., Zhang, G. & Lin, H. 2010, 'A fuzzy matching based recommendation approach for mobile products/services', *2010 10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 645-50.

CHAPTER 2
LITERATURE REVIEW

This chapter presents a discussion of relevant work in connection with this research. In Section 2.1, the concepts and techniques of recommender systems are expatiated. Section 2.2 reviews the recommender system applications. Section 2.3 reviews the tree similarity measures which are related to this research on the measurement of trees.

2.1 RECOMMENDATION TECHNIQUES

Recommender systems can be defined as programs which attempt to recommend the most suitable items (products or services) to particular users (individuals or businesses) by predicting a user's interest in an item based on related information about items, users and the interactions between items and users (Bobadilla et al. 2013). The aim of developing a recommender system is to reduce information overload by retrieving the most relevant information and services from a very large amount of data, thereby providing personalized services. The most important feature of a recommender system is its ability to "guess" a user's preferences and interests by analyzing this user and/or other users' behaviors to generate personalized recommendations (Amoroso & Reinig 2004).

E-service personalization techniques are typified by recommender systems, which have gained much attention in the past 20 years (Adomavicius & Tuzhilin 2005b). Early research in recommender systems grew out of information retrieval and filtering research (Goldberg et al. 1992), and recommender systems emerged as an independent research area in the mid-1990s when researchers started to focus on recommendation problems that explicitly rely on the ratings structure (Adomavicius & Tuzhilin 2005b). There have

been many recommendation techniques developed since the emergence of recommender systems, including the classic techniques, such as collaborative filtering (CF) (Schafer et al. 2007), content-based (CB) (Pazzani & Billsus 2007) and knowledge-based (KB) (Burke 2000) techniques, and many recently developed advanced recommendation techniques, such as social network-based recommender systems (He & Chu 2010), fuzzy recommender systems (Lu et al. 2013; Zhang et al. 2013), context aware-based recommender systems (Adomavicius & Tuzhilin 2011) and group recommender systems (Masthoff 2011). Recommender systems have been widely used in a variety of web-based applications in e-commerce (Markellou et al. 2005), e-learning (Lu 2004), e-government (Lu et al. 2010), and e-tourism (Batet et al. 2012) as well as in areas such as the recommendation of news, movies, books, videos and online research papers.

In this section, the main recommendation techniques, including traditional methods such as content-based, collaborative filtering based, knowledge-based, and hybrid methods, and recently developed advanced methods, such as computational intelligence based, fuzzy set-based, social network-based, context awareness-based, and group recommendation approaches, will be reviewed.

2.1.1 CONTENT-BASED RECOMMENDATION TECHNIQUES

Content-Based (CB) recommendation techniques recommend articles or commodities that are similar to items previously preferred by a specific user (Pazzani & Billsus 2007). The basic principles of CB recommender systems are: 1) To analyse the description of the items preferred by a particular user to determine the principal common attributes (preferences) that can be used to distinguish these items. These preferences are stored in a user profile. 2) To compare each item's attributes with the user profile so that only items that have a high degree of similarity with the user profile will be recommended (Pazzani & Billsus 2007).

In CB recommender systems, two techniques have been used to generate recommendations. One technique generates recommendations heuristically using the traditional information retrieval methods, such as the cosine similarity measure. The other technique generates recommendations using statistical learning and machine

learning methods. This technique mainly builds models that can learn users' interests from the users' historical data (training data). Creating a learning model of a user's preferences is a form of classification learning. The algorithms used in classification learning, such as decision tree, naive Bayesian and *k*-nearest neighbours, create a probability function that has the potential to provide the probability estimation for a user's interest to an unseen item. The attained probability can be used to provide users with a sorted list of recommendations (Pazzani & Billsus 2007). Some examples of CB recommender systems are WebWatcher (Armstrong et al. 1995) and Websail (Chen et al. 2000). In previous researches, item attributes and user profiles are represented as flat vectors, and tree-structured items and user profiles are not dealt with.

The advantages of the CB recommender systems are that they adopt semantic content of items and recommend to a specific user items similar to the preferred items in his/her profile. As a result, a CB recommender system would be able to recommend new items and unpopular items. Furthermore, it can provide a clarification of recommended items by listing content-features based on which an item is to be recommended. It does not need to have information about preferences of other users in making recommendations, so it does not suffer from the sparseness problem associated with CF systems. One of the main limitations of CB recommender systems is the new user problem. The CB recommendation approach is not able to offer accurate recommendations to a new user since he/she has few rated items. The CB approach also has the overspecialization problem. It can only recommend items to a user according to the preferred items in his/her user profile so it cannot recommend items outside the user's profile. Additionally, in some particular cases, it may not be desirable for a recommender system to recommend items which are too similar to users, such as different news articles that describe the same event. Another limitation of the CB approach is the item content dependency problem. As the CB approach makes recommendations according to contents of items, it is hard to use CB to recommend items which cannot be represented as keywords, such as image and movies. The CB approach cannot distinguish between the items which are represented by the same set of content features.

2.1.2 COLLABORATIVE FILTERING-BASED RECOMMENDATION TECHNIQUES

The collaborative filtering (CF) based recommendation techniques help people to make choices based on the opinions of other people who share similar interests (Deshpande & Karypis 2004). Resnick and Varian stated that the CF approach built on a significant assumption that "a good way to find interesting content is to find other people who have similar interests, and then recommend titles that those similar users like" (Resnick & Varian 1997). The CF approach not only provides new suggestions and recommendations to people, but also tries to provide the right information to the right user (Shardanand & Maes 1995). It has been demonstrated that the CF recommendation approach is the most successful and widely used approach for recommender systems (Herlocker et al. 1999; Huang, Zeng & Chen 2007; Schafer et al. 2007). CF-based recommender systems have been developed and used in many fields including recommending news (Resnick et al. 1994), articles, movies, music, products, books (Linden, Smith & York 2003) and web pages.

The CF-based techniques can be grouped into two general classes: memory-based and model-based (Adomavicius & Tuzhilin 2005b; Breese, Heckerman & Kadie 1998). Memory-based techniques basically are heuristics that make rating predictions based on the entire collection of previously rated items by the users, which include user-based and item-based CF approaches (Sarwar et al. 2001). The model-based CF algorithms use the existing ratings to build a model which is then used to make predictions for un-rated items. Building a model is a fundamental step for the model-based recommendation techniques. Different machine learning algorithms are used to accomplish the model building process such as the Bayesian network (Breese, Heckerman & Kadie 1998), clustering (Jia, Jin & Liu 2010) and rule-based techniques. These algorithms mainly use a probabilistic approach to compute prediction values for un-rated items (Adomavicius & Tuzhilin 2005b; Schafer et al. 2007).

The memory-based CF technique can be divided into user-based and item-based CF approaches (Sarwar et al. 2001). The user-based CF approach recommends those items to

an active user which are most liked by the user's nearest neighbours. First, it analyses the user-item matrix and creates a vector for each user which contains all the user's ratings. Then, it computes the similarity between the active user's vector and the vectors of other users. Next, the most similar users to the active user are selected as the user's nearest neighbours. Finally, predictions are generated using a weighted average of the neighbours' ratings to items. The similarity measure between users is the core technique in the user-based CF approach. There are a number of similarity measures, such as Pearson correlation-based similarity (Resnick et al. 1994) and constrained Pearson correlation-based (CPC) similarity, which are described in detail as follows:

1) The Pearson correlation-based similarity between two users a and b is calculated by

$$s_{pc}(a,b) = \frac{\sum_{i \in I_{a,b}} (r_{a,i} - \overline{r_a}) \times (r_{b,i} - \overline{r_b})}{\sqrt{\sum_{i \in I_{a,b}} (r_{a,i} - \overline{r_a})^2} \times \sqrt{\sum_{i \in I_{a,b}} (r_{b,i} - \overline{r_b})^2}}, \quad (2\text{-}1)$$

where $I_{a,b}$ is the set of co-rated items by both users a and b, $r_{a,i}$ and $r_{b,i}$ represent the ratings of users a and b on item i respectively, and $\overline{r_a}$ and $\overline{r_b}$ represent the average ratings of users a and b on all of the items in $I_{a,b}$.

2) The constrained Pearson correlation (CPC)-based similarity between two users a and b is calculated by

$$s_{cpc}(a,b) = \frac{\sum_{i \in I_{a,b}} (r_{a,i} - r_{mid}) \times (r_{b,i} - r_{mid})}{\sqrt{\sum_{i \in I_{a,b}} (r_{a,i} - r_{mid})^2} \times \sqrt{\sum_{i \in I_{a,b}} (r_{b,i} - r_{mid})^2}}, \quad (2\text{-}2)$$

where $I_{a,b}$ is the set of co-rated items by both users a and b, $r_{a,i}$ and $r_{b,i}$ represent the ratings of users a and b on item i respectively, and r_{mid} represents the mid-point of the rating scale.

The item-based CF approach recommends items to an active user that are similar to the items the user has rated highly in the past. First, the item-based CF approach creates a

vector for each item which contains the ratings from all users to the item. For a target item to an active user, it then computes the similarity between the rated items of the active user and the target item. The most similar rated items to the target item are then selected as the item's nearest neighbours. Finally, the prediction for the target item is generated by taking a weighted average of the active user's ratings on the neighbour items. It is found that the item-based algorithms are able to provide the same quality of provided services as the user-based algorithm but with less online computation because the relationships between items are relatively static compared with the relationships between users (Sarwar et al. 2001). The similarity between items can be calculated by cosine-based similarity (Sarwar et al. 2001) and adjusted cosine-based similarity measures (Deshpande & Karypis 2004), which are described in detail as follows:

1) The cosine-based similarity between two items *i* and *j* is calculated by:

$$s_{\cos}(i,j) = \frac{\sum_{u \in U_{i,j}} r_{u,i} \times r_{u,j}}{\sqrt{\sum_{u \in U_{i,j}} r_{u,i}^{2}} \times \sqrt{\sum_{u \in U_{i,j}} r_{u,j}^{2}}}, \tag{2-3}$$

where $U_{i,j}$ are the users who rated both items *i* and *j*, $r_{u,i}$ and $r_{u,j}$ represent the ratings of user *u* to items *i* and *j* respectively.

2) The adjusted cosine-based similarity between two items *i* and *j* is calculated by:

$$s_{acos}(i,j) = \frac{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r_u}) \times (r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in U_{i,j}} (r_{u,i} - \overline{r_u})^2} \times \sqrt{\sum_{u \in U_{i,j}} (r_{u,j} - \overline{r_u})^2}}, \tag{2-4}$$

where $U_{i,j}$ are the users who rated both items *i* and *j*, $r_{u,i}$ and $r_{u,j}$ represent the ratings of user *u* to items *i* and *j* respectively, and $\overline{r_u}$ represents the average ratings of user *u* to all of the items.

When calculating the similarity between items using the above measures, only users who have rated both items are considered. This can have an impact when items which have received a very small number of ratings express a high level of similarity with other

items (Shambour & Lu 2011a). To improve similarity accuracy, the difference between the number of common users who rated both items and the total number of users who rated each item individually was considered, and an enhanced item-based CF approach was presented by combining the adjusted cosine approach with the Jaccard metric as a weighting scheme (Shambour & Lu 2011a). To compute the similarity between users, the Jaccard metric was used as a weighting scheme with the CPC to obtain a weighted CPC measure (Shambour & Lu 2011c). To deal with the disadvantage of the single-rating based approach, multi-criteria collaborative filtering was developed (Shambour & Lu 2010, 2011b).

The main advantage of using CF recommendation techniques is that they work for any type of items without the need to extract features related to the items. It not only suggests similar items according to user interests and preferences, but also suggests new items based on the other people who have the same or similar tastes and interests to a specific user. The major limitations of CF methods are data sparseness and cold-start problems (Adomavicius & Tuzhilin 2005b; Sarwar et al. 2001; Schafer et al. 2007). The data sparseness 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 to the number of ratings that need to be predicted, a recommender system becomes unable to locate similar neighbours and produces poor recommendations. The cold-start (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 neighbours using rating similarity so 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 few ratings for CS items, CF-based approaches cannot appropriately locate similar neighbours using rating similarity and will be unlikely to recommend them (Adomavicius & Tuzhilin 2005b; Rodríguez et al. 2010).

2.1.3 KNOWLEDGE-BASED RECOMMENDATION TECHNIQUES

Knowledge-Based (KB) recommendation techniques offer items to users based on knowledge about the users and/or items. Usually, a KB recommender system retains a functional knowledge base that describes how a particular item meets a specific user's requirement, which can be performed based on inferences about the relationship between a user's need and a possible recommendation (Burke 2002).

Case-based reasoning (CBR) technique is a common example of KB recommendation techniques (Smyth 2007). Case-based reasoning systems rely on the idea of using the past problem solving experiences as a primary source to solve the new problem (Aamodt & Plaza 1994). It is represented by a four-step (4Rs) cycle: retrieve, reuse, revise and retain (Aamodt & Plaza 1994). The past problem solutions are stored in a database as cases, each case is typically made up of two parts, the specification part and the solution part. The specification part describes the problem at hand, whereas the solution part describes the solution that used to solve this problem. A new problem is solved by retrieving a case whose specification is similar to the current problem and then fitting the attained solution to match the current problem. Case-based recommender systems represent items as cases and generate the recommendations by retrieving the most similar cases to the user's query or profile. In these systems, items are described in terms of a well defined set of features (e.g., price, colour, make, etc.) (Smyth 2007). Case-based recommenders borrow heavily from the core concepts of retrieval and similarity in case-based reasoning. The case-based recommender system can be seen as a special type of content based recommender systems. There are two important ways in which case-based recommender systems can be distinguished from other types of content systems: (1) the manner in which products are represented; and (2) the way in which product similarity is assessed (Smyth 2007). Case-based recommender systems rely on more structured representations of item content. In the existing case-based recommender systems, cases are usually represented as fixed predefined feature vectors. There appears to have been no system to deal with hierarchical tree-structured cases yet. The second important distinguishing

feature of case-based recommender systems relates to their use of various sophisticated approaches to similarity assessment when it comes to judging which cases to retrieve in response to some user query. Similarity assessment is clearly a key issue for case-based reasoning and case-based recommender systems. The existing similarity measures focus on feature vector represented cases. The similarity measure for tree-structured cases still needs to be researched.

The underlying semantic knowledge associated with users and items has been exploited to generate recommendations in certain types of KB recommender systems called semantic-based recommender systems (Ruiz-Montiel & Aldana-Montes 2009). The semantic knowledge about items consists of the attributes of the items, the relation between items, and the relation between items and meta-information (Resnik 1995). Taxonomies and ontologies as the major source of semantic information can be taken advantage of in recommender systems, since they provide a means of discovering and classifying new information about the items to recommend, user profiles and even context (Ruiz-Montiel & Aldana-Montes 2009). For example, product taxonomies have been presented in several recommender systems to utilize the relevant semantic information to improve recommendation quality (Albadvi & Shahbazi 2009; Cho & Kim 2004; Hung 2005). In a business environment, product categories are used to evaluate the semantic similarity between businesses (Guo & Lu 2007; Lu et al. 2010; Shambour & Lu 2012). Ontology-based Recommender Systems (Middleton, Roure & Shadbolt 2009) are typical KB recommender systems. An ontology is a conceptualization of a domain into a human-understandable, but machine-readable format consisting of entities, attributes, relationships, and axioms (Guarino & Giaretta 1995). Ontology-based recommender systems classify items using ontological classes, represent user profiles in terms of ontological terms, use ontological inference to improve user profiling, and use ontological knowledge to bootstrap a recommender system and support profile visualization to improve profiling accuracy. The semantic similarity between items can be calculated based on the domain ontology (Al-hassan, Lu & Lu 2011). Ontologies are usually expressed as hierarchical tree structures. However, the items or user profiles are not tree-structured in current research. Cantador (2008) used ontologies to represent the

semantics and exploit the semantic relations in recommender systems, and developed a hybrid recommendation approach that merges semantic content-based and collaborative information. The user preferences and item features are modelled as vectors of ontological concepts, but tree structured users or items are not considered. The usage of the semantic information can provide additional explanation about 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 (Shambour & Lu 2012). In this study, to make accurate recommendations of tree-structured items, the semantic information of tree-structured items or user profiles should be fully considered. A comprehensive semantic similarity measure on tree-structured data will be proposed.

The KB recommender systems have some advantages when compared with CB and CF-based recommender systems. As KB recommender systems exploit deep knowledge about the product/service domain, they are able to support intelligent explanations and product recommendations which are determined by a set of explicitly defined constraints (Felfernig et al. 2006; Felfernig et al. 2008). KB approaches are in the majority of cases applied for recommending complex products and services such as consumer goods, technical equipment, or financial services (Felfernig et al. 2008). A KB recommender system has no cold-start problem as a new user can get recommendations based on a simple knowledge of his/her interests. A KB recommender system generates recommendations by computing the similarities between the existing cases and the user's request, so it doesn't require the user to rate or purchase many items in order to generate good recommendations. KB recommender systems still have some limitations (Burke 2002; Leung 2009). For instance, a KB recommender system requires extensive effort to acquire and maintain the knowledge, and to retain the information about items and users for making recommendations. It also requires more feedback and involvement from an active user in order to make an appropriate recommendation for the user.

2.1.4 HYBRID RECOMMENDATION TECHNIQUES

To achieve higher performance and overcome the drawbacks of traditional recommendation techniques, a hybrid recommendation technique that combines the best features of two or more recommendation techniques into one hybrid technique has been proposed (Burke 2007). According to Burke (2007), there are seven basic hybridization mechanisms of combinations used in recommender systems to build hybrids:

1) weighted: scores of each of the recommendation approaches are combined numerically to produce a single prediction (Mobasher, Jin & Zhou 2004);

2) mixed: results from different recommendation approaches are presented together, either in a single presentation or combined in separate lists (Smyth & Cotter 2000);

3) switching: one of the recommendation approaches is selected to make the prediction when certain criteria are met using decision criteria (Billsus & Pazzani 2000);

4) feature combination: a single prediction algorithm is provided with features from different recommendation approaches (Basu, Hirsh & Cohen 1998);

5) feature augmentation: the output from one recommendation approach is fed to another (Melville, Mooney & Nagarajan 2002);

6) cascade: one recommendation approach refines the recommendations produced by another (Burke 2002);

7) meta-level: the entire model produced by one recommendation approach is utilized by another (Pazzani 1999).

The most common practice in the existing hybrid recommendation techniques is to combine the CF recommendation techniques with the other recommendation techniques in an attempt to avoid cold-start, sparseness and/or scalability problems (Adomavicius & Tuzhilin 2005b; Bellogin et al. 2013).

2.1.5 COMPUTATIONAL INTELLIGENCE-BASED RECOMMENDATION TECHNIQUES

In recommender systems, computational intelligence (CI) techniques, such as Bayesian techniques, artificial neural networks, clustering techniques, genetic algorithms, and fuzzy set techniques, are widely used to construct recommendation models. Bayesian classifiers are popular for model-based recommender systems (Amatriain et al. 2011) and are often used to derive the model for CB recommender systems. They have also been used in the CF setting to introduce user control to improve performance (Pronk et al. 2007). An artificial neural network (ANN) is used to construct model-based recommender systems (Amatriain et al. 2011), and applied in many applications, such as web navigation (Berka et al. 2002), TV recommendation (Hsu et al. 2007), and movie recommendation (Christakou, Vrettos & Stafylopatis 2007). Clustering techniques can be used to reduce the computation cost for finding the *k*-nearest neighbours (Amatriain et al. 2011), to smooth the unrated data for individual users (Xue et al. 2005), to address the cold start problem in recommender systems by grouping items (Sobhanam & Mariappan 2013), and to identify and solve the gray-sheep users problem (Ghazanfar & Prügel-Bennett 2014). Genetic algorithms (GA) have mainly been used in two aspects of recommender systems (Bobadilla et al. 2011): clustering (Kim & Ahn 2005, 2008) and hybrid user models (Al-Shamri & Bharadwaj 2008). A GA-based K-means clustering is applied to a real-world online shopping market segmentation case for personalized recommender systems in (Kim & Ahn 2005). A genetic algorithm method is presented for obtaining optimal similarity functions (Bobadilla et al. 2011). Fuzzy set theory offers a rich spectrum of methods for the management of non-stochastic uncertainty. It is well suited to handling imprecise information, the un-sharpness of classes of objects or situations, and the gradualness of preference profiles (Zadeh 1965; Zenebe & Norcio 2009). Therefore, the use of fuzzy set techniques provides opportunities to model item features and user preferences under uncertainty (Zenebe, Zhou & Norcio 2010) and to handle the fuzziness and uncertain issues in the recommendation process.

Each CI technique has its own merits and its special usage in recommender systems. This study mainly focuses on fuzzy set techniques to generate recommendations to customers based on incomplete and uncertain information. The fuzzy set techniques in recommender systems are reviewed in the next sub-section.

2.1.6 FUZZY SET TECHNIQUES IN RECOMMENDER SYSTEMS

2.1.6.1 PRELIMINARIES ON FUZZY SETS

In this sub-section, some basic definitions and properties of fuzzy sets from Zadeh (1965; 1975), Sakawa (1993), Zhang (1998), and Zhang and Lu (2003; 2004; 2009) are reviewed.

Definition 2-1. A fuzzy set $\tilde{A}$ in a universe of discourse X is characterized by a membership function $\mu_{\tilde{A}}(x)$ which associates with each element x in X a real number in the interval [0, 1]. The function value $\mu_{\tilde{A}}(x)$ is termed the grade of membership of x in $\tilde{A}$.

Definition 2-2. A fuzzy set $\tilde{A}$ in a universe of discourse X is convex if and only if for any $x_1, x_2 \in X$, $\mu_{\tilde{A}}(\lambda x_1 + (1-\lambda)x_2) \geq \min\left(\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2)\right)$, where $\lambda \in [0,1]$.

Definition 2-3. A fuzzy set $\tilde{A}$ in a universe of discourse X is called a normal fuzzy set implying that there exists $x_0 \in X$ such that $\mu_{\tilde{A}}(x_0) = 1$.

Definition 2-4. A fuzzy number $\tilde{a}$ is a fuzzy subset on the space of real number R that is both convex and normal.

Definition 2-5. The λ – cut of fuzzy number $\tilde{a}$ is defined as $a_\lambda = \{x; \mu_{\tilde{a}}(x) \geq \lambda, x \in R\}$, a_λ is a non-empty bounded closed interval contained in X and it can be denoted by $a_\lambda = [a_\lambda^L, a_\lambda^R]$, a_λ^L and a_λ^R are the lower and upper bounds of the closed interval, respectively.

Let $F(R)$ be the set of all fuzzy numbers. By the decomposition theorem of fuzzy set, we have $\tilde{a} = \bigcup_{\lambda \in (0,1]} \lambda[a_\lambda^L, a_\lambda^R]$ for every $\tilde{a} \in F(R)$.

Definition 2-6. A triangular fuzzy number $\tilde{a}$ can be defined by a triplet (a_0^L, a, a_0^R) and the membership function $\mu_{\tilde{a}}(x)$ is defined as:

$$\mu_{\tilde{a}}(x)=\begin{cases}(x-a_0^L)/(a-a_0^L), & a_0^L \le x \le a,\\ (a_0^R-x)/(a_0^R-a), & a \le x \le a_0^R,\\ 0, & others.\end{cases} \tag{2-5}$$

Let $F^*(R)$ be the set of all finite fuzzy numbers on R.

Definition 2-7. Let $\tilde{a}, \tilde{b} \in F^*(R)$, then the quasi-distance function of $\tilde{a}$ and $\tilde{b}$ is defined as:

$$d(\tilde{a},\tilde{b})=\left(\int_0^1 \tfrac{1}{2}[(a_\lambda^L-b_\lambda^L)^2+(a_\lambda^R-b_\lambda^R)^2]d\lambda\right)^{\frac{1}{2}}. \tag{2-6}$$

Definition 2-8. A linguistic variable is a variable whose values are words or sentences in a natural or artificial language. A linguistic variable is characterized by a quintuple (χ, $T(\chi)$, U, G, M) in which χ is the name of the variable; $T(\chi)$ is the term-set of χ, that is, the collection of its linguistic values; U is a universe of discourse; G is a syntactic rule which generates the terms in $T(\chi)$; and M is a semantic rule which associates with each linguistic value X its meaning, $M(X)$, where $M(X)$ denotes a fuzzy subset of U.

2.1.6.2 FUZZY SET TECHNIQUES IN RECOMMENDER SYSTEMS

Fuzzy set theory and fuzzy logic provide a way to quantify the uncertainty, vagueness and imprecision. The main applications of fuzzy set techniques in recommender systems include the following aspects:

1) Item representation using fuzzy sets

Item representation is an important issue in recommender systems, but item features are defined and rated by domain experts; this process is usually subjective and uncertain. Fuzzy sets are used to deal with this issue. Yager (2003) defined a set of primitive assertions/statements to describe items, which is denoted as $A=\{A_1,\dots,A_n\}$. Each item can be seen as a fuzzy subset over the space A. Zenebe et al. (2009; 2010) defined a feature set for items, and a set of values for each feature. The items are represented as the

fuzzy sub set over the values, denoted by a feature vector. Lu et al. (2009) proposed a fuzzy semantic product relevance (FSPR) model to represent the relevance of a business partner's products to a target product category using a set of linguistic terms. Cao and Li (2007) used linguistic terms for domain experts to evaluate the features of consumer electronic products. These linguistic terms are dealt with by fuzzy numbers. Herrera and Martinez proposed a 2-tuple fuzzy linguistic model (FLM) (Herrera & Martinez 2000) as a continuous model of representation of information that allows a reduction in the loss of information typical of other fuzzy linguistic approaches, and also proposed a linguistic hierarchy (Herrera & Martinez 2001) to represent multi-granular linguistic information. Based on the multi-granular FLM, a recommender system for research resources called SIRE2IN was developed in (Porcel, López-Herrera & Herrera-Viedma 2009), a multi-disciplinary recommender system to advise research resources in university digital libraries was presented in (Porcel, Moreno & Herrera-Viedma 2009), and a fuzzy linguistic recommender system as a communication tool for researchers interested in common research lines based on the Google Wave technology in university digital libraries was proposed in (Serrano-Guerrero et al. 2011). The resources in these systems are represented as 2-tuple linguistic vectors.

2) User preference representation using fuzzy sets

User preference can be expressed as user to item ratings, items preferred by the user, the explicit requirements of the user and user profiles.

- *User to item rating using fuzzy sets*: Lu et al. (2009) defined a linguistic term set for users to represent the preference degrees to items. The fuzzy numbers are applied to deal with the linguistic terms. de Campos et al. (2008) employed fuzziness in the rating process. The user rate, which is represented as a fuzzy label, is considered as a fuzzy subset of the label set.

- *Fuzzy preference from experienced items:* Yager (2003) assumed that an item experienced by a user would be rated with a value from the unit interval, which indicates the score the user has attributed to the item. The preferred items are then viewed as a fuzzy subset over the items experienced by the user, using the ratings as the membership

function. Zenebe et al. (2010) inferred user preference to item features from the experienced items and associated ratings. Four preference categories, preferred, non-preferred, indifferent and unknown, were defined. The user preference is represented as four preference vectors, where the element represents the membership of the item feature value belonging to the preference category.

- *Fuzzy user requirements:* Cao and Li (2007) allowed users to use linguistic terms, which are described by fuzzy numbers, to express their needs for the item features. Porcel et al. developed a method to deal with incomplete information in fuzzy linguistic recommender systems and allowed users to provide their preferences by means of an incomplete fuzzy linguistic preference relation (Porcel & Herrera-Viedma 2010). To represent the user preferences, Yager (2003) introduced a primal preference module (PPM) and a basic preference module (BPM). The user preference profile is expressed as a set of BPMs. The overall recommendation degree of an item is the combination of its recommendation degrees of the BPM set.

- *Fuzzy user profile*: Al-Shamri and Bharadwaj (2008) pointed out that the crisp description of the user profile, such as age, does not reflect a rationale for human decisions. In their model, the attributes of a user are fuzzified. Each attribute is associated with several fuzzy sets, and a membership vector is obtained for a user.

3) Fuzzy inference

With the fuzzy representation of items and user preferences, recommendations are made by fuzzy computation and inference. A set of fuzzy set theoretical similarity measures to calculate the similarity between items represented by fuzzy sets was introduced in (Zenebe & Norcio 2009). The Pearson correlation coefficient is used to calculate CF fuzzy similarity between two items' linguistic rating vectors (Lu, Shambour & Zhang 2009). The standard vector-based cosine similarity is used to calculate the semantic fuzzy similarity between two items' linguistic semantic relevance vectors (Lu, Shambour & Zhang 2009). A new linguistic similarity measure based on cosine measure to deal with 2-tuple linguistic vectors is introduced by Porcel et al. to calculate the matching degree between user requirement and items (Porcel & Herrera-Viedma 2010;

Porcel, López-Herrera & Herrera-Viedma 2009). Fuzzy rules (Yager 2003) and fuzzy inference (Cornelis et al. 2007) are also used to calculate the recommendation confidence scores of the items. In (Cornelis et al. 2007), the user preferences are represented as two fuzzy relations from user set to item set, which are positive and negative feelings. 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 based on user preferences. The user similarity is generated with 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 above fuzzy relations.

Although the application of fuzzy set techniques in recommender systems has obtained much attention in previous research, to the best of the available knowledge, there has been no research focused on solving fuzziness problems on tree-structured item recommendation. How to use fuzzy set techniques in recommender systems to handle tree-structured items or user profiles, as a consequence, needs to be investigated.

2.1.7 SOCIAL NETWORK-BASED RECOMMENDATION TECHNIQUES

Social networks analysis (SNA) has been used in recommender systems as a result of the dramatic growth of social networking tools in web-based systems in recent years. Social network-based recommendation techniques utilize users' social relationships to make recommendations.

"Trust" is a widely discussed relationship in social network studies. Considering the real world situation in which one's decision to purchase is more likely to be influenced by suggestions from friends than by website advertising, a user's social network may be an important source if it exists in a recommender system. Likewise, due to the inability of standard CF approaches to find sufficient similar neighbours in sparse data sets, users' social relationships are emerging as another improvement facet for recommender systems. Trust represents an intuitive opinion to other users. In a recommender system, the word "trust" is usually known as "how well does Alice trust Bob concerning the

specific product or taste" (Ben-Shimon et al. 2007). It has been shown that there is positive correlation between trust and user similarity in online communities (Ziegler & Lausen 2004). Researchers have conducted a series of studies on integrating trust into recommender systems. A systematic algorithm, TidalTrust, was proposed by Golbeck (2005) to address the trust-based rating prediction problem and has been considered to be effective in the forming process of numeric trust networks in several systems (Golbeck 2006; Golbeck & Hendler 2006). Ben-Shimon et al. (2007) constructed personal social trees for active users by using a breadth-first search algorithm and then computed the distances from active users to others, which can be seen as a reflection of trust, as the final rating prediction weights. Hwang and Chen (2007) analysed the local trust matrix and global trust matrix respectively in a recommender system. Their results indicate that both local trust-awareness and global trust-awareness (also known as reputation) can stimulate increases in recommendation coverage and accuracy. Typically, trust-based approaches are thought to be able to increase recommendation coverage while maintaining accuracy.

Besides trust relationships, researchers have abstracted social networks from social bookmarks (Shiratsuchi, Yoshii & Furukawa 2006), physical context (Woerndl & Groh 2007), online communications (Yan 2008), social tags, "co-author" relationships (Lopes et al. 2010), etc.; and applied them into recommender systems. Shiratsuchi et al. (2006) developed an online information recommender system based on a "co-citation" network drawn from online bookmarks, in which the number of "co-cited" bookmarks of two users is considered as the weight of the connection between them. Woerndl and Groh (2007) considered the entire relevant social context as a vector and integrated it into the rating data to construct a multi-dimensional user-item-context matrix. In the recommender system in (Fan et al. 2008), the metric measuring the relationship degree of two users in a social network is represented by the proportion of the "co-neighbours" in the union set of their neighbours. The work of Tyler and Zhang (2008) indicates that exploiting social context information and trust networks for recommender systems results in good performance, especially for those networks with open domains. Yang et al. (2008) developed an algorithm to find cohesive subgroups to address the neighbour-

finding problem, which can enhance recommendation performance for a large-scale social network. Other social interactions such as online comments, messages and tagging photos are addressed in (Carmagnola, Vernero & Grillo 2009). All these online actions between two users are weighted and combined in an overall evaluation of their social relationship in this system. Ma et al. proposed two systems fused with a probabilistic matrix factorization method and social context/trust information (Ma et al. 2008).

Researchers also conducted several studies on the social networks of recommender systems based only on the user-item rating matrix. Palau et al. (2004) structured social networks to present the collaborative relationships and proposed several measures to explain how collaboration is achieved in the recommendation framework. O'Donovan (2009) presented a trust calculation model from rating data in their trust-based recommendation architecture to make the system more explainable without decreasing prediction accuracy.

2.1.8 CONTEXT AWARENESS-BASED RECOMMENDATION TECHNIQUES

Context awareness-based recommendation techniques utilize the context information, such as time, geometrical information, or the company of other people (friends, family or colleagues for example), to make recommendations. Context in the recommender system field is a multifaceted concept used across various disciplines, with each discipline adopting a certain angle and putting its "stamp" on this concept (Adomavicius & Tuzhilin 2011). With context awareness, the rating function is no longer a two-dimensional (2D) function (R: $User \times Item \rightarrow Rating$) but becomes a multi-dimensional function (R: $User \times Item \times Context \rightarrow Rating$), where *User* and *Item* are the domains of users and items respectively, *Rating* is the domain of ratings, and *Context* specifies the contextual information associated with the application. To incorporate the contextual information in recommender systems, Adomavicius and Tuzhilin (2011) proposed three process steps to make such information computable and valuable: Contextual Pre-Filtering, Contextual Post-Filtering, and Contextual Modelling. By processing the three steps the system can detect the useful and compliable contextual information for making suggestions.

2.1.9 GROUP RECOMMENDATION TECHNIQUES

Group recommender systems (GRS) are proposed to produce a group of users suggestions when group members are unable to gather for face-to-face negotiation, or their preferences are not clear in spite of meeting each other (Jameson & Smyth 2007). GRS have been applied to many domains including movies, music, web pages, events and complex issues such as travel plans.

Many strategies, inspired by social choice theory and decision making procedure, are used for aggregating all the members into a group. Masthoff (2011) summarized eleven strategies including least misery (Gorla et al. 2013), average (Quijano-Sanchez et al. 2013), most pleasure (Quijano-Sánchez et al. 2012) and their adaptations, as the most common in GRS. Other strategies, such as approval voting (Popescu 2013) and sum, are also used in aggregation. Besides the aggregating methods, asynchronous and synchronous communications are also involved in GRS for multi-user support. An asynchronous communication mechanism for users was developed (Jameson, Baldes & Kleinbauer 2004) which allowed users in a group to view (and also copy) other members' choices. McCarthy et al. (2006) implemented a synchronous conversational system to produce ski holiday suggestions for groups. The features predefined in this system, both for resorts and accommodation, can be critiqued by group members. All the members' feedback can be aggregated and recommendations that satisfy the group as a whole are ultimately generated.

2.2 RECOMMENDER SYSTEM APPLICATIONS

To understand the recommender system application developments, the literature on recommender system applications, which either presents a recommender system software or develops a system framework for a specific application, are reviewed in this section. In particular, the recommender systems in domains of e-government, e-business, e-learning and e-library which are highly relevant to this study are selected.

2.2.1 E-GOVERNMENT RECOMMENDER SYSTEMS

Electronic government (e-government) refers to the use of the Internet and other information and communication technologies to support governments in providing improved information and services to citizens and businesses. The rapid growth of e-government has caused information overload, leaving businesses and citizens unable to make effective choices from the range of information to which they are exposed. Increases in this information overload could clearly hamper the effectiveness of e-government services, and difficulties in locating the right information for the right users will increasingly impact on loyalty of users. Recommender systems can overcome this problem and have been adopted in e-government applications (Guo & Lu 2007; Terán et al. 2012; Terán & Meier 2010, 2011).

In this section, the developments and applications of e-government recommender systems will be reviewed, in particular e-government web interface personalization and adaptation and e-government service recommendation, which include government-to-citizen (G2C) and government-to-business (G2B) services.

1) G2C service recommendation

To help citizens to access personalized online services, a conceptual framework of a domain specific personalized e-government service, Pe-Gov, was proposed using the citizen-centric approach (Al-hassan, Lu & Lu 2009). The proposed framework has several distinguishing features including being citizen-centric, building comprehensive user profiles, adopting personalization techniques, using domain ontology and using the user community concept to generate intelligent recommendations. To deliver personalized services to citizens, several recommender system frameworks have been developed. For example, to provide personalized e-government tourism services, a specific framework for developing personalized e-government tourism service systems using a user-centric approach was developed by Al-hassan et al. (2011). Several advanced recommendation techniques, ontology, semantic reasoning and probability techniques are employed in the framework. To provide personalized recommendation of tourist activities, an agent-based recommender system called Turist@ was developed (Batet et al. 2012), which incorporates a mixture of CB and CF recommendation strategies. To support citizens in their access to personalized and adapted services

supplied by Public Administration offices, a multi-agent system was presented by De Meo et al. (De Meo et al. 2007; De Meo, Quattrone & Ursino 2008). The proposed system identifies and suggests the most interesting services for a user by considering both the user's profile and the profile of the device being used. To assist voters to make decisions in the e-election process, a recommender system was proposed (Terán et al. 2012; Terán & Meier 2010), which uses fuzzy clustering methods and provides information about candidates close to voters' preferences.

2) G2B service recommendation

In G2B services, many items from a business perspective are one-time items, such as events, which typically receive ratings only after they have ended. Traditional CF techniques cannot recommend these kinds of items due to the sparse rating data. To handle this problem, Guo and Lu (2007) proposed a new approach which handles an attribute-considered recommendation issue by integrating the semantic similarity techniques with the traditional item-based CF. A recommender system called Smart Trade Exhibition Finder (STEF), which suggests suitable trade exhibitions to businesses, has been developed. To reflect flexibly the graded/uncertain information in the G2B domain, Cornelis et al. (2005) modelled user and item similarities as fuzzy relations. They also proposed a novel hybrid CF-CB approach whose rationale is concisely summed up as "recommending future items if they are similar to past items that similar users have liked". A hybrid fuzzy logic-based recommendation framework was then developed (Cornelis et al. 2007) to improve the trade exhibition recommender system for e-government.

To support government to effectively recommend the proper business partners (e.g., international buyers, agents, distributors, and retailers) to individual businesses (e.g., exporters), a recommender system called BizSeeker (Lu et al. 2010) was developed. Business users can obtain a recommendation list of potential business partners from BizSeeker, as shown in Figure 2-1. A product semantic relevance model was proposed to calculate semantic similarity, and a hybrid semantic recommendation approach combining item-based CF similarity and item-based semantic similarity techniques was then developed. A real-world case study shows that BizSeeker helps to resolve the

sparsity problem and increases recommendation accuracy. To handle linguistic terms in users' interests and the opinions of experts on product relevance, a fuzzy similarity measure and a hybrid fuzzy semantic recommendation (HFSR) approach were proposed (Lu, Shambour & Zhang 2009). These were implemented to solve the fuzzy problems in the BizSeeker recommender system (Lu et al. 2013).

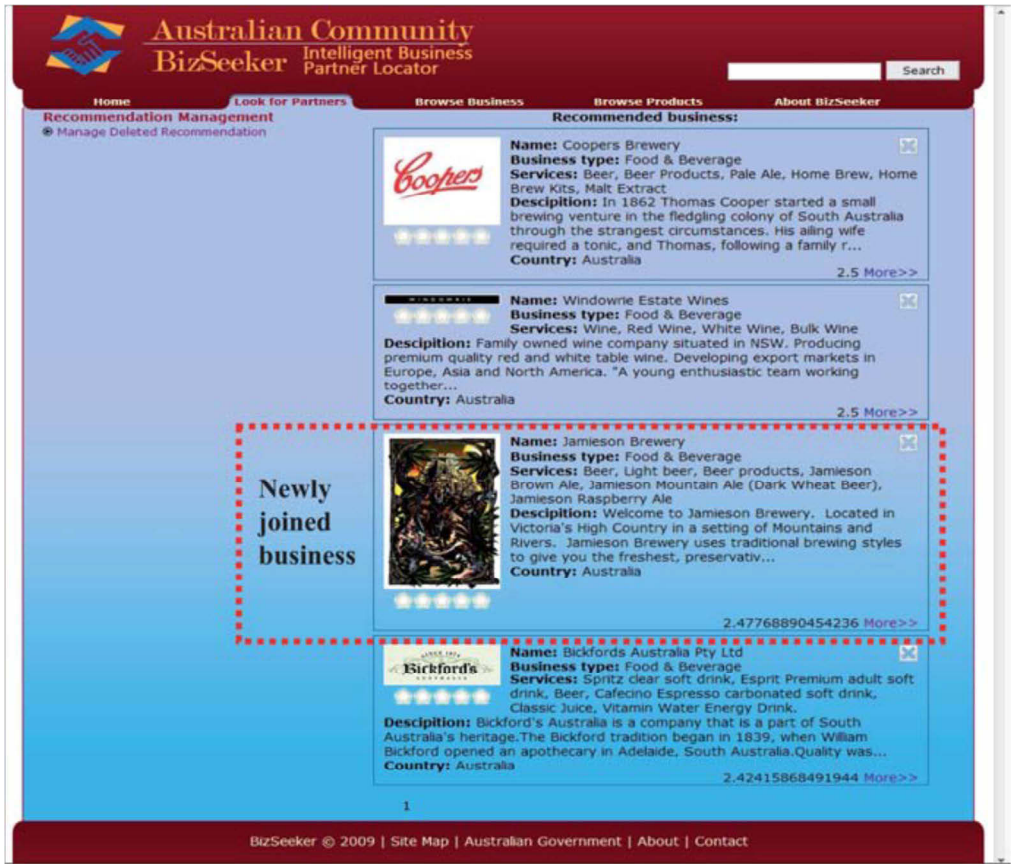


Figure 2-1. The recommendation list of potential business partners generated by BizSeeker (Lu et al. 2010)

To improve similarity accuracy in G2B recommender systems on the BizSeeker platform, the ratio of common users who rated both items to the total number of users who rated each item individually was considered, and an enhanced item-based CF approach was presented (Shambour & Lu 2011a) in the G2B e-government domain. In the business partner selection process, trust or reputation information is crucial and has significant influence on a business user's decision regarding whether or not to do business with other business entities. A hybrid trust-enhanced CF recommendation

(TeCF) approach, which integrates the implicit trust filtering and the enhanced user-based CF approaches, was proposed (Shambour & Lu 2011c) to alleviate the sparsity and cold start user problems and achieve better accuracy. Because a single overall rating of a business quality does not provide the necessary granularity and flexibility, and can bias the recommendation, a multi-criteria CF was used (Shambour & Lu 2010). A personalized hybrid recommender system framework which employs a hybrid trust-based multi-criteria recommendation model was proposed to support exporters seeking business partners in G2B e-services (Shambour & Lu 2010).

In addition to traditional CF and CB techniques, ontology, semantic web, agent, and fuzzy techniques are used in the above-mentioned Pe-Gov, STEF and BizSeeker recommender systems to form a set of hybrid recommendation approaches to improve personalized e-government service performance.

2.2.2 E-BUSINESS RECOMMENDER SYSTEMS

Many recommender systems have been developed for e-business applications. In general, some systems focus on recommendations generated to individual customers, which are business-to-consumer (B2C) systems, while others aim to provide recommendations about products and services to other business users, which are business-to-business (B2B) systems.

1) B2C recommender systems

Many of the largest commerce web sites, such as Amazon and eBay, already use recommender systems to help their customers find products to purchase (Huang, Chung & Chen 2004; Schafer, Konstan & Riedl 2001). In these B2C e-commerce web sites, products can be recommended based on the top overall sellers, customer demographics, or an analysis of the past buying behaviour of the customer as a prediction for future buying behaviour. Some advanced models are also proposed by academic literatures for different criteria of e-shopping environments. For example, KB analyses are usually employed in systems where it is difficult to collect user rating data. The Wasabi Personal Shopper (WPS) (Burke 1999) is a domain-independent database browsing tool designed

for online information access, particularly for electronic product catalogues. WPS is based on a line of academic research called the FindMe system. FindMe is built in several different languages, and uses custom-built ad-hoc databases and KB similarity retrieval. Fuzzy techniques are also employed in CB e-shopping recommender systems. For example, a fuzzy-based recommender system was developed for products made up of different components (Cao & Li 2007). When buying a laptop, for instance, shoppers may consider the individual performance of each component, such as the CPU, motherboard and memory. In this application, the weights of a shopper's needs on each component are collected and the most satisfied candidates are then generated according to a fuzzy similarity measure model. In the e-shopping system proposed in (Bahrainian et al. 2010), a recommendation framework is implemented as part of the electronic customer relationship management (E-CRM) strategy. By incorporating collaborative and non-collaborative filtering methods, the user interface (UI) can alter automatically as the customer profile changes, enabling personalized dynamic recommendations to be provided for users. Furthermore, in some music sharing websites such as the Last.fm system, there are various types of music and user relations in the music social community. To utilize better the rich social information, a hyper graph model is introduced in the music recommendation approach proposed in (Tan et al. 2011) to treat the rich social media information. Certain shopping assistant systems have an interest in explaining the recommendations made to users. For example, when buying expensive goods, buyers expect to be skilfully steered through the options by well-informed sales assistants who are capable of balancing the user's various requirements. In addition, users often need to be educated about the product-space, especially if they are to understand what is available and why certain options are recommended by the sales assistant. To provide an equivalent virtual recommendation explanation such as "why product A is better than B", McCarthy et al. (2004) developed a shopping assistant website called Qwikshop.com on which compound critiques were used as explanations. A set of so-called *critique patterns* is generated by comparing each remaining case to the current recommended case; the relative feature differences make up the critique pattern. The best candidate products, for example those with the highest cost-performance ratio, will be recommended to users. Another issue is purchasing a bundle of items or bundle

promotion. In the systems developed by Garfinkel et al. (2006), the authors extended the one-product-at-a-time search approach used in "shopbot" (a shopping search engine) to consider purchasing plans for a bundle of items. This recommender system leverages bundle-based pricing and promotional deals frequently offered by online merchants to extract substantial savings.

In summary, B2C recommender systems are usually implemented in web-based online purchasing for both digital products (music, movies, etc.) and physical goods (books, DVDs, etc.). From the application perspective, researchers have developed several unique e-shopping systems in which to employ their novel algorithms. These systems provide guidelines for how to implement recommender systems for e-shopping in practice.

2) B2B recommender systems

A framework for a recommender system for B2B e-commerce was designed in (Zhang & Wang 2005), which defines the system's main components and processes. A virtual B2B e-market, Bicycle-Mall, was used to show the effectiveness of the framework. To help catalogue administrators in B2B marketplaces maintain up-to-date product databases, an ontology-based product-recommender system was presented (Lee et al. 2006), in which keyword-based, ontology and Bayesian belief network techniques are used to generate recommendations. To help business users select trusted online auction sellers, a recommender system was designed (Wang & Chiu 2008) in which trading relationships are used to calculate the level of recommendations. To help business users find appropriate B2B e-business apps in the SAP store, a hybrid recommender system that uses KB, CF, and CB sub-recommenders was proposed (Oprea et al. 2013). A novel hybrid weighting scheme incorporating confidence scoring for predictions was presented, so that sub-recommenders contribute to recommendations according to their confidence weight.

Usage History

Month	CallTimetoOptus	CallTimetoOther	Data	MSGtoOptus	MSGtoOther
June, 2010	638	935	50	60	65
May, 2010	654	945	57	75	75
April, 2010	634	938	57	77	25
March, 2010	655	910	58	74	35
February, 2010	645	919	58	47	43
January, 2010	635	914	45	87	42

1 2 3

Average useage in last six months

Calls to Optus	318	Calls to Others	445
TotalTimeToOptus	643	TotalTimeToOther	926
TotalCallValue	1445.95	Message To Optus	70
Message To Other	47	Total Message Value	29.25
Data	54.167	Data Value	832.00
Total Value	2307.20	Actual Fee	862.44

Plan and package upgrade option for you!

New Plan	99 Timeless Cap
New monthly cost	128.99

Customer gets Unlimited call & message and save 733.45 dollars every month

Figure 2-2. Plan and package recommendation for a customer in the Fuzzy-based Telecom Product Recommender System (Zhang et al. 2013)

Customer relationship management is very important for the telecom industry. To support telecom companies in recommending suitable products and services to their business and individual customers, Zhang et al. (2013) designed and implemented a personalized recommendation approach and a software system called the fuzzy-based telecom product recommender system (FTCP-RS). The FTCP-RS can generate the service plan and package recommendations for a customer and can also give recommendation explanations, as shown in Figure 2-2. To deal with sparsity problems and improve prediction accuracy, particularly in handling customer data uncertainty and fully using business knowledge in the recommendation process, the proposed approach integrates item-based CF (IBCF) and user-based CF (UBCF) with fuzzy set techniques and a KB method (business rules). The implemented system has undergone preliminary testing in a telecom company and has been shown to have an excellent performance.

It is found that in B2B recommender system researches, a clear feature is that the KB approaches, such as ontology and semantic techniques, are widely integrated with CF and

other recommendation methods. The main reason for this is that e-businesses have a high need for knowledge to assist their recommendations.

2.2.3 E-LEARNING RECOMMENDER SYSTEMS

Online e-learning systems have become increasingly popular in educational institutions since the early 2000s. Online e-learning systems usually aim to assist learners to choose the courses that interest them and the appropriate learning materials. With more than ten years of accumulated study on this topic, many practicable e-learning systems have been proposed by researchers.

Zaiane (2002) proposed an approach to build a software agent that uses data mining techniques such as association rule mining to construct a model that represents online user behaviours, and used this model to suggest activities or shortcuts. The suggestions generated assist learners to better navigate the online materials by finding relevant resources more quickly using the recommended shortcuts. A personalized e-learning material recommender system (PLRS) was proposed in the work of Lu (2004). Under the framework, a multi-criteria student requirement analysis model is developed to identify a student's requirements; a fuzzy matching method is used to deal with the uncertain criteria values in real life situations. Once a learning material database or a learning activity database is created and a learner's registration information is obtained by the system, the PLRS uses a computational analysis model to identify the leaner's learning requirement and then uses matching rules to generate a recommendation of learning materials (or activities) for the learner. Web usage mining is the process of applying data mining techniques to the discovery of behaviour patterns based on web click-stream data, which provides information to help understand users' preferences. By hybridizing web usage mining with CF and CB techniques, Lu et al. (2004) developed and implemented a subject e-learning recommender system (SRS) that aimed to locate the right subject information for the right students according to their individual interests and needs in subject selection. In the personalized courseware recommender system (PCRS) continuously developed in (Chen, Duh & Liu 2004) and (Chen & Duh 2008), a fuzzy item response theory (FIRT) is proposed to initiatively collect a learner's preferences,

following which the learner provides a fuzzy response as a percentage of their understanding of the learned courseware. The system framework presented in (Chen, Duh & Liu 2004) contains both online and off-line modules. The online modules provide the evaluations of a learner's preference and the matching process between learners and courseware. The off-line module provides a courseware management agent to assess the difficulty level of each course, in support of the matching process. In another e-learning system, CourseAgent, developed by Farzan and Brusilovsky (2006), students are able to provide feedback in implicit and explicit ways. They can directly evaluate courses with respect to the relevance to each career goal as well as the difficulty level of the course. They can also provide implicit feedback when they plan or register for a course. The basic and evident benefit of the system to students is that it offers a course management system that retains the information about courses they have taken and facilitates communication with their advisors. This work is a good attempt at providing social navigation support and community-based recommendations which can generate more benefit and encouragement to use the system. In addition to implicitly mining from web usage or explicitly obtaining recommendations through a response/feedback system, relevant pedagogical rules should also be considered. Pedagogical rules describe pedagogy-oriented relations between the characteristics of learners and characteristics of learning activities (Drachsler, Hummel & Koper 2008b). For example, "recommended learning activities should have a level a little bit above learners' current competence level" (Vygotskiĭ 1978) is accepted as a basic pedagogical rule applied in Moodle e-learning systems (Dougiamas & Taylor 2003; Sevarac, Devedzic & Jovanovic 2012). The corresponding ontologies of learners and learning objectives are discussed in literature. Biletskiy et al. (2009) described a technical solution for a personalized search of learning objects on the web which proposes a comparison of learner (user) profiles and learning object descriptions. This comparison is based not only on the values of attributes of the learner profiles and the attributes of the learning object descriptions, but also on the importance of these attributes for the learner. In the framework, a comparator is proposed to evaluate the "matching score" between learners and learning objectives, based on comparison rules. The CF recommendation approach was adapted to be used in an e-learning context by considering the learners' knowledge levels in (Bobadilla, Serradilla

& Hernando 2009). Attributes of materials are considered in the e-learning material recommendations in (Salehi & Kmalabadi 2012), and CB, CF and some hybrid approaches are used to generate recommendations. To alleviate the stability vs. plasticity problem of technology enhanced learning recommender systems, a recommendation approach that combines a fuzzy collaborative filtering algorithm with a content based one, using learners' preferences and importance of knowledge was proposed in (Maâtallah & Seridi 2012). In order to improve the quality of learning material recommendations, the multi-dimensional attributes of material, rating of learners, and the sequential patterns of the learner's accessed material are considered in (Salehi & Kamalabadi 2013), where a sequential-based recommendation module was developed to discover the latent patterns of accessing materials, and a learner preference tree was introduced to describe the learner profiles. However, to the best of available knowledge, there has been no research focusing on comprehensively solving the fuzzy tree-structured data in e-learning recommendations.

From the above reviews, it is clear that KB pedagogical rules play a more important part in making recommendations in e-learning systems (Shishehchi et al. 2011) than in other recommender systems, because e-learning systems usually lack sufficient historical data sets for CF or CB algorithms. The architecture of an e-learning system usually consists of three parts: 1) using web analysis techniques to collect learners' profiles and identify their personalized demands; 2) collecting the metadata of learning objectives to identify the features; 3) acquiring related pedagogical knowledge to evaluate the matching degree between learners and learning objectives. It should also be mentioned that some advanced techniques, such as fuzzy theories (Chen & Duh 2008; Sevarac, Devedzic & Jovanovic 2012) and social network analysis (Xin, Jamaliding & Okamoto 2009) are also integrated in the matching process to improve system performance.

2.2.4 E-LIBRARY RECOMMENDER SYSTEMS

Digital libraries are collections of digital objects, along with the associated services delivered to user communities (Gonçalves et al. 2004). Recommender systems can be used in digital library applications to help users locate and select information and

knowledge sources (Porcel & Herrera-Viedma 2010). In this section, e-library recommender systems are reviewed.

Fab, part of the Stanford University Digital Library Project, was reported in (Balabanovic & Shoham 1997). It is a hybrid recommender system which combines both the CB and CF recommendation techniques. To provide better personalized e-library services, a system called CYCLADES (http://www.ercim.org/cyclades) was subsequently presented (Renda & Straccia 2005). CYCLADES provides an integrated environment for individual users and group users (communities) in a highly personalized and flexible way. The recommendation algorithms rely on both personalized information organization and users' opinions, and use CB and CF methods separately and in combination. Porcel et al. researched and developed the recommender system to recommend research resources in university digital libraries (UDL) (Porcel & Herrera-Viedma 2010; Porcel, Moreno & Herrera-Viedma 2009). A fuzzy linguistic recommender system was proposed, in which multi-granular fuzzy linguistic modelling (FLM) was used to represent and handle flexible information by means of linguistic labels, and a hybrid recommender system that combines both CB and CF approaches was presented. To reduce the user input effort, users are allowed to nominate their preferences by means of incomplete fuzzy linguistic preference relation in this system. Based on the above researches, Serrano-Guerrero et al. (2011) presented a recommender system which can incorporate Google Wave technology in UDL.

In the e-library recommender systems discussed above, the hybrid recommendation approach which combines CB and CF techniques is used. To represent and handle flexible information of linguistic labels, fuzzy techniques, and in particular multi-granular fuzzy linguistic modelling, are used.

2.2.5 COMPREHENSIVE ANALYSIS AND FINDINGS

The reviewed recommender systems above are summarized. For each application domain, the number of reviewed recommender systems and the recommendation techniques used in the systems are summarized and presented in Table 2-1. The details of

each recommender system reviewed, including the applied recommendation techniques, user types and periods of use, are listed in Table 2-2.

Table 2-1. Summary of recommendation techniques in main application domains

	CB	CF	KB	Hybrid	Computational Intelligence	Social Network	System Number
E-government	1	7	2	6	4		10
E-business	6	4	8	5	6	4	16
E-learning	3	1	9	1	4	1	10
E-library	2	2		3	1		3
Total	12	14	19	15	15	5	39

From the previous research, it can be seen that: 1) most of recommender systems in the four application domains hybridize several recommendation techniques to achieve better performance; 2) knowledge-based recommendation approaches which utilise the underlying semantic knowledge play an important role. That is probably because the items to be recommended in these application domains are relatively complex; 3) fuzzy techniques are widely used to deal with the uncertain information in these applications. When considering the semantic information of items or user profiles in these application domains, one issue is that they are usually presented as tree structures. However, to the best of available knowledge, none of previous recommender systems model them as tree-structure data. To fully utilise the semantic information of tree-structured items and user profiles, this study presents a comprehensive similarity measure on tree-structured data and develops tree similarity based recommendation approaches.

Table 2-2. Summary of the recommender systems developed, the techniques applied and users suited

Systems	Techniques	User type	Period	References
Pe-Gov	KB	Individual	2009	(Al-hassan, Lu & Lu 2009)
Pe-Gov tourism	CF, Hybrid	Individual	2011	(Al-hassan, Lu & Lu 2011)

service system				
Turist@	CB, CF	Individual	2012	(Batet et al. 2012)
A multi-agent e-government system	KB	Individual	2005	(De Meo et al. 2007; De Meo, Quattrone & Ursino 2008)
eElections	Fuzzy clustering	Individual	2010	(Terán et al. 2012; Terán & Meier 2010, 2011)
Smart trade exhibition finder	CF, Hybrid	Business	2007	(Guo & Lu 2007)
A trade exhibition recommender system for e-government	CF, Hybrid, Fuzzy logic	Business	2007	(Cornelis et al. 2005; Cornelis et al. 2007)
BizSeeker	CF, Hybrid	Business	2010	(Lu et al. 2010)
Intelligent business partner locator (IBPL)	CF, Hybrid, Fuzzy sets	Business	2009	(Lu, Shambour & Zhang 2009)
Smart BizSeeker	CF, Hybrid, Fuzzy sets	Business	2013	(Lu et al. 2013)
Wasabi Personal Shopper (WPS)	KB	Individual	1999	(Burke 1999)
A supermarket product RS	CB, CI	Individual	2001	(Lawrence et al. 2001)
Libra: A book RS	CB	Individual	2000	(Mooney & Roy 2000)
An e-book RS	CB	Individual	2012	(Núñez-Valdéz et al. 2012)
An intelligent electronic books RS	KB	Individual	2011	(Crespo et al. 2011)
CAESAR	Social network, Context aware	Individual	2009	(Ramaswamy et al. 2009)
MusicBox	Social tag, CF, Hybrid	Individual	2010	(Nanopoulos et al. 2010)
Consumer electronic products RS	CB, Fuzzy techniques	Individual	2007	(Cao & Li 2007)
VALA	CB, KB, CI	Individual	2010	(Bahrainian et al. 2010)
MRH	Social network, CI, CF, Hybrid	Individual	2011	(Tan et al. 2011)
Qwikshop	KB	Individual	2004	(McCarthy et al. 2004)
Bundle purchases RS	CB, KB, Hybrid	Individual	2006	(Garfinkel et al. 2006)
An ontology-based product RS	KB, Bayesian belief network	Business	2006	(Lee et al. 2006)
Auction seller RS	Social network	Business	2008	(Wang & Chiu 2008)
B2B App RS	CF, KB, Hybrid	Business	2013	(Oprea et al. 2013)
Telecom recommender system	CF, KB, Hybrid, Fuzzy sets	Business	2010-2013	(Zhang et al. 2013)
Fab	CB, CF, Hybrid	Individual	1997	(Balabanovic & Shoham 1997)
CYCLADES	CB, CF, Hybrid	Individual, Group	2005	(Renda & Straccia 2005)
University digital library recommender system	Hybrid, Fuzzy linguistic modeling	Individual	2009-2011	(Porcel & Herrera-Viedma 2010; Porcel, López-Herrera & Herrera-Viedma

				2009; Porcel, Moreno & Herrera-Viedma 2009; Serrano-Guerrero et al. 2011)
An e-learning recommender agent	KB, Rule mining	Individual	2002	(Zaiane 2002)
SRS	CF, CB, Web usage Analysis, Hybrid	Individual	2004	(Lu et al. 2004)
Personalized courseware recommender system (PCRS)	KB, CI, Fuzzy item response theory	Individual	2004	(Chen & Duh 2008; Chen, Duh & Liu 2004)
PLRS	CB, KB	Individual	2004	(Lu 2004)
CourseAgent	KB	Individual	2006	(Farzan & Brusilovsky 2006)
A RS for formal and informal learning	KB	Individual	2008	(Drachsler, Hummel & Koper 2008a)
PSDLO	KB, CB	Individual	2009	(Biletskiy et al. 2009)
ReMashed	KB, CI	Individual	2009	(Drachsler et al. 2010)
A group learning support system	Social network, KB	Individual	2009	(Xin, Jamaliding & Okamoto 2009)
Neuro-fuzzy pedagogical recommender	KB, Fuzzy techniques	Individual	2012	(Sevarac, Devedzic & Jovanovic 2012)

2.3 TREE SIMILARITY MEASURE

Tree structured data are becoming very frequently used nowadays in many applications. They are widely used to represent information in computational biology (Ouangraoua & Ferraro 2009), XML matching (Sanz et al. 2008; Sanz, Pérez & Berlanga 2008), schema matching (Melnik, Garcia-Molina & Rahm 2002), ontology management (Xue et al. 2009), document classification (Lin et al. 2008), e-business applications (Bhavsar, Boley & Yang 2004; Yang et al. 2005), and case-based reasoning (Ricci & Senter 1998). As a tree structure can express the complex relationships between its nodes, it is an essential information representation model in the above applications. It is a key problem to evaluate the similarity between tree structured data in many applications. For example, in case-based reasoning applications, the most similar tree structured cases to the new problem should be retrieved (Ricci & Senter 1998). In e-business applications, complex products and customers' requirements are both represented by tree structured

data. The products most matched to the customer's requirements should be found out (Bhavsar, Boley & Yang 2004). An effective and efficient similarity measure model on tree structured data is vital to these applications.

2.3.1 TREE STRUCTURED DATA

Tree is a commonly used concept in both mathematics and computer science. As a commonly used data structure model in mathematics, a tree is defined as a directed, acyclic, connected graph $T = (V, E)$, in which V is a set of vertices, and E is a set of edges that are represented by ordered pairs of vertices. Any edge (x, y) in E represents a parent-child relationship. Every vertex except one (the root r) has exactly one parent vertex: the root has no parent. (Ouangraoua & Ferraro 2009)

The pure mathematical definition of a tree contains very limited semantics. In real applications, tree structured data are usually extended from the basic mathematical definition to express richer information by adding more features to the basic concept.

To express more information, the vertices of a tree are assigned labels. A mapping from a vertex set V to a label set Σ is introduced (Ricci & Senter 1998; Xue et al. 2009), which makes the tree a labelled tree.

In some situations, such as document representation, the order of vertices is important, so a left-right order over children of each vertex is defined (Lin et al. 2008), which makes the tree an ordered tree. The trees without the left-right order definition over children of each vertex are unordered trees. In more complex situations, some of the vertices are ordered while the others are unordered. To handle the order of vertices flexibly, a semi-ordered tree was defined (Ouangraoua & Ferraro 2009).

To represent more semantic meanings of the nodes, the labels of nodes can be assigned semantic meanings by defining the semantic relationships between labels. For example, the tree can be extended to a concept tree by introducing a concept similarity between labels (Xue et al. 2009).

In more complex situations, such as the e-business environment, the edges of the tree can also be labelled to represent the finer meanings of relationships between parent and child vertices (Yang et al. 2005). The vertices can be assigned weights to represent their importance degrees among their siblings (Yang et al. 2005).

Assigning labels or attributes to vertices or edges of trees and defining the semantic meanings of these labels can make the trees more expressive and enable them to describe more complex objects.

Tree structured data are also used to represent information in different granularities. The nodes in the low level represent the micro information, while the nodes in the high level represent the macro information (De Leeuw & Meijer 2008). For example, in the applications of educational statisticians, students are nested in classes, and classes are nested in schools. And perhaps schools are nested in districts, and so on. In this kind of tree structures, all nodes have values. The leaf nodes contain the information about individuals, and their parent nodes contain the information about groups which these individuals belonged to. The higher level nodes contain the information of larger groups.

It is hard to propose a uniform tree structured data model to represent every kind of hierarchical data objects due to the diversity of applications. A specific tree structured data model needs to be designed for a specific problem. However, some common components, such as the semantic definitions of the attributes, should always be included. In this study, a comprehensive tree structured data model which can assign nodes concept, value and weight will be proposed.

2.3.2 TREE SIMILARITY MEASURE METHODS

Since the similarity measure on tree structured data is essential for many applications, the research into the similarity measure models on tree structured data is attracting considerable attention from many application fields.

The tree edit distance model (Bille 2005; Kailing et al. 2004; Zhang 1993) is the most widely used method to measure the distance between ordered or unordered labelled trees. The model defines several edit operations on trees and the costs of these edit operations.

Three basic kinds of edit operations on trees are relabel, delete and insert (Bille 2005; Zhang 1993):

Relabel a node $v \in V$ means changing the label of the node v.

Delete a node v means making the children of v become the children of the parent of v and then removing v. For ordered trees, the children are inserted in the place of v as a subsequence in the left-to-right order of the children of the parent of v. For unordered trees, the operation works on a subset instead of a sub-sequence.

Insert is the complement of delete. For ordered trees, insert a node v as a child of $w \in V$ making v the parent of a consecutive sub-sequence of the children of w. For unordered trees, v is made the parent of a subset of the children of w.

The tree edit distance model measures the distance between two trees by the minimum cost of the edit operation sequences that convert one tree into another (Bille 2005). 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 labelled trees (Zhang 1993). The distance between two trees can then be defined as the minimum cost of the edit distance mapping. It has been shown that the tree edit distance is equal to the minimum cost of the edit distance mapping. The set of allowed edit operations and their related cost definitions in different applications may be different (Yang, Kalnis & Tung 2005). For example, the conceptual similarity measure between labels was introduced in the cost of edit operations to compare concept trees of ontology in (Xue et al. 2009). In some tree edit distance algorithms, the edit distance mapping must satisfy specific structural constraints (Lu, Su & Tang 2001; Richter 1997; Zhang 1996).

Another kind of tree similarity measure is based on a maximum common sub-tree (MCS) or sub-tree isomorphism between two trees (Akutsu & Halldórsson 2000). This method uses the size of MCS between two trees, or metrics defined by MCS as the similarity measure. In (Torsello, Hidovic & Pelillo 2004), four novel distance measures for attributed trees based on the notion of a maximum similarity sub-tree isomorphism were proposed. In (Lin et al. 2008), the number of all common embedded sub-trees

between two trees was used as the measure of similarity. To evaluate the similarity between tree structured cases, Ricci and Senter (1998) developed a similarity measure which takes into account both the tree structures and node labels' semantics. In their method, a sub-tree isomorphism with the minimum semantic distance was constructed and the minimum semantic distance was taken as the similarity measure.

These methods mentioned above mostly deal with node-labelled trees. In (Bhavsar, Boley & Yang 2004), node-labelled, arc-labelled and arc-weighted trees were used as product/service descriptions to represent the hierarchical relationship between the attributes. To compare these trees, a recursive algorithm was designed which performs a top-down traversal of trees and the bottom-up computation of similarity. It is worth while noting that their trees had to conform to the same standard schema, i.e. the trees should have the same structure and use the same labels, though some sub-trees were allowed to be missing (Yang et al. 2005).

From the previous research, it can be seen that:

1) To compare two trees, tree mappings such as tree edit distance mapping or isomorphism usually need to be constructed;

2) The similarity or distance measures are defined on the tree mappings;

3) The tree mapping with the maximum similarity is finally constructed and the corresponding similarity measure is taken as the similarity between two trees;

4) When comparing two trees and constructing their tree mappings, both the structural and semantic aspects are taken into consideration.

2.3.3 STRUCTURAL CONSTRAINTS AND SEMANTIC SIMILARITY ON TREE-STRUCTURED DATA MEASURE

From the previous tree similarity methods, it can be seen that trees are compared from both the structural aspect and semantic aspect. In this sub-section, the structural constraints and semantic similarity issues in the tree similarity measures are reviewed.

2.3.3.1 STRUCTURAL CONSTRAINTS

In real applications, the tree structures can express their semantic concepts. Consequently, some kinds of structural constraints must be satisfied in particular situations. In this sub-section, four structural constraints, which are isomorphic mapping, edit distance mapping, constrained edit distance mapping and less constrained edit distance mapping, are summarized.

1) Isomorphic mapping

The formal definition of isomorphic mapping is defined as follows (Ricci & Senter 1998):

Let $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ be two trees.

A mapping $f: V_1 \rightarrow V_2$ is a morphism if and only if $(u, v) \in E_1 \Rightarrow (f(u), f(v)) \in E_2$.

T_1 and T_2 are isomorphic if and only if there exists a morphism $f: V_1 \rightarrow V_2$ satisfying that f is bijection and the inverse of f is also a morphism.

Intuitively speaking, the isomorphic mapping requires that the mapped sub-trees be identical from the point of view of the structure.

2) Edit distance mapping

The edit distance mapping is derived from the tree edit operations (Bille 2005). According to the features of tree edit operations discussed above, the edit distance mapping (Zhang 1996) is formally described as follows:

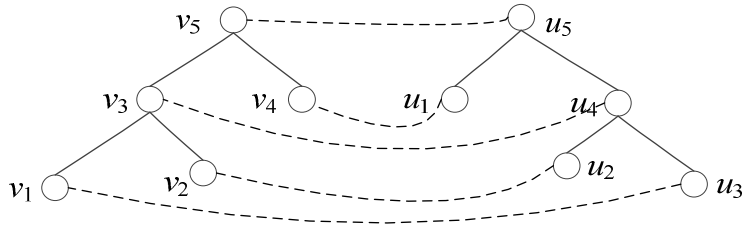
Let V_1 and V_2 be node sets of tree-structured data T_1 and T_2, respectively. A mapping $M \subseteq V_1 \times V_2$ is an edit distance tree mapping if it satisfies the following conditions:

For any pair (v_1, u_1), $(v_2, u_2) \in M$, where $v_1, v_2 \in V_1$, $u_1, u_2 \in V_2$,

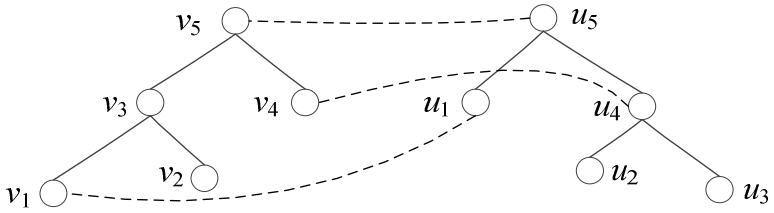
- $v_1 = v_2$ if and only if $u_1 = u_2$;

- v_1 is an ancestor of v_2 if and only if u_1 is an ancestor of u_2;
- v_1 is to the left of v_2 if and only if u_1 is to the left of u_2.

The first constraint condition of the edit distance mapping requires that the mapping is a one-to-one mapping. The second constraint condition requires that the ancestor and descendant relations must be preserved in the mapping. The third constraint condition is for ordered trees, which requires that the sibling order must be preserved in the mapping. The edit distance mapping loosens the mapping restrictions of the isomorphic mapping, and brings more flexibility.



(a)



(b)

Figure 2-3. (a) an isomorphic mapping and (b) an edit distance mapping

Figure 2-3 shows two examples of tree mappings. The trees in the figure are unordered labelled trees. The dashed lines connect corresponding nodes in the mappings. Mapping (a) is an isomorphic mapping since the two mapped trees have the same structure. It is also an edit distance mapping. Mapping (b) is an edit distance mapping since it satisfies the constraint conditions of the edit distance mapping. However, mapping (b) is not an isomorphic mapping, because (v_1, u_1) and (v_5, u_5) are both in the

mapping and u_1 and u_5 are connected by an edge, but v_1 and v_5 are not connected by an edge.

3) Constrained edit distance mapping

The constrained edit distance mapping adds more constraint conditions to the edit distance mapping. The intuitive idea of this kind of mapping is that two separate sub-trees of T_1 should be mapped to two separate sub-trees of T_2 (Zhang 1993). Zhang (1993, 1996) proposed constrained edit distance. Richter (1997) introduced the structural respecting edit distance, which is equivalent to constrained edit distance. To change an edit distance mapping into a constrained edit distance mapping, an additional constraint condition should be added:

For any pair (v_1, u_1), (v_2, u_2), $(v_3, u_3) \in M$, where $v_1, v_2, v_3 \in V_1$, $u_1, u_2, u_3 \in V_2$. Let $lca(v_1, v_2)$ be the lowest common ancestor of v_1 and v_2. $lca(v_1, v_2)$ is not an ancestor or a descendant of v_3 if and only if $lca(u_1, u_2)$ is not an ancestor or a descendant of u_3.

The condition can also be described equivalently as:

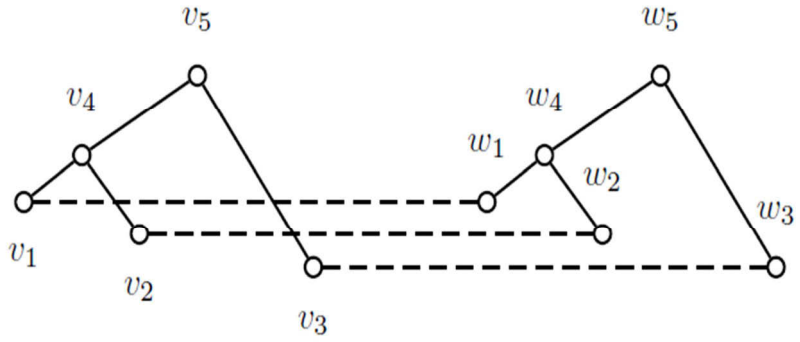
For any pair (v_1, u_1), (v_2, u_2), $(v_3, u_3) \in M$, where $v_1, v_2, v_3 \in V_1$, $u_1, u_2, u_3 \in V_2$. Let $lca(v_1, v_2)$ be the lowest common ancestor of v_1 and v_2. $lca(v_1, v_2) = lca(v_1, v_3)$ if and only if $lca(u_1, u_2) = lca(u_1, u_3)$.

4) Less constrained edit distance mapping

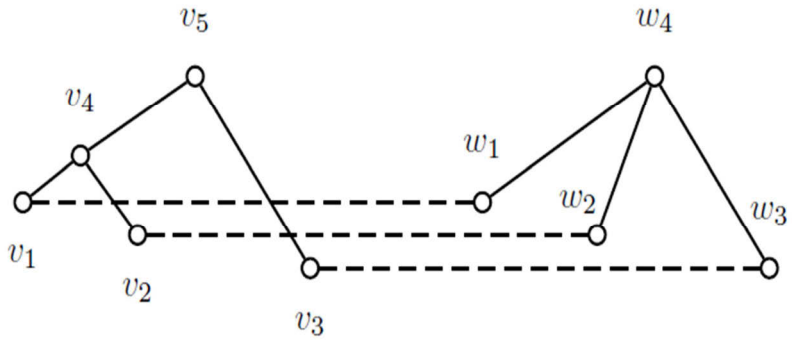
Lu et al. (2001) introduced less constrained edit distance, which relaxes the constrained mapping. Intuitively speaking, the less constrained edit distance mapping allows one sub-tree of T_1 to be mapped to more than one sub-tree of T_2, which can construct a 1-to-many mapping (Lu, Su & Tang 2001). By contrast, the constrained edit distance mapping can only construct a 1-to-1 mapping. The edit distance mapping *M* is a less constrained edit distance mapping if *M* satisfies the following constraint:

For all (v_1, u_1), (v_2, u_2), $(v_3, u_3) \in M$ such that none of v_1, v_2, and v_3 is an ancestor of the others, $depth(lca(v_1, v_2)) \geq depth(lca(v_1, v_3))$ and also $lca(v_1, v_3) =$

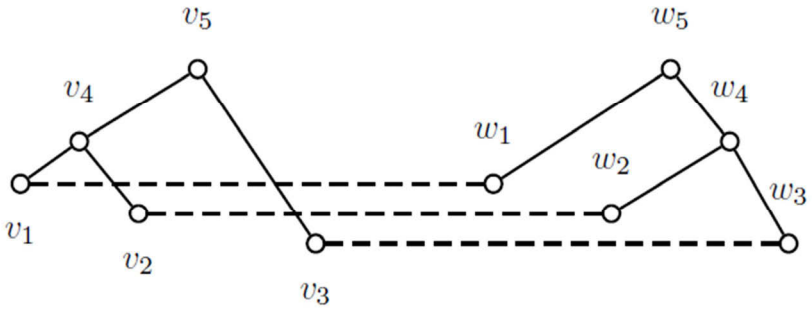
$lca(v_2, v_3)$ if and only if $depth(lca(u_1, u_2)) \geq depth(lca(u_1, u_3))$ and $lca(u_1, u_3) = lca(u_2, u_3)$.



(a)



(b)



(c)

Figure 2-4. Three examples of different mappings: (a) is a constrained edit distance mapping; (b) is a less constrained edit distance mapping; (c) is an edit distance mapping which is neither constrained nor less-constrained. (Bille 2005)

Figure 2-4 shows some examples of tree edit distance mappings under different structural constraints. The dashed lines connect corresponding nodes in the mappings. Mapping (a) is a constrained mapping since $lca(v_1, v_2) \neq lca(v_1, v_3)$ and $lca(w_1, w_2) \neq$

$lca(w_1, w_3)$. Consequently, it is also a less constrained mapping. Mapping (b) is not a constrained mapping since $lca(v_1, v_2) = v_4 \neq lca(v_1, v_3) = v_5$, while $lca(w_1, w_2) = w_4 = lca(w_1, w_3)$. However, the mapping is less constrained since $depth\big(lca(v_1, v_2)\big) > depth(lca(v_1, v_3))$, $lca(v_1, v_3) = lca(v_2, v_3)$, $lca(w_1, w_2) = lca(w_1, w_3)$, and $lca(w_1, w_3) = lca(w_2, w_3)$. Mapping (c) is not constrained since $lca(v_1, v_3) = v_5 = lca(v_2, v_3)$, while $lca(w_1, w_3) = w_5 \neq lca(w_2, w_3) = w_4$. (c) is not a less constrained mapping either since $depth\big(lca(v_1, v_2)\big) > depth(lca(v_1, v_3))$ and $lca(v_1, v_3) = lca(v_2, v_3)$, while $lca(w_1, w_3) \neq lca(w_2, w_3)$.

As the recursive structure of tree structured data, dynamic programming algorithms are used to construct the tree mapping with the maximum similarity when comparing two trees. For unordered trees, the isomorphic mapping can be constructed with polynomial time and space complexity algorithms (Ricci & Senter 1998). Many researches have been done to develop algorithms to calculate the tree edit distance. For ordered trees, the ordered tree edit distance can be solved by algorithms with polynomial time and space complexity (Chen 2001; Klein 1998; Tai 1979; Zhang & Shasha 1989). For unordered trees, it has been shown that the tree edit distance problem is NP-complete (Zhang, Statman & Shasha 1992). However, considering the constraints, the constrained edit distance mapping for unordered trees can be constructed with polynomial time and space complexity algorithms (Zhang 1996). The less constrained edit distance mapping for ordered trees also has a polynomial time and space complexity algorithm (Lu, Su & Tang 2001).

In real applications, different tree-structured data may have different structures. Meanwhile, disjoint sub-trees in the tree-structured data usually represent different semantic concepts. Taking both the flexibility and restriction of different kinds of tree mappings into consideration, the constrained edit distance mapping is constructed to compare two tree-structured data in this study.

2.3.3.2 SEMANTIC SIMILARITY ON TREE STRUCTURED DATA MEASURE

Tree structured data objects can represent complex semantic concepts. Besides comparing their structures, the semantic meanings and concepts of nodes should also be compared when evaluating the similarity between two trees.

The similarity between labels was introduced for labelled trees in (Ricci & Senter 1998). When constructing the sub-tree isomorphism, it was required that only nodes whose labels' similarity greater than zero can be mapped. The requirement means only nodes with similar concepts can be matched. A conceptual similarity measure was defined for concept lexicons of different trees in (Xue et al. 2009). When transforming one tree into the other by use of transformation operations, the conceptual similarity between nodes' labels determines the transformation operations and the cost of them.

To summarise, when evaluating the similarity on tree structured data objects, only conceptual similar nodes can be matched.

In this study, to comprehensively evaluate the similarity between tree-structured data, besides the tree structures, node concepts, values and weights are all considered. A comprehensive similarity measure on tree-structured data will be developed.

2.4 SUMMARY

In summary, there are the following research gaps in previous research:

1) Previous recommender systems have not shown the ability to handle tree-structured items or user profiles;

2) Previous tree-structured data model and relevant similarity measure are not sufficiently expressive to cover all the features of the tree-structured items or user profiles in recommender systems;

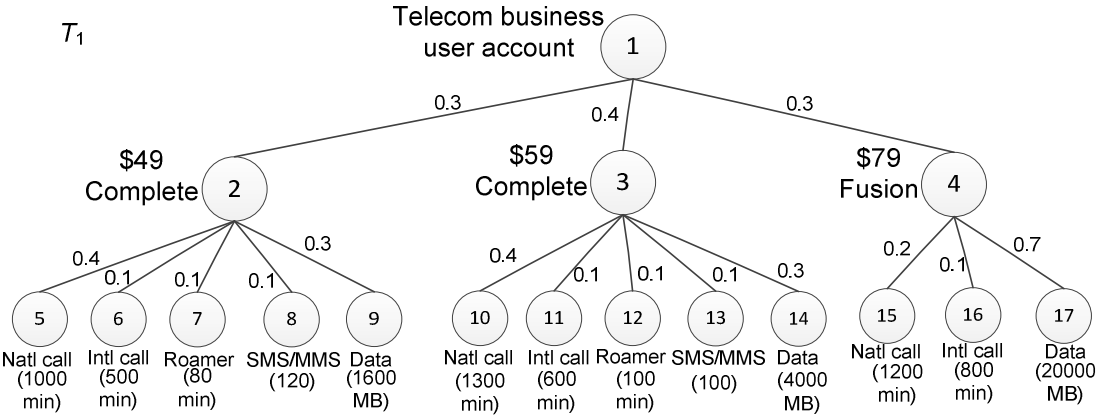
3) Even though fuzzy set techniques are widely applied to deal with uncertainty data in recommender systems, no research has been found which has focused on solving fuzziness problems on tree-structured item recommendation.

To solve these problems, this study proposes and develops a set of recommendation approaches for the recommender system applications which have tree-structured items or user profiles. To evaluate the semantic similarity between tree-structured items or users effectively, a comprehensive similarity measure method and relevant algorithms on tree-structured data will be developed. To deal with the uncertainty issues, a fuzzy tree-structured preference model will be proposed and a fuzzy preference tree-based recommendation approach will be developed. The developed recommendation approaches will also be applied to develop a business partner recommender system and an e-learning recommender system.

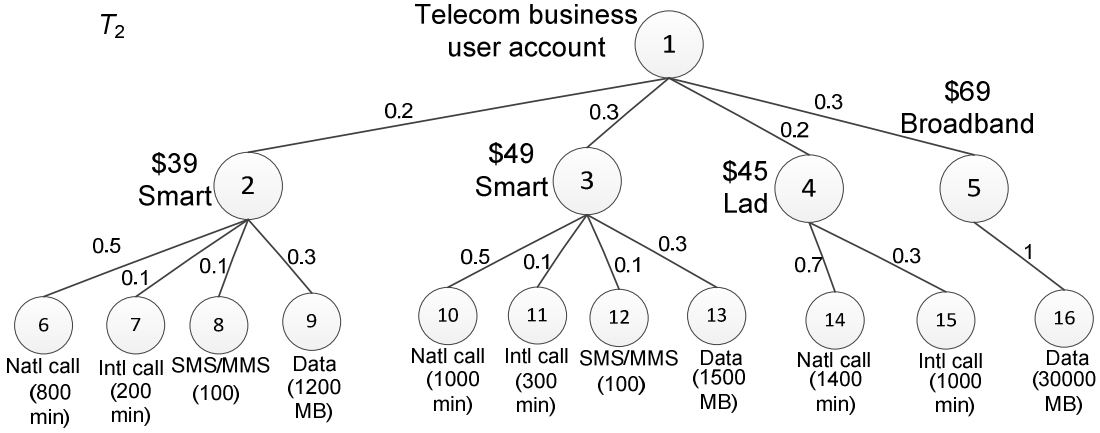
CHAPTER 3
SIMILARITY MEASURE ON TREE-STRUCTURED DATA

3.1 INTRODUCTION

Tree structured data are becoming ubiquitous nowadays in many application fields, such as XML matching (Sanz et al. 2008; Sanz, Pérez & Berlanga 2008), document classification (Lin et al. 2008), ontology management (Born et al. 2008; Shvaiko & Euzenat 2013; Solskinnsbakk et al. 2012; Xue et al. 2009), computational biology (Ouangraoua & Ferraro 2009), case-based reasoning (Ricci & Senter 1998) and e-business applications (Bhavsar, Boley & Yang 2004; Yang et al. 2005). The tree similarity measure has attracted a considerable amount of attention in these fields because evaluating the data similarity is usually an essential part of these applications. For example, in case-based reasoning, a key is to search for the most similar existing cases to a new problem. As ontology usage becomes more prevalent in e-business decision support systems, it is essential to assess the similarity between tree structured ontologies (Zhao et al. 2012). However, the tree-structured data models and similarity measures in the above applications are designed for their specific application requirements. In many situations, such as e-business and e-learning recommender systems, item and user profile data present tree structures with more complicated features. To make proper recommendations in these systems, it is essential to find the similar users or similar items. Therefore, a more comprehensive tree-structured data model and tree similarity measure are needed to evaluate the semantic similarity between items or users in these recommender systems.



(a)



(b)

Figure 3-1. (a) (b) Two examples of tree-structured business data

The tree-structured business data are first introduced. Figure 3-1 shows two examples of business data, which are the usage of two telecom business user accounts. The first user has three plans, two mobile service plans ("$49 Complete" and "$59 Complete") and a fusion service plan ("$79 Fusion"), while the second user has four plans, two mobile service plans ("$39 Smart" and "$49 Smart"), one landline service plan ("$45 Lad") and a broadband service plan ("$69 Broadband"). Each plan provides some services and the relevant usage of the user is recorded. Clearly, they are viewed as tree structures. From the tree structures, it can be seen that each node in the tree represents some specific meaning. Some nodes, such as those represent the usage, are assigned values. The

numbers in the bracket under the nodes in Figure 3-1 are their values. Every node in the tree is assigned a weight to reflect its importance degree to its siblings.

From the two examples, the following features of tree-structured data can be summarised:

1) Every tree node is associated with a concept;

2) All concepts represented by tree nodes construct a hierarchical structure. Nodes at different depths represent concepts with different abstraction levels. The child nodes can be viewed as a refinement of the concept expressed by their parent node;

3) A node can be assigned values to express the quantitative features of the relevant node;

4) Every node is assigned a "weight" to represent its importance degree to its siblings.

In addition, the difference between the two telecom service packages in Figure 3-1 shows that the tree structures, node weights, terms and values of different tree-structured business data can all be different. To evaluate the similarity between these tree-structured business data, all the information should be considered. However, to the best of available knowledge, none of the existing tree-structured data models can handle or even express all the features of the business data considered in this study.

In this chapter, a comprehensive tree-structured data model is defined; a similarity measure model, which considers all the information on tree structures, nodes' concepts, weights and values, is developed, and a set of similarity measurement algorithms are also developed, to deal with tree-structured data in recommender systems.

3.2 TREE-STRUCTURED DATA MODEL

The tree-structured data model is based on the basic mathematical definition of trees, which is given as follows.

Definition 3-1 (Cayley 1857)**.** 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 $root(T)$, so that for any node $v \in V$, there is a path in T from $root(T)$ to node v.

The definition only defines the hierarchical relations between the nodes. In real applications, the definition is usually extended to represent practical objects. In this research, a tree-structured data model which assigns tree nodes concepts, values and weights is defined.

Definition 3-2. A tree-structured data model is a tree, in which the following features are added to the tree nodes:

1) Node concept: A set of attributes $A = \{a_1, a_2, \dots, a_n\}$ to represent the concept of a node are introduced. A range set $D = \{d_1, d_2, \dots, d_n\}$ is defined accordingly. For each attribute a_i, an attribute assignment function $a_i: V \to d_i$ is defined so that each node can be assigned values for attribute a_i.

2) Based on the attributes in A, a node concept similarity $sc(\cdot)$ between two nodes is defined as $sc: D \times D \to [0, 1]$.

3) Node value types: A set of node value types $VTP = \{tp_1, tp_2, \dots, tp_m\}$ are defined. For each $tp_i \in VTP$, a node value range $vr_i = vr(tp_i)$ is defined, and the value similarity of type tp_i is defined as $sv_{tp_i}: vr_i \times vr_i \to [0, 1]$. A value type assignment function is defined as $vtp: V \to \{\Lambda \cup VTP\}$, where Λ represents the *null* set. Therefore, a tree node can be assigned one value type and assigned a value. It is noteworthy that not all nodes are assigned values.

4) Node value: if a node *n* has been assigned a value type tp_i, a value from the value range $vr_i = vr(tp_i)$ can be assigned to the node, which is denoted as $v(n)$.

5) A weight function $w: V \to [0, 1]$ is defined to assign a weight to each node to represent its importance degree to its siblings.

Two issues should be noticed about the above Definition 3-2. First, for the node concept, a tree node can be assigned several attributes to represent its concept. For example, a node representing a product can be assigned two attributes, a product name and a category, to express the semantic meanings. Second, for node values, a node can only be assigned one type of value. Only the values of the same type can be compared. For a parent node, its children's values can be aggregated only if they are the same value type.

Now a tree-structured data example is given to show how the proposed tree-structured data model is used to model business data.

Take the business data in Figure 3-1 as the example. According to the tree-structured data model definition (Definition 3-2), the features in the definition are described as follows.

1) Node concept

For each tree node, the node concept is defined by a *label* attribute. It is defined as a *label* function $label: V \rightarrow L$, where L is a label term set. In this example, the label terms consist of telecom plan names (such as "$49 Complete" and "$45 Lad"), plan family names (such as "Smart", "Complete", and "Fusion"), service item names (such as "Natl call", "Intl call" and "Data") and so on. These label terms are pre-defined by domain experts.

2) Node concept similarity

A conceptual similarity measure within the label set is defined as $sc_L: L \times L \rightarrow [0, 1]$. The node concept similarity between two nodes $u, v \in V$ is then defined as $sc(u, v) = sc_L(label(u), label(v))$. Evidently, the node concept similarity meets the commutative law, i.e. $sc(u, v) = sc(v, u)$.

The conceptual similarity between two labels can be given by domain experts or inferred from the domain ontology that describes the relations between the labels. As an example, the conceptual similarity between the labels of T_1 and T_2 in Figure 3-1 is

defined, as shown in Table 3-1. Based on Table 3-1, the conceptual similarity between any two labels l_1 and l_2 is calculated as follows: (1) if $l_1 = l_2$, $sc_L(l_1, l_2) = 1$; (2) if $l_1 \neq l_2$ and they are in Table 3-1, $sc_L(l_1, l_2)$ is defined as the corresponding value in the table; (3) if $l_1 \neq l_2$ and they are not in Table 3-1, $sc_L(l_1, l_2) = 0$.

Table 3-1. The conceptual similarity between node labels

Label l_1	Label l_2	Conceptual similarity $sc_L(l_1, l_2)$
\$49 Complete	\$49 Smart	0.75
\$49 Complete	\$39 Smart	0.7
\$59 Complete	\$49 Smart	0.7
\$59 Complete	\$39 Smart	0.6
\$79 Fusion	\$45 Lad	0.5
\$79 Fusion	\$69 Broadband	0.6

3) Node value types

Four value types, which are "voice usage", "mobile data usage", "broadband data usage" and "SMS/MMS amount" are defined. Each value type is associated with a value range and a value similarity measure. Every leaf node in the trees, which represents the usage of a service item, is assigned a value type. The nodes, which are labelled with "Natl call", "Intl call" or "Roamer", are assigned "voice usage" type. The nodes labelled with "Data" for the mobile services are assigned "mobile data usage" type, and the nodes labelled with "Data" for broadband services are assigned "broadband data usage". The nodes labelled with "SMS/MMS" are assigned "SMS/MMS amount" type.

4) Node value

The leaf nodes in a tree, which represent the usage of service items, are assigned values from the relevant value range. In the example, the node values are shown as the numbers in the bracket under the nodes.

5) Node weight

In the example, the number beside an edge represents the weight of the child.

3.3 SIMILARITY MEASURE METHOD ON TREE-STRUCTURED DATA

Different tree-structured data have different tree structures, node concepts, values and weights. To evaluate the similarity between tree-structured data comprehensively, all these kinds of information should be compared. Because a tree-structured data represents specific concepts and values, the similarity between two tree-structured data is evaluated from both the conceptual and value aspects. The conceptual similarity and value similarity between two tree-structured data are defined respectively, and the final similarity measure between them is assessed as the weighted sum of their conceptual and value similarities.

It should be noted that in contrasting application scenarios, the requirements for similarity measures are different. For example, when computing the semantic similarity between two tree-structured items, the weights of both trees should be considered. Another example is matching a tree-structured item to a user's tree-structured request, where the weights of the user's request tree should be mainly weighted. Therefore, the similarity measure should consider the two situations respectively. In the first example's situation, the similarity measure is symmetric, denoted as sim_{sym}. In the second example's situation, the similarity measure is asymmetric, denoted as sim_{asym}. The subscript can be omitted if there is no confusion.

Given a tree, it is usually convenient to use a numbering to refer to the nodes of the tree. In the following sections, trees and nodes are represented with the following symbols.

Let $t[i]$ be the ith node of the tree T in the given numbering, $T[i]$ be the sub-tree rooted at $t[i]$ and $F[i]$ be the unordered forest obtained by deleting $t[i]$ from $T[i]$. Let $t[i_1]$, $t[i_2]$, ..., $t[i_{n_i}]$ represent the children of $t[i]$.

Given two trees $T_1[i]$ and $T_2[j]$ to be compared, let their conceptual similarity be denoted as $sc_T(T_1[i], T_2[j])$, and their value similarity be denoted as $sv_T(T_1[i], T_2[j])$. The final comprehensive similarity measure between $T_1[i]$ and $T_2[j]$ is defined as:

$$sim(T_1[i], T_2[j]) = \alpha_1 \cdot sc_T(T_1[i], T_2[j]) + \alpha_2 \cdot sv_T(T_1[i], T_2[j]), \quad (3\text{-}1)$$

where $\alpha_1 + \alpha_2 = 1$. The definitions and computation methods of conceptual similarity $sc_T(T_1[i], T_2[j])$ and the value similarity $sv_T(T_1[i], T_2[j])$ are described in the following sections.

3.3.1 CONCEPTUAL SIMILARITY BETWEEN TWO TREE STRUCTURED DATA

The concept of a tree is derived from the concepts of node attributes and tree structures. Both aspects should be considered when computing the conceptual similarity between two trees. To compare two trees, their concept corresponding parts should be identified first; the corresponding node pairs are then compared separately and finally aggregated into one value. During the conceptual similarity computation, the matched node pairs are recorded, which construct the maximum conceptual similarity tree mapping. The matched node pairs should satisfy both the structural and conceptual constraints. For the conceptual constraints, a node concept similarity measure $sc(\cdot)$ is pre-defined based on the node attributes. Only conceptually similar nodes are to be matched. For the structural constraints, it is required that disjoint sub-trees should be mapped to disjoint sub-trees, because the tree structures represent specific semantic relations between nodes in a sub-tree. To satisfy the requirements of the structural constraints, the constrained edit distance mapping (Bille 2005; Zhang 1996) is introduced and applied.

The constrained edit distance tree mapping is formally defined as follows:

Definition 3-3: Constrained edit distance tree mapping. Let V_1 and V_2 be node sets of tree-structured data T_1 and T_2, respectively. A mapping $M \subseteq V_1 \times V_2$ is a constrained edit distance tree mapping if it satisfies the following conditions:

For any pair (v_1, u_1), (v_2, u_2), $(v_3, u_3) \in M$, where $v_1, v_2, v_3 \in V_1$, $u_1, u_2, u_3 \in V_2$,

1) $v_1 = v_2$ if and only if $u_1 = u_2$

2) v_1 is an ancestor of v_2 if and only if u_1 is an ancestor of u_2;

3) Let $lca(v_1, v_2)$ be the lowest common ancestor of v_1 and v_2. $lca(v_1, v_2) = lca(v_1, v_3)$ if and only if $lca(u_1, u_2) = lca(u_1, u_3)$.

In the above Definition 3-3, the first condition ensures that the mapping is a one-to-one mapping. Condition 2 ensures that the ancestor-descendant relations are preserved in the mapping. Condition 3 ensures that disjoint sub-trees are mapped to disjoint sub-trees.

Given two trees $T_1[i]$ and $T_2[j]$, their conceptual similarity $sc_T(T_1[i], T_2[j])$ is computed as follows considering the constrained edit distance tree mapping constraint.

According to the matching situations of their roots $t_1[i]$ and $t_2[j]$, three cases are considered: ($C1$) $t_1[i]$ is matched to $t_2[j]$'s child node; ($C2$) $t_2[j]$ is matched to $t_1[i]$'s child node; ($C3$) $t_1[i]$ and $t_2[j]$ are matched. The conceptual similarities in the three cases are denoted as $sc_{T_{C1}}(T_1[i], T_2[j])$, $sc_{T_{C2}}(T_1[i], T_2[j])$, and $sc_{T_{C3}}(T_1[i], T_2[j])$, respectively, and $sc_T(T_1[i], T_2[j])$ is computed as:

$$sc_T(T_1[i], T_2[j]) = \max\{sc_{TC1}(T_1[i], T_2[j]), sc_{TC2}(T_1[i], T_2[j]), sc_{TC3}(T_1[i], T_2[j])\}. \quad (3\text{-}2)$$

The matched node pairs in the case with the maximum conceptual similarity value are finally chosen to construct the maximum conceptual similarity tree mapping.

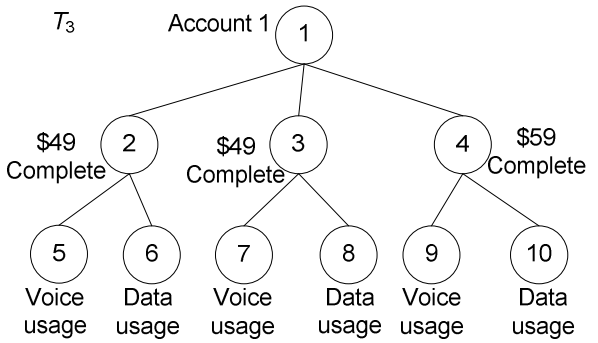
In Case $C1$, $t_1[i]$ is matched to $t_2[j]$'s child node. In this case, the concept level of $t_1[i]$ is lower than that of $t_2[j]$. $T_1[i]$ is mapped to one sub-tree of $T_2[j]$ which has maximum conceptual similarity with $T_1[i]$. $sc_{T_{C1}}(T_1[i], T_2[j])$, is computed as:

$$sc_{TC1}(T_1[i], T_2[j]) = \max_{1 \le t \le n_j} \{w_t \cdot sc_T(T_1[i], T_2[j_t])\}, \quad (3\text{-}3)$$

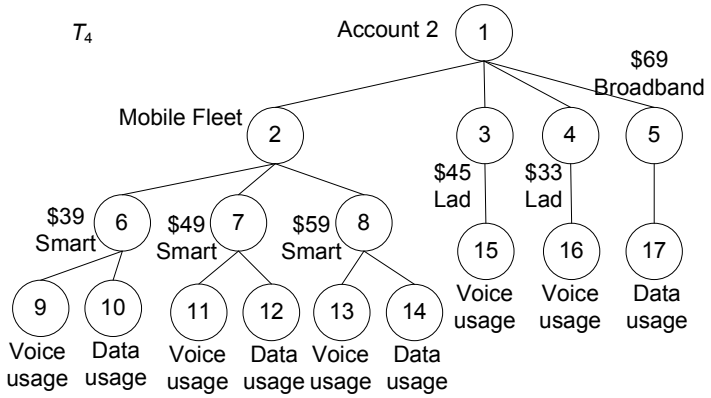
where w_t is the weight of the matched node pair. If the measure is a symmetric measure, both of the corresponding nodes' weights should be considered, $w_t = (1 + w(t_2[j_t]))/2$.

If the measure is an asymmetric measure, only the first node's weight is considered, $w_t = 1$.

For example, Figure 3-2 shows the usage record structures of two business users in the telecom industry. The first customer as shown with T_3 has only mobile services, while the second customer as shown with T_4 has mobile, landline and broadband services. When comparing the two tree-structured data in Figure 3-2, node 1 in T_3 is probably matched to node 2 in T_4.



(a)



(b)

Figure 3-2. (a) (b) Two examples of tree-structured business data

Case *C*2 is similar to *C*1. The conceptual similarity between $T_1[i]$ and $T_2[j]$ is calculated as:

$$sc_{TC2}(T_1[i], T_2[j]) = \max_{1 \le t \le n_i} \{w_t \cdot sc_T(T_1[i_t], T_2[j])\}, \quad (3\text{-}4)$$

where w_t is the weight of the matching node pair. If the measure is a symmetric measure, both of the corresponding nodes' weights should be considered, $w_t = (w(t_2[j_t]) + 1)/2$. If the measure is an asymmetric measure, only the first node's weight is considered, $w_t = w(t_2[j_t])$.

In Case *C*3, $t_1[i]$ and $t_2[j]$ are matched. If the node concept similarity between $t_1[i]$ and $t_2[j]$, $sc(t_1[i], t_2[j]) = 0$, $t_1[i]$ and $t_2[j]$ are not similar, so they cannot be matched according to the conceptual constraint, and $sc_{T_{C3}}(T_1[i], T_2[j]) = 0$. If $sc(t_1[i], t_2[j]) > 0$, $t_1[i]$ and $t_2[j]$ are recorded as a matched node pair, and $sc_{T_{C3}}(T_1[i], T_2[j])$ is calculated. The conceptual similarity between $T_1[i]$ and $T_2[j]$ is calculated as:

$$sc_{T_{C3}}(T_1[i], T_2[j]) =$$
$$\begin{cases} 0, & sc(t_1[i], t_2[j]) = 0 \\ sc(t_1[i], t_2[j]), & sc(t_1[i], t_2[j]) > 0, F_1[i] = \emptyset, F_2[j] = \emptyset \\ \alpha \cdot sc(t_1[i], t_2[j]) & \\ +(1-\alpha) \cdot \sum_{t=1}^{n_j} w_{j_t} \cdot sc_T(T_1[i], T_2[j_t]), & sc(t_1[i], t_2[j]) > 0, F_1[i] = \emptyset, F_2[j] \neq \emptyset \\ \alpha \cdot sc(t_1[i], t_2[j]) & \\ +(1-\alpha) \cdot \sum_{t=1}^{n_i} w_{i_t} \cdot sc_T(T_1[i_t], T_2[j]), & sc(t_1[i], t_2[j]) > 0, F_1[i] \neq \emptyset, F_2[j] = \emptyset \\ \alpha \cdot sc(t_1[i], t_2[j]) & \\ +(1-\alpha) \cdot sc_F(F_1[i], F_2[j]), & sc(t_1[i], t_2[j]) > 0, F_1[i] \neq \emptyset, F_2[j] \neq \emptyset \end{cases}, \quad (3\text{-}5)$$

where w_{j_t} and w_{i_t} are the weights of $t_2[j_t]$ and $t_1[i_t]$ respectively, and α is the influence factor of the parent node. Five situations are listed in Formula (3-5). When $sc(t_1[i], t_2[j]) = 0$, then $sc_{T_{C3}}(T_1[i], T_2[j]) = 0$, which means the conceptual similarity between $T_1[i]$ and $T_2[j]$ is 0 and these two sub-trees are not matched. When $sc(t_1[i], t_2[j]) > 0$, four other situations are dealt with according to the condition of whether $t_1[i]$ and $t_2[j]$ are leaves. In the second situation, $t_1[i]$ and $t_2[j]$ are both leaves, and their conceptual similarity is equivalent to their node conceptual similarity. In the third and fourth situations, 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 the last situation, both $t_1[i]$ and $t_2[j]$ have children. Their children construct two forests $F_1[i] =$

$\{T_1[i_1], T_1[i_2], \dots, T_1[i_{n_i}]\}$ and $F_2[j] = \{T_2[j_1], T_2[j_2], \dots, T_2[j_{n_j}]\}$, which are compared with the forest similarity measure $sc_F(F_1[i], F_2[j])$.

Take the two tree-structured data in Figure 3-1 as examples. Node 12 in T_1 and node 11 in T_2 are both leaves, the conceptual similarity between $T_1[12]$ and $T_2[11]$ is therefore calculated as the node conceptual similarity between $t_1[12]$ and $t_2[11]$, i.e. $sc_{TC3}(T_1[12], T_2[11]) = sc(t_1[12], t_2[11]) = sc_L(\text{"}Roamer\text{"}, \text{"}Intl\ call\text{"})$. Node 2 in T_1 and node 3 in T_2 are both inner nodes, the conceptual similarity between $T_1[2]$ and $T_2[3]$ is therefore calculated by aggregating the node conceptual similarity between $t_1[2]$ and $t_2[3]$ and the conceptual similarity between their children, i.e. $sc_{TC3}(T_1[2], T_2[3]) = \alpha \cdot sc_L(\text{"\$49 Complete"}, \text{"\$49 Smart"}) + (1-\alpha) \cdot sc_F(F_1[2], F_2[3])$.

To calculate the conceptual similarity between two forests $F_1[i]$ and $F_2[j]$, $sc_F(F_1[i], F_2[j])$, the conceptual corresponding sub-trees are first identified based on both their concepts and structures, and are then compared separately. Finally, these local similarities are weight aggregated. To identify the conceptual corresponding node pairs, a bipartite graph $G_{ij} = (V_{1,i} \cup V_{2,j}, E)$ is constructed, in which $V_{1,i} = \{t_1[i_1], t_1[i_2], \dots, t_1[i_{n_i}]\}$, $V_{2,j} = \{t_2[j_1], t_2[j_2], \dots, t_2[j_{n_j}]\}$. For any $t_1[i_p] \in V_{1,i}$ and $t_2[j_q] \in V_{2,j}$, a weight is assigned to edge $(t_1[i_p], t_2[j_q])$ as $w_{p,q} = sc_T(T_1[i_p], T_2[j_q])$. A maximum weighted bipartite matching of G_{ij}, denoted as $MWBM_{ij}$, needs to be constructed. Solving the bipartite graph matching problem in a brute force manner by enumerating all permutations and selecting the one that maximize the objective function leads to an exponential complexity which is unreasonable (Riesen & Bunke 2009). However, there exists polynomial-time algorithms (Fankhauser, Riesen & Bunke 2011), such as Hungarian (Burkard, Dell'Amico & Martello 2009; Kuhn 1955), Munkres (Munkres 1957) and Volgenant Jonker (Jonker & Volgenant 1987). Hungarian is one of the best-known and historically most important combinatorial algorithm (Jungnickel 2008), which is selected in this study. By constructing $MWBM_{ij}$, the matched corresponding node pairs are identified and recorded. The conceptual similarity between $F_1[i]$ and $F_2[j]$ is calculated as

$$sc_F(F_1[i],F_2[j]) = \sum_{(t_1[i_p],t_2[j_q])\in MWBM_{ij}} w_{i_p,j_q} \cdot sc_T(T_1[i_p],T_2[j_q]), \quad (3\text{-}6)$$

where w_{i_p,j_q} is the weight of the matched node pair $t_1[i_p]$ and $t_2[j_q]$. If the measure is a symmetric measure, $w_{i_p,j_q} = (w(t_1[i_p]) + w(t_2[j_q]))/2$. If the measure is an asymmetric measure, $w_{i_p,j_q} = w(t_1[i_p])$. In Formula (3-6), the maximum weighted bipartite matching $MWBM_{ij}$ identifies the most conceptual corresponding node pairs amongst $t_1[i]$ and $t_2[j]$'s children. The contribution of their children can therefore be fully considered when evaluating the conceptual similarity between $T_1[i]$ and $T_2[j]$.

For example, $t_1[1]$ and $t_2[1]$ both have children in Figure 3-1. To compute $sc_F(F_1[1],F_2[1])$, a bipartite graph $G_{1,1}$ is constructed as shown in Figure 3-3 (a). Let $\alpha = 0.5$, the conceptual similarities between the sub-trees of $t_1[1]$ and $t_2[1]$ are calculated, which are set as the weights of edges in Figure 3-3 (a). The maximum weighted bipartite matching of $G_{1,1}$ is then constructed, which is shown in Figure 3-3 (b). Then, the conceptual similarity between $T_1[1]$ and $T_2[1]$ is calculated by $sc_{T_{C3}}(T_1[1],T_2[1]) = \alpha \cdot sc(t_1[1],t_2[1]) + (1-\alpha) \cdot (((0.3+0.2)/2) \cdot sc_T(T_1[2],T_2[2]) + ((0.4+0.3)/2) \cdot sc_T(T_1[3],T_2[3]) + ((0.3+0.3)/2) \cdot sc_T(T_1[4],T_2[5])) = 0.856$.

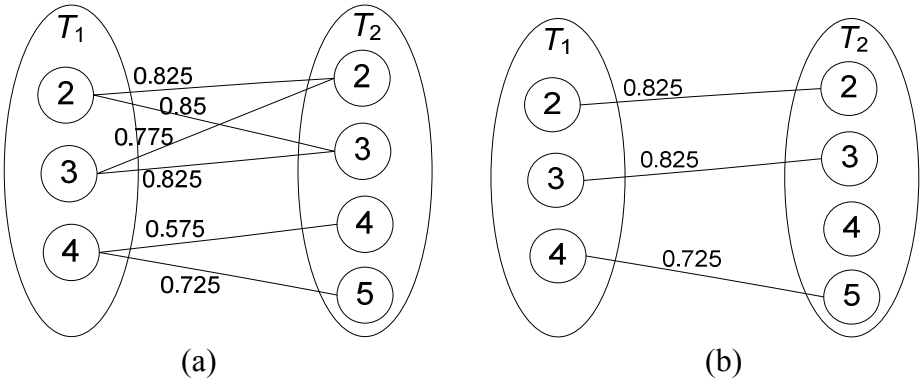


Figure 3-3. (a) a bipartite graph G_{ij} and (b) its maximum weighted bipartite matching

During the computation process of the conceptual similarity between two trees $T_1[i]$ and $T_2[j]$, the matched conceptual corresponding node pairs are recorded, which construct the maximum conceptual similarity tree mapping between $T_1[i]$ and $T_2[j]$, denoted as M_{S12}.

For example, during computing the conceptual similarity between T_1 and T_2 in Figure 3-1, their maximum conceptual similarity tree mapping M_{S12} is constructed, which is illustrated in Figure 3-4. The matched nodes in the maximum conceptual similarity tree mapping in Figure 3-4 are connected by dashed lines.

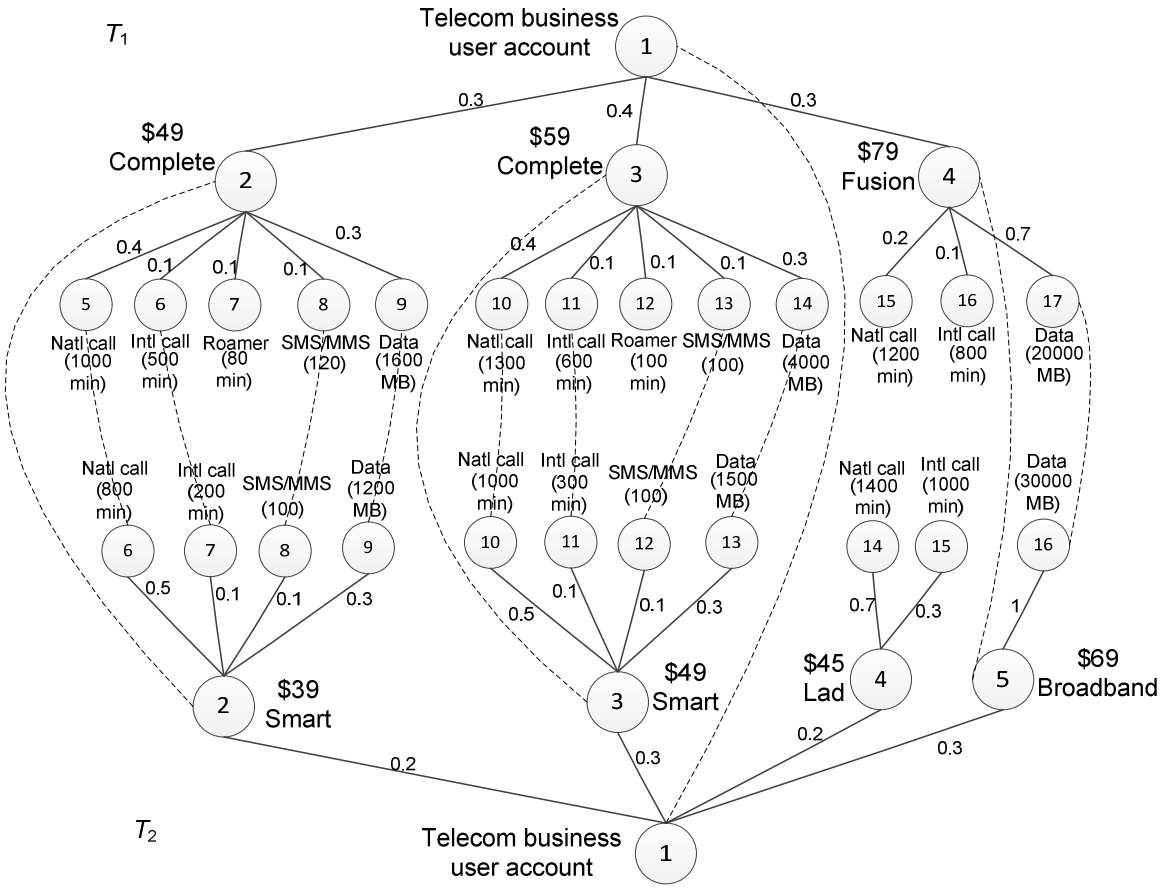


Figure 3-4. The maximum conceptual similarity tree mapping between tree-structured data T_1 and T_2 in Figure 3-1

3.3.2 VALUE SIMILARITY BETWEEN TWO TREE STRUCTURED DATA

For a specific tree-structured object, some nodes are assigned values to express the quantitative features about the nodes. Because nodes in a branch represent a common concept at different levels, only one node for each branch of a tree is assigned a value. Apart from the concepts, the values of two trees should also be compared to comprehensively evaluate their similarity.

Let $v(t[i])$ represent the value of node $t[i]$, $v(t[i]) = null$ if $t[i]$ is not assigned a value. Let $v(T[i])$ represent the aggregated value of tree $T[i]$. If $v(t[i]) \neq null$ or $t[i]$ is a leaf node, $v(T[i]) = v(t[i])$; otherwise, if the value types of the $t[i]$'s children are the same, $v(T[i]) = \sum_{t=1}^{n_i} v(T[i_t])$, else $v(T[i]) = null$. Given two trees $T_1[i]$ and $T_2[j]$, a maximum conceptual similarity tree mapping M_{S12} has been constructed. The following two issues must be considered when computing the value similarity between two trees:

1) The value similarity between two sub-trees can be calculated only if the two sub-trees are matched and they are assigned valid values. Otherwise, their value similarity is zero;

2) Because the nodes in lower levels represent more specific concepts, the value similarity should be evaluated by comparing the values of levels that are as low as possible.

To deal with the node matching and node values conveniently, a matching tree is constructed by extracting the matched node pairs in M_{S12}. The matching tree between two trees is defined as follows.

Definition 3-4. Let $T_1[i]$ and $T_2[j]$ be two tree-structured data. The matching tree between $T_1[i]$ and $T_2[j]$ is denoted as T_{M12}. Each node in T_{M12} contains a matched node pair $(t_1[x], t_2[y])$, where $t_1[x]$ and $t_2[y]$ are two nodes in $T_1[i]$ and $T_2[j]$ that are matched.

T_{M12} can be constructed by pre-order traversing $T_1[i]$, selecting the nodes that are in M_{S12}, and adding the relevant matched node pairs into T_{M12} by preserving the ancestor descendant relations of $T_1[i]$. During the construction of T_{M12}, the node values of the matched sub-trees are checked. If the nodes of a sub-tree in $T_1[i]$ or $T_2[j]$ are not assigned values, the matched node pairs in the sub-tree and its matched counterparts will not be added into T_{M12}. As a result, if T_{M12} can be constructed, the matched two nodes contained in each leaf node in T_{M12} both have valid values, either assigned initially or computed by aggregating values of its children.

For example, based on the maximum conceptual similarity mapping between T_1 and T_2, which is shown in Figure 3-4, the matching tree between T_1 and T_2 can be constructed, as shown in Figure 3-5. The number pairs beside the nodes in Figure 3-5 represent the matched node numbers in T_1 and T_2 respectively.

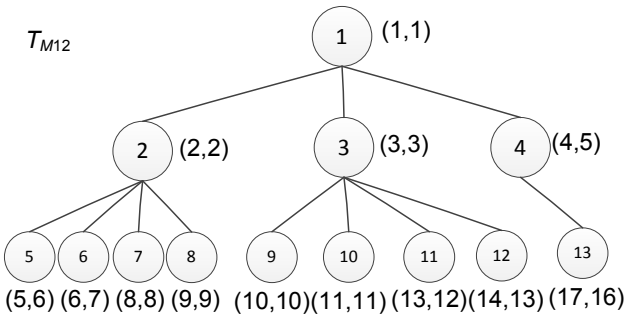


Figure 3-5. The matching tree between trees T_1 and T_2 in Figure 3-1

Suppose there is a numbering in T_{M12}. The kth node is represented as $t_{M12}[k]$, which contains a matching pair ($t_1[x_k]$, $t_2[y_k]$). $t_{M12}[k]$'s children are represented as $t_{M12}[k_1]$, …, $t_{M12}[k_{n_k}]$. For two matched nodes $t_1[x_k]$ and $t_2[y_k]$, the value similarity between their sub-trees $T_1[x_k]$ and $T_2[y_k]$, $sv_T(T_1[x_k], T_2[y_k])$, is calculated by $sv_M(T_{M12}[k])$ as follows.

$$sv_M(T_M[k]) = \begin{cases} 0, & T_M[k] = \emptyset \\ sv_{tp_k}\big(v(T_1[x_k]), v(T_2[y_k])\big), & T_M[k] \neq \emptyset, F_M[k] = \emptyset, \\ \sum_{s=1}^{n_k} w_{k_s} \cdot sv_T(T_M[k_s]), & T_M[k] \neq \emptyset, F_M[k] \neq \emptyset \end{cases} \quad (3\text{-}7)$$

where $sv_{tp_k}(\cdot)$ is the value similarity measure of the value type of $t_1[x_k]$ and $t_2[y_k]$, w_{k_s} is the weight of the matched node pair $t_1[x_{k_s}]$ and $t_2[y_{k_s}]$. If the measure is a symmetric measure, $w_{k_s} = (w^*(t_1[x_{k_s}]) + w^*(t_2[y_{k_s}]))/2$. If the measure is an asymmetric measure, $w_{k_s} = w^*(t_1[x_{k_s}])$. $w^*(t_1[x_{k_s}])$ is the continued product of the weights of nodes in the branch from $t_1[x_{k_s}]$ to $t_1[x_k]$'s child, and $w^*(t_2[y_{k_s}])$ is the continued product of the weights of nodes in the branch from $t_2[y_{k_s}]$ to $t_2[y_k]$'s child. There are three cases listed in the formula. In the first case, $T_M[k]$ is empty, which means that there are no matched sub-trees or the matched sub-trees do not have comparable values. Thus, the value similarity in this case is 0. In the second case, $t_{M12}[k]$ is a leaf node. The

matched nodes $t_1[x_k]$ and $t_2[y_k]$ have no matched descendant nodes, or the matched descendant nodes are not assigned comparable values. The values of $t_1[x_k]$ and $t_2[y_k]$ are compared directly in this case. In the third case, $t_{M12}[k]$ has children. The value similarity between $T_1[x_k]$ and $T_2[y_k]$ is calculated by aggregating their matched sub-trees' value similarities.

To show the computation of the $sv_M(T_{M12}[k])$, nodes $t_{M12}[5]$ and $t_{M12}[2]$ in the matching tree T_{M12} in Figure 3-5 are taken as examples. The node $t_{M12}[5]$ is a leaf node and it contains the matched node pair ($t_1[5]$, $t_2[6]$). Both $t_1[5]$ and $t_2[6]$ are assigned values of the "voice usage" type, and $v(t_1[5]) = 1000$, $v(t_2[6]) = 800$. According to the discussion above about the calculation of the value of a sub-tree, since $t_1[5]$ and $t_2[6]$ are leaf nodes in T_1 and T_2 respectively, $v(T_1[5]) = v(t_1[5]) = 1000$, $v(T_2[6]) = v(t_2[6]) = 800$. Thus, $sv_M(T_{M12}[5]) = sv_{voiceusage}(1000, 800)$. The value similarity measure of the "voice usage" type $sv_{voiceusage}(\cdot)$ will be discussed in detail in the following sections.

The node $t_{M12}[2]$ is an inner node which contains the matched node pair ($t_1[2]$, $t_2[2]$), and it has four children $t_{M12}[5]$, $t_{M12}[6]$, $t_{M12}[7]$ and $t_{M12}[8]$. Thus, $sv_M(T_{M12}[2]) = \sum_{p=5}^{8} w_p \cdot sv_M(T_{M12}[p])$. w_p is the weight of the matched node pair $t_1[x_p]$ and $t_2[y_p]$ contained in $t_{M12}[p]$, $p = 5, 6, 7, 8$. In this example, the similarity measure is a symmetric measure. Therefore, $w_p = (w^*(t_1[x_p]) + w^*(t_2[y_p]))/2$. For example, $w_5 = (w^*(t_1[5]) + w^*(t_2[6]))/2$. $w^*(t_1[5])$ is calculated as the continued product of the weights of nodes in the branch from $t_1[5]$ to $t_1[2]$'s child, and $w^*(t_1[5]) = w(t_1[5]) = 0.4$. Similarly, $w^*(t_2[6]) = w(t_2[6]) = 0.5$. w_5 is then calculated as 0.45.

Based on Formula (3-7), the value similarity between $T_1[i]$ and $T_2[j]$, $sv_T(T_1[i], T_2[j])$, is calculated as

$$sv_T(T_1[i], T_2[j]) = w_r \cdot sv_M(T_{M12}), \tag{3-8}$$

where w_r is the weight of the matching node pair ($t_1[x_r]$, $t_2[y_r]$) contained in the root of T_{M12}. If the similarity measure is a symmetric measure, $w_r = (w^*(t_1[x_r]) +$

$w^*(t_2[y_r]))/2$. If the measure is an asymmetric measure, $w_r = w^*(t_1[x_r])$. $w^*(t_1[x_r])$ is the continued product of the weights of nodes in the branch from $t_1[x_r]$ to the root of $T_1[i]$, and $w^*(t_2[y_r])$ is the continued product of the weights of nodes in the branch from $t_2[y_r]$ to the root of $T_2[j]$.

For example, given T_1 and T_2 illustrated in Figure 3-1 and their matching tree T_{M12} illustrated in Figure 3-5, the value similarity between T_1 and T_2 is calculated by $sv_T(T_1, T_2) = w_r \cdot sv_M(T_{M12}[1])$. Because the roots of T_1 and T_2 are contained in $t_{M12}[1]$, the weight w_r is calculated as 1.

After the conceptual similarity $sc_T(T_1[i], T_2[j])$ and value similarity $sv_T(T_1[i], T_2[j])$ are computed, the final similarity measure between $T_1[i]$ and $T_2[j]$ is calculated by Formula (3-1).

3.3.3 SIMILARITY MEASUREMENT ALGORITHMS

Based on the similarity measure model on tree-structured data and the formulas (3-1) to (3-8) proposed above, a set of algorithms to compute the similarity between two tree-structured data are designed and presented in this section. The entire computation process is described by the flowchart as shown in Figure 3-6.

It can be seen from Figure 3-6 that $sc_T(T_1[i], T_2[j])$ is firstly computed by calling $sc_T(T_1[i], T_2[j], M)$, where *M* is a node mapping set. All the conceptual corresponding node pairs identified during computing $sc_T(T_1[i], T_2[j])$ are recorded in *M*, which constitute the maximum conceptual similarity tree mapping between $T_1[i]$ and $T_2[j]$. To deal with the node matching and node values conveniently, the matched node pairs in *M* are extracted to construct the matching tree between $T_1[i]$ and $T_2[j]$, T_{M12}, by calling $constructMatchingTree(T_1[i], M, null)$. $sv_T(T_1[i], T_2[j])$ is then computed based on T_{M12}. Finally, the similarity between $T_1[i]$ and $T_2[j]$ is returned by aggregating their conceptual and value similarities. The algorithm is illustrated by Algorithm 3-1 as follows.

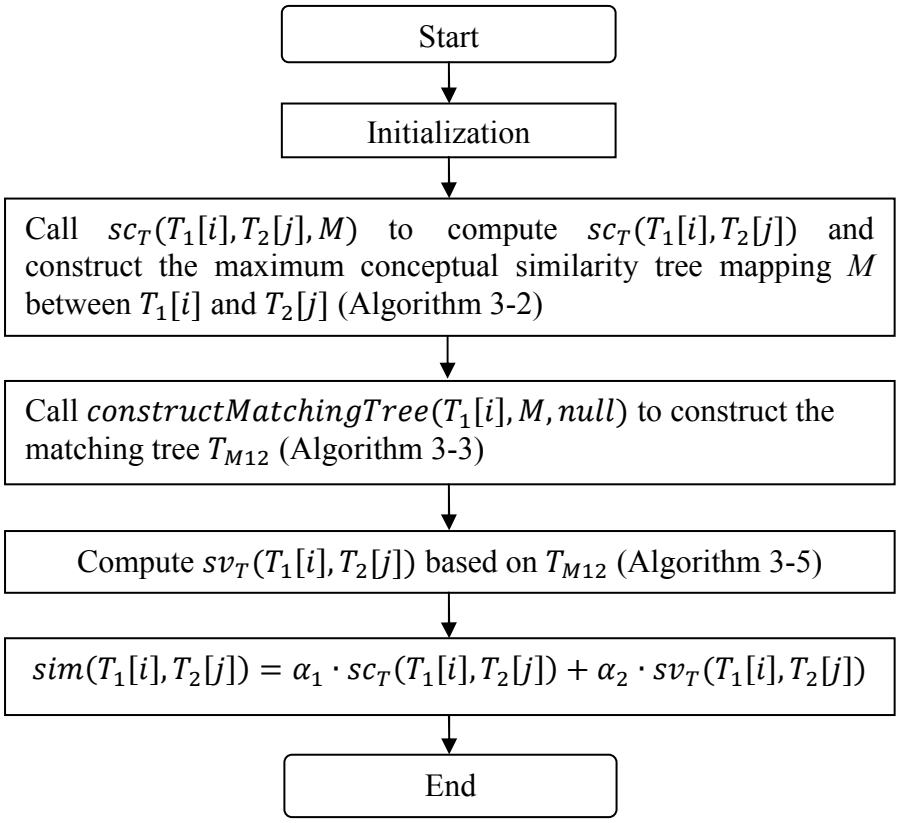


Figure 3-6. Flowchart to compute the similarity between two tree-structured data

Algorithm 3-1. Similarity measure algorithm for tree-structured data

$similarity(T_1[i], T_2[j])$

input: two tree structured data $T_1[i]$ and $T_2[j]$

output: the similarity between $T_1[i]$ and $T_2[j]$

1 new mapping set M

2 $sc_T \leftarrow sc_T(T_1[i], T_2[j], M)$

3 $T_{M12} \leftarrow constructMatchingTree(T_1[i], M, null)$

4 $sv_T \leftarrow w_r \cdot sv_M(T_{M12})$

5 return $\alpha_1 \cdot sc_T + \alpha_2 \cdot sv_T$

6 END

The Algorithm 3-1 contains the following sub-algorithms: the conceptual similarity computation algorithm, the matching tree construction algorithm and the value similarity computation algorithm, which are described as follows.

3.3.3.1 CONCEPTUAL SIMILARITY COMPUTATION ALGORITHM

According to the conceptual similarity measure model and the formulas (3-2) to (3-6) proposed in Section 3.3.1, the algorithm to compute the conceptual similarity between two trees is designed, which is illustrated by Algorithm 3-2. The algorithm has three inputs: the two trees $T_1[i]$ and $T_2[j]$ to be compared 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 $T_1[i]$ and $T_2[j]$.

Algorithm 3-2. Conceptual similarity computation algorithm

$sc_T(T_1[i], T_2[j], M)$

input: two trees $T_1[i]$, $T_2[j]$ and the mapping set *M*

output: the conceptual similarity between $T_1[i]$ and $T_2[j]$

1 if $sc(t_1[i], t_2[j]) = 0$

2 $sc_{T1} \leftarrow 0$

3 else

4 mapping set $M_1 \leftarrow \{(t_1[i], t_2[j])\}$

5 if $F_1[i] = \emptyset$, $F_2[j] = \emptyset$

6 $sc_{T1} \leftarrow sc(t_1[i], t_2[j])$

7 else if $F_1[i] = \emptyset$, $F_2[j] \neq \emptyset$

8 $sc_{T1} \leftarrow \alpha \cdot sc(t_1[i], t_2[j]) + (1-\alpha) \cdot \sum_{t=1}^{n_j} w_{j_t} \cdot sc_T(T_1[i], T_2[j_t], \emptyset)$

9 else if $F_1[i] \neq \emptyset$, $F_2[j] = \emptyset$

10 $sc_{T1} \leftarrow \alpha \cdot sc(t_1[i], t_2[j]) + (1-\alpha) \cdot \sum_{t=1}^{n_i} w_{i_t} \cdot sc_T(T_1[i_t], T_2[j], \emptyset)$

11 else if $F_1[i] \neq \emptyset$, $F_2[j] \neq \emptyset$

12 $V_i \leftarrow \{t_1[i_1], t_1[i_2], \dots, t_1[i_{n_i}]\}$

13 $V_j \leftarrow \{t_2[j_1], t_2[j_2], \ldots, t_2\left[j_{n_j}\right]\}$

14 for s=1 to n_i

15 for t=1 to n_j

16 new mapping set $M_{s,t}$

17 $ew_{s,t} \leftarrow sc_T(T_1[i_s], T_2[j_t], M_{s,t})$

18 $m \leftarrow ComputeMatching(V_i \cup V_j, ew)$

19 for each $(t_1[i_s], t_2[j_t]) \in m$,

20 $M_1 \leftarrow M_1 \cup M_{s,t}$

21 $sc_{T1} \leftarrow \alpha \cdot sc(t_1[i], t_2[j])$

$$+(1-\alpha) \cdot \sum_{(t_1[i_s], t_2[j_t]) \in m} w_{s,t} \cdot ew_{s,t}$$

22 $sc_{T2} \leftarrow 0$, mapping set $M_2 \leftarrow \emptyset$

23 for t=1 to n_j

24 new mapping set $M_{i,t}$

25 $sc_t \leftarrow w_t \cdot sc_T(T_1[i], T_2[j_t], M_{i,t})$

26 if $sc_{T2} < sc_t$

27 $sc_{T2} \leftarrow sc_t$, $M_2 \leftarrow M_{i,t}$

28 $sc_{T3} \leftarrow 0$, mapping set $M_3 \leftarrow \emptyset$

29 for t=1 to n_i

30 new mapping set $M_{t,j}$

31 $sc_t \leftarrow w_t \cdot sc_T(T_1[i_t], T_2[j], M_{t,j})$

32 if $sc_{T3} < sc_t$

33 $sc_{T3} \leftarrow sc_t$, $M_3 \leftarrow M_{t,j}$

34 for p=1, 2, 3

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

36 $M \leftarrow M \cup M_p$

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

38 END

In Algorithm 3-2, a recursive process follows from the conceptual similarity definition. Lines 1-21 deal with the case where the roots of two trees are matched. According to Formula (3-5), five situations are processed. Lines 1 and 2 deal with the situation when $sc(t_1[i], t_2[j]) = 0$. If $sc(t_1[i], t_2[j]) > 0$, according to the condition of whether $t_1[i]$ and $t_2[j]$ are leaves, four situations are handled. Lines 5 and 6 deal with the situation when both $t_1[i]$ and $t_2[j]$ are leaves. Lines 7 to 10 deal with the situations when one node is a leaf and the other is an inner node. The most complex part of the algorithm is shown in lines 11-21, where both $t_1[i]$ and $t_2[j]$ are internal nodes. A bipartite graph is constructed, taking their children as nodes, and the conceptual similarity between these sub-trees as the weights of edges. Function *ComputeMatching*() in line 18 returns the maximum weighted bipartite matching, which identifies most conceptual corresponding node pairs among $t_1[i]$ and $t_2[j]$'s children. The Hungarian algorithm in (Jungnickel 2008) is used to implement the function *ComputeMatching*(). Lines 22-27 deal with the case where $t_1[i]$ is matched to a child of $t_2[j]$ according to Formula (3-3). Lines 28-33 deal with the case where $t_2[j]$ is matched to a child of $t_1[i]$ according to Formula (3-4). In lines 34-37, 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. During the computation process, the most conceptual corresponding node pairs are recorded in *M*.

The computation complexity of the conceptual similarity computation algorithm is analysed. The whole algorithm can be divided into four sequential parts: 1) lines 1-21; 2) lines 22-27; 3) lines 28-33; 4) lines 34-37. The complexity of the fourth part is constant. The complexities of the second and third parts are bounded by $O(n_j)$ and $O(n_i)$ respectively. The first part has five branches in which the most complex is in lines 11-21. When calling $ComputeMatching(V_i \cup V_j, ew)$ to construct the maximum weighted bipartite matching, the maximum weighted bipartite matching method in (Jungnickel 2008) is applied, whose complexity is bounded by $O(n_i \times n_j \times \min\{n_i, n_j\})$, where n_i and n_j are the degrees of tree nodes $t_1[i]$ and $t_2[j]$ respectively. The complexity of the first part is then bounded by $O(n_i \times n_j + n_i \times n_j \times \min\{n_i, n_j\} + \min\{n_i, n_j\})$. The complexity of computing $sc_T(T_1[i], T_2[j])$ for any node pair $t_1[i]$ and $t_2[j]$ is then

obtained, which is bounded by $O(n_i \times n_j + n_i \times n_j \times \min\{n_i, n_j\} + \min\{n_i, n_j\} + n_i + n_j)$. Therefore, the complexity of the whole method is $\sum_{i=1}^{|T_1|}\sum_{j=1}^{|T_2|} O(n_i \times n_j + n_i \times n_j \times \min\{n_i, n_j\} + \min\{n_i, n_j\} + n_i + n_j)$. To omit the lower order parts, the complexity of the algorithm is finally denoted as $O(|T_1| \times |T_2| \times \sqrt{\deg(T_1) \times \deg(T_2)})$, where $\deg(T_1)$ and $\deg(T_2)$ are the degrees of T_1 and T_2 respectively.

3.3.3.2 MATCHING TREE CONSTRUCTION ALGORITHM

Given the maximum conceptual similarity tree mapping *M* between $T_1[i]$ and $T_2[j]$, their matching tree T_{M12} can be constructed. Each node of the matching tree contains a matching node pair of $T_1[i]$ and $T_2[j]$. The attributes of the matching tree node are defined in Table 3-2. The purpose of constructing the matching tree between $T_1[i]$ and $T_2[j]$ is to calculate the value similarity between $T_1[i]$ and $T_2[j]$. To make the computation of the value similarity more convenient, the values and weights of the matched node pairs are also defined in the matching tree node. The matching tree construction algorithm is illustrated by Algorithm 3-3. The algorithm has three inputs: tree $T_1[i]$, the maximum conceptual similarity tree mapping *M* between $T_1[i]$ and $T_2[j]$, and the parent node of the constructed matching tree's root. The output of the algorithm is the root of the constructed matching tree.

Table 3-2. The attribute definition of a matching tree node

Attribute name	Description
*node*1	The matched node from $T_1[i]$
*w*1	The weight of *node*1
*node*2	The matched node from $T_2[j]$
*w*2	The weight of *node*2
vtp	The value type of *node*1 and *node*2
*v*1	The value of the sub-tree rooted at *node*1
*v*2	The value of the sub-tree rooted at *node*2
children	The children of the matching tree node

Algorithm 3-3. Matching tree construction algorithm

$constructMatchingTree(T_1[i], M, pNode)$

input: tree $T_1[i]$, the maximum conceptual similarity tree mapping *M* between $T_1[i]$ and $T_2[j]$, and the parent node, $pNode$, of the constructed matching tree node

output: the matching tree of $T_1[i]$ and $T_2[j]$

```
1    if t1[i] is in the mapping M
2        t2[mi] ← t1[i]'s matched node in M
3        new matching tree node tM
4        node1(tM) ← t1[i]; node2(tM) ← t2[mi];
5        w1(tM) ← setWeight(t1[i], pNode);
6        w2(tM) ← setWeight(t2[mi], pNode);
7        if v(t1[i]) ≠ null or v(t2[mi]) ≠ null
8            if v(T1[i]) ≠ null and v(T2[mi]) ≠ null
9                vtp(tM) ← vtp(t1[i]);
10               v1(tM) ← v(T1[i]);
11               v2(tM) ← v(T2[i]);
12               return tM
13           else
14               if F1[i] ≠ ∅
15                   for s=1 to ni
16                       tc ← constructMatchingTree(T1[is], M, tM)
17                       if tc ≠ null
18                           add tc to tM's children
19                   if tM's children size > 0
20                       return tM
21   else
22       if v(t1[i]) = null and F1[i] ≠ ∅
23           for s=1 to ni
24               tM ← constructMatchingTree(T1[is], M, pNode)
25               if tM ≠ null
26                   return tM
27   return null
```

28 END

In Algorithm 3-3, lines 1-20 deal with the situation where the root of $T_1[i]$ is matched in the maximum conceptual similarity tree mapping, in which a new matching tree node t_M is constructed. The weights of the matched node pairs are calculated by the $setWeight()$ algorithm, as shown in Algorithm 3-4. If $t_1[i]$ or its mapped node is assigned value, as shown in lines 7-12, the relevant value type and values of t_M will be assigned and t_M will be returned; otherwise, if $t_1[i]$ has child nodes and its children can construct the matching tree nodes, they will be added into t_M's children. If $t_1[i]$ is not in the maximum conceptual similarity tree mapping, as shown in lines 21-26, the matching tree node constructed by its child node will be returned. If the nodes in the sub-tree $T_1[i]$ are not matched or the matched nodes are not assigned valid comparable values, the algorithm will return *null*.

The $setWeight(t_1,t_M)$ called in the matching tree construction algorithm sets the weight of a matched node. It has two inputs: t_1 is the matched node, and t_M is the parent of the matching tree node containing t_1. If t_M is *null*, $setWeight(t_1,t_M)$ calculates the continued product of the weights of nodes in the branch from t_1 to the root of that tree. If t_M is not *null*, let the ancestor of t_1 contained in t_M be denoted as t_{M1}. $setWeight(t_1,t_M)$ then calculates the continued product of the weights of nodes in the branch from t_1 to t_{M1}'s child.

Algorithm 3-4. The algorithm for setting the weights of matched nodes

setWeight(t_1,t_M)
input: the matched node t_1, and the current matching tree node's parent node t_M
output: the weight of the node t_1

1 $weight \leftarrow w(t_1)$
2 if $t_M \neq null$
3 while $parent(t_1) \neq null, parent(t_1) \neq node1(t_M), parent(t_1) \neq node2(t_M)$
4 $t_1 \leftarrow parent(t_1)$
5 $weight \leftarrow weight * w(t_1)$
6 else

7 while $parent(t_1) \neq null$ and $parent(t_1)$ is not the root
8 $t_1 \leftarrow parent(t_1)$
9 $weight \leftarrow weight * w(t_1)$
10 return *weight*
11 END

The computation complexity of the matching tree construction algorithm is analysed. The algorithm traverses the tree $T_1[i]$ and each node is visited once. Therefore, the complexity of Algorithm 3-3 is bounded by $O(|T_1|)$.

3.3.3.3 VALUE SIMILARITY COMPUTATION ALGORITHM

Given the matching tree $T_M[k]$, according to the value similarity measure model proposed in Section 3.3.2, the algorithm to compute the value similarity of the matched sub-trees is developed, and illustrated by Algorithm 3-5.

Algorithm 3-5. Value similarity computation algorithm

$sv_M(T_M[k])$
input: the matching tree $T_M[k]$
output: the value similarity of the matched sub-trees
1 if $T_M[k] = \emptyset$
2 return 0
3 else
4 if $F_M[k] = \emptyset$
5 return $sv_{tp_k}\big(v1(t_M), v2(t_M)\big)$
6 else
7 return $\sum_{s=1}^{n_k} w_{k_s} \cdot sv_T(T_M[k_s])$
8 END

The algorithm implements the Formula (3-7). The three branches in the algorithm calculate the similarity value under the three conditions listed in Formula (3-7) respectively. In line 7 of Algorithm 3-5, w_{k_s} is the weight of the matched node pair $node1(t_M[k_s])$ and $node2(t_M[k_s])$. If the measure is a symmetric measure, $w_{k_s} =$

$(w1(t_M[k_s]) + w2(t_M[k_s]))/2$. If the measure is an asymmetric measure, $w_{k_s} = w1(t_M[k_s])$. Evidently, the complexity of the value similarity computation algorithm is bounded by $O(|T_M[k]|)$.

The sub-algorithms of Algorithm 3-1 have been presented. With these sub-algorithms, the similarity between two tree-structured data can be compared comprehensively. From the complexity analysis of the three sub-algorithms, it can be seen that the complexity orders of the matching tree construction algorithm and the value similarity computation algorithm is much lower that of the conceptual similarity computation algorithm. Therefore, the complexity of the whole similarity measurement algorithm is dominated by the conceptual similarity computation algorithm, which is polynomial time complexity as discussed above. In this study, the similarity measure is used to evaluate the semantic similarity between tree-structured items or users, which have limited scales from the point of view of tree node numbers. Therefore, the complexity of the similarity measurement algorithm is acceptable.

3.4 TWO ILLUSTRATIVE EXAMPLES AND COMPARISON WITH OTHER APPROACHES

To show the effectiveness of the proposed similarity measure model on tree-structured data, two examples are provided in this section. In the first example, the process of computing the similarity between two tree-structured data in Figure 3-1 is presented to show the behaviour of the proposed algorithms in Section 3.3.3. In the second example, the proposed tree similarity model is used in the retrieve stage of a simple case-based reasoning (CBR) system to demonstrate the effectiveness of the model. The proposed model is then compared with other tree similarity evaluation methods.

3.4.1 SIMILARITY MEASURE COMPUTATION BETWEEN TWO TREE-STRUCTURED DATA

The similarity between T_1 and T_2 in Figure 3-1 is computed by the proposed similarity measurement algorithm as follows.

Step 1: the conceptual similarity between T_1 and T_2, $sc_T(T_1[1], T_2[1])$ is computed by calling $sc_T(T_1[1], T_2[1], M_{S12})$. Let the coefficient α be 0.5, $sc_T(T_1[1], T_2[1])$ is computed as 0.856. During the computation process, the maximum conceptual similarity tree mapping between T_1 and T_2, M_{S12} is constructed, which is illustrated in Figure 3-4. The matched nodes in the maximum conceptual similarity tree mapping in Figure 3-4 are connected by dashed lines.

Step 2: based on the maximum conceptual similarity mapping between T_1 and T_2, the matching tree between T_1 and T_2, T_{M12}, is constructed, as shown in Figure 3-5.

Step 3: the value similarity between T_1 and T_2, $sv_T(T_1[1], T_2[1])$ is then computed as $sv_M(T_{M12})$ by calling Algorithm 3-5. Let the value ranges of the four value types be defined in Table 3-3. The value similarity measure for each value type is defined as $sv_{tp_i}(a_1, a_2) = 1 - |a_1 - a_2|/dm_{tp_i}$, where a_1 and a_2 are two values, and dm_{tp_i} is the maximum value in the value range of the value type tp_i. $sv_T(T_1[1], T_2[1])$ is computed as 0.697.

Table 3-3. The value ranges of four value types

Value type	voice usage	mobile data usage	broadband data usage	SMS/MMS amount
Value range	[0, 2000min]	[0, 5000MB]	[0, 100000MB]	[0, 1000]

Step 4: let the weights α_1 and α_2 be both 0.5. The final similarity measurement between T_1 and T_2, $sim(T_1, T_2)$ is computed by $0.5 \cdot sc_T(T_1[1], T_2[1]) + 0.5 \cdot sv_T(T_1[1], T_2[1])$=0.776.

3.4.2 SIMILAR TREE-STRUCTURED CASES RETRIEVAL

The proposed tree similarity measure model is used to retrieve similar tree-structured cases in a CBR system in the following example.

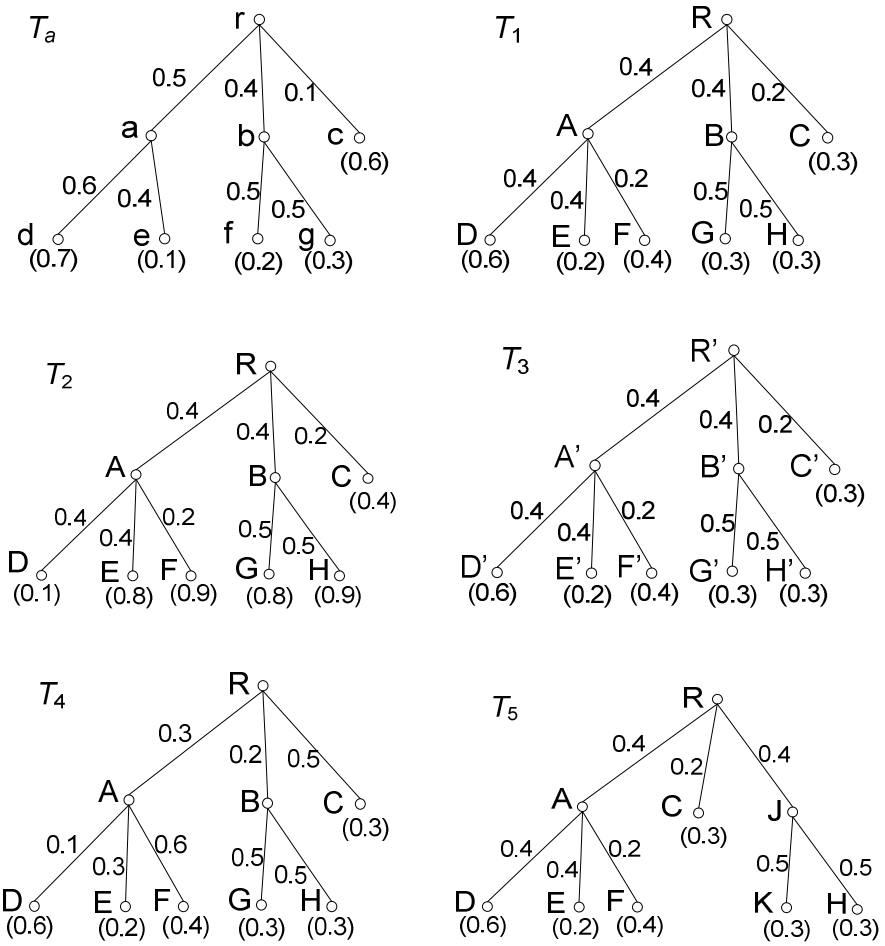


Figure 3-7. A new case T_a and five existing cases in a case base

As illustrated in Figure 3-7, all the cases in the CBR system are represented as tree-structured data. Each node in the trees is assigned a label. The leaves of the trees are assigned values. All the values are from the same range [0, 1] in the example. T_a represents a new problem to be resolved and T_1, …, T_5 represent five solved problems in the case base. The conceptual similarity between the labels is defined as follows: $sc_L(r, R) = 0.7$, $sc_L(a, A) = 0.9$, $sc_L(a, B) = 0.6$, $sc_L(b, A) = 0.5$, $sc_L(b, B) = 0.8$, $sc_L(d, D) = 1$, $sc_L(d, E) = 0.5$, $sc_L(d, G) = 0.4$, $sc_L(e, D) = 0.5$, $sc_L(e, E) = 0.9$, $sc_L(e, H) = 0.4$, $sc_L(f, F) = 1$, $sc_L(f, H) = 0.6$, $sc_L(f, G) = 0.7$, $sc_L(g, G) = 0.9$, $sc_L(g, H) = 0.7$, $sc_L(r, R') = 0.6$, $sc_L(a, A') = 0.7$, $sc_L(a, B') = 0.6$, $sc_L(b, A') = 0.5$, $sc_L(b, B') = 0.7$, $sc_L(d, D') = 0.9$, $sc_L(d, E') = 0.4$, $sc_L(d, G') = 0.3$, $sc_L(e, D') = 0.4$,

$sc_L(e, E') = 0.8$, $sc_L(e, H') = 0.3$, $sc_L(f, F') = 0.7$, $sc_L(f, H') = 0.6$, $sc_L(f, G') = 0.6$, $sc_L(g, G') = 0.7$, $sc_L(g, H') = 0.6$.

To retrieve the most similar cases to T_a, the similarities between T_a and cases in the case base are evaluated using the proposed tree similarity model. Let the coefficients α, α_1 and α_2 be all 0.5 in the model, and the similarity between two values be calculated as one minus the distance between them. The results are illustrated in Table 3-4. As can be seen in Table 3-4, T_1 is most similar to T_a, so T_1 is retrieved.

Table 3-4. Similarity between T_a and cases in the case base

	T_1	T_2	T_3	T_4	T_5
$sc_T(T_a, T_i)$	0.703	0.703	0.600	0.623	0.548
$sv_T(T_a, T_i)$	0.745	0.304	0.745	0.537	0.365
$sim(T_a, T_i)$	0.724	0.504	0.672	0.580	0.456

As seen from Figure 3-7, T_2 and T_1 are the same except for their values. Therefore, the conceptual similarity $sc_T(T_a, T_1)$ and $sc_T(T_a, T_2)$ are equal. However, as T_1's values are much closer to T_a's than T_2's, $sv_T(T_a, T_1)$ is larger than $sv_T(T_a, T_2)$, which makes T_1 more similar to T_a than T_2. T_3 and T_1 are different in terms of attributes. The concepts of T_1's attributes are more similar to T_a's than T_3's, which makes T_1 more similar to T_a than T_3. T_4 and T_1 have different attribute weights. The weights of nodes corresponding to T_a in T_4 are smaller than those in T_1, which makes T_4 less similar to T_a than T_1.

The example shows that the proposed tree similarity model takes into account all the information on nodes' structures, concepts, weights and values and it can be used to retrieve the most similar tree-structured cases effectively in CBR systems.

3.4.3 COMPARISON WITH OTHER APPROACHES

From the above examples, it can be seen that the proposed similarity evaluation model for tree-structured data has five features: 1) it considers nodes' conceptual similarities; 2) it considers the hierarchical relations between concepts; 3) it compares corresponding nodes' values; 4) it considers the influence of nodes' weights; 5) it considers the semantics of nodes' structures. The proposed method is compared with

other tree similarity evaluation methods from these five aspects. We take into account the methods of Ricci and Senter's (1998), Xue et al.'s (2009) and Bhavsar, Boley and Yang's (2004), as they can represent different types of methods, respectively. The comparison results are illustrated in Table 3-5, where " √ " represents that the method has the related feature. Table 3-5 demonstrates that none of the earlier methods can compare the tree structured data as comprehensively as our method.

Table 3-5. Comparison between our proposed method and other methods

Method	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5
Our method	√	√	√	√	√
Ricci's method	√			√	
Xue's method	√				√
Bhavsar's method	√			√	√

However, these features are essential to evaluate the similarity between complex tree-structured data. As different tree-structured data usually have different structures and attribute terms, the corresponding nodes between two tree-structured data must be identified by evaluating their conceptual similarity. As attributes in tree-structured data construct a hierarchical structure, the hierarchical relations between concepts and the semantics of nodes' structures must be considered. Nodes' values and relevant weights are essential to describe the data, so they must be compared when comparing two tree-structured data. With the above five features, the tree-structured data can be compared comprehensively and accurately.

3.5 SUMMARY

In this chapter, a comprehensive tree-structured data model which assigns tree nodes concept, value and weight is defined. A similarity measure to compare this kind of tree-structured data and the related similarity computation algorithms are presented. The proposed similarity measure takes all the information about tree structures, node

concepts, weights, and values into consideration, and can compare two tree-structured data comprehensively and accurately.

The proposed similarity measure on tree-structured data can be used to evaluate the semantic similarity between tree-structured items or user profiles in recommender systems. The conceptual computation algorithm, which is the sub-algorithm of the whole similarity measurement algorithm, can be used by itself to evaluate the conceptual similarity between two tree-structured data or to construct a maximum conceptual similarity tree mapping between two trees to identify the conceptual corresponding parts between the two trees.

CHAPTER 4
ITEM TREE AND USER REQUEST TREE-BASED HYBRID RECOMMENDER SYSTEM

4.1 INTRODUCTION

The usefulness of and necessity for semantic information has been demonstrated in many application fields of recommender systems, such as in business e-services recommendations (Lu et al. 2010, 2013; Shambour & Lu 2012). In this chapter, a hybrid recommendation approach which fully utilizes the semantic information of items and users and the requirement matching knowledge of users to items is developed.

However, it is a big challenge to evaluate the semantic similarity between items or users in some complicated applications, such as in the business to business (B2B) e-service recommender systems, because the semantic information of items or users in these applications is often presented in tree structures. For example, in a business partner searching recommender system, a business may supply several product categories, and each product category may contain several sub-categories. Under most sub-categories, there are several specific products that the business can supply which form a tree (hierarchical) structure. Different businesses have different product trees, and the differences between businesses are reflected in the following aspects:

1) The product tree structures are different for different businesses. The number of product categories provided by each business is likely to vary, which leads to the existence of differently structured product trees;

2) The concepts of nodes in trees are different, because different businesses are likely to provide dissimilar product categories;

3) The values of nodes in trees are different for different businesses, due to the fact that the product supplies or demands of different businesses are different;

4) The weights of sub-trees are different, because businesses focus on a wide range of products or product categories.

Because the similarity measure is the core technique of the recommendation approach, a semantic similarity measure comprehensively considering all the features of tree-structured data appearing in recommender systems is required. However, to the best of available knowledge, the existing recommendation methods represent user profiles or item features just as vectors of semantic concepts (Cantador 2008), and none of them considers tree-structured items or user profiles in a recommender system.

To solve this challenge, the tree-structured data model proposed in Chapter 3 is applied to model the semantic information of tree-structured items and users. During the recommendation generation process, the features of items and the requirements of users should be the main consideration. Therefore, an item tree and a user request tree are defined to represent semantic features of items and users. The similarity measure model developed in Chapter 3 is employed to evaluate the semantic similarity between item trees or user request trees. An item tree and user request tree-based hybrid recommendation approach is then developed. This approach fully utilizes the semantic information of tree-structured items and users with the requirement matching knowledge, and takes advantage of the merits of CF-based recommendation approaches.

The remainder of this chapter is organized as follows. Section 4.2 defines the item tree and user request tree for tree-structured items and user requests. An item tree and user request tree-based hybrid (IUTH) recommendation approach is presented in Section

4.3. A case study by use of the proposed IUTH recommendation approach to show its effectiveness is presented in Section 4.4. The approach has been tested by using an Australian business dataset and the Movielens dataset. The experimental evaluations and results are given in Section 4.5. Finally, the summary is given in Section 4.6.

4.2 ITEM TREE AND USER REQUEST TREE

4.2.1 ITEM TREE AND USER REQUEST TREE DEFINITIONS

Based on the tree-structured data model defined in Section 3.2, the item trees and user request trees are defined as follows.

Definition 4-1. An item tree is a tree-structured data model to describe the characteristics of tree-structured items in a specific application domain. The features of an item tree are described as follows:

1) A set of node attributes are defined to assign each node a concept to express one aspect of item features. Nodes at different depths represent concepts with different abstraction levels. The nodes at lower levels represent more specific item features;

2) A node concept similarity measure is defined on the node attributes to infer the semantic similarity between nodes;

3) A set of value types and related value ranges are defined for the item tree nodes to express the quantity or quality degrees of the item features. The value similarity of each value type is also defined;

4) For some specific nodes in an item tree, a value type and a value are assigned;

5) Each node is assigned a weight to express the importance degree of the item feature represented by the node.

Definition 4-2. A user request tree is a tree-structured data model to describe the tree-structured requirements of a user in the application domain. The user request tree is defined based on the item tree. They have the same node concept definition. The node

value represents the quantity or quality degree of the user's requirement for a specific item feature. The node weights represent the user's preferences for different item features.

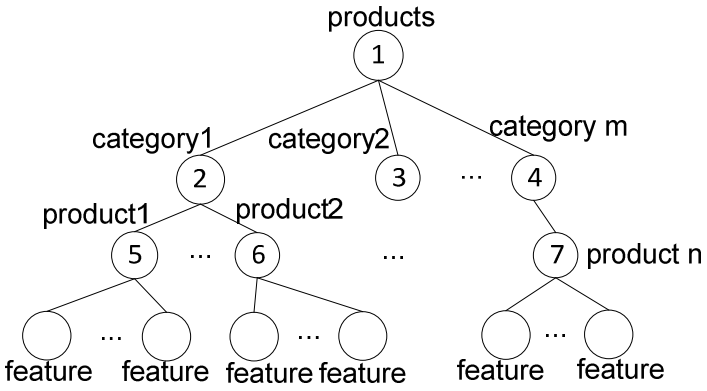
The semantic similarity between item trees or user request trees is evaluated by use of the similarity measure on tree-structured data developed in Chapter 3. As discussed in Chapter 3, there are two types of similarity measures, symmetric (denoted as sim_{sym}) and asymmetric (denoted as sim_{asym}), for different application situations. When two tree-structured data to be compared are equally treated and the node weights of both trees are considered, the symmetric measure will be applied. Otherwise, when the node weights of one tree are mainly considered, the asymmetric measure will be applied.

An example of item trees and user request trees in business partner recommender systems is given below to illustrate the item tree and the user request tree definitions and their semantic similarity measure evaluation.

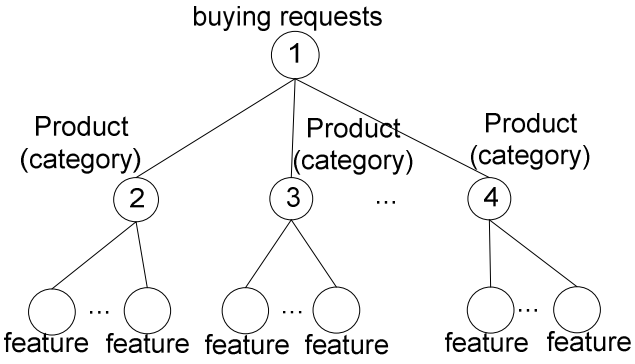
4.2.2 AN EXAMPLE OF THE ITEM TREE AND USER REQUEST TREE

In business partner (buyers or suppliers) recommendation applications, "items" refer to the potential business partners (potential supplier partners supplying products and potential buyer partners having buying requests), and "users" refer to the businesses finding partners. The business product and buying request data are the main attributes of "items" and "users", which often present in tree structures. Therefore, item trees and user request trees are employed to describe items and users respectively during the recommendation process. A business's products construct a product tree, and its buying requests construct a buying request tree. When a user wants to obtain recommendations of supplier partners, the potential suppliers' product trees are taken as the item trees, and the user's buying request tree is taken as the user request tree. When a user wants to obtain recommendations of buyer partners, the potential buyers' buying request trees are taken as the item trees, and the user's product tree is taken as the user request tree.

The structures of a product tree and a buying request tree are illustrated respectively in Figure 4-1. In the product trees and buying request trees, businesses can specify the product categories and sub-categories, the specific products, and even the features of the products. Due to the flexibility of tree structures, several levels of product categories can be specified in the trees.



(a)



(b)

Figure 4-1. A product tree structure (a) and a buying request tree structure (b)

According to the tree-structured data model definition, the features of a product tree and a buying request tree are described as follows.

1) Node concept

For each tree node, two attributes are defined to represent its concept:

- *Label*: This attribute assigns a node some specific meaning. It is defined as a $label$ function, $label: V \rightarrow L$, where L is a label term set. In this example, the label terms consist of product or product category names, product feature names and some indicative labels. The product category and product feature names are pre-defined by domain experts. The product names are collected when businesses input their products into the system. The indicative labels, such as "root", and "product", "buying request" as shown in Figure 4-1, are used to construct the tree structures.

- *Category*: This attribute is used to infer the semantic relations between tree nodes. It is defined as a $category$ function, $category: V \rightarrow \{\Lambda \cup C\}$, where C is the product category name set. When tree nodes represent products/categories, it is not sufficient to compare them by only comparing the product/category names. Therefore, a product category is introduced to infer the semantic relations between the products/categories. A product category is usually presented in a tree structure. The product category tree in this example is illustrated in Figure 4-2.

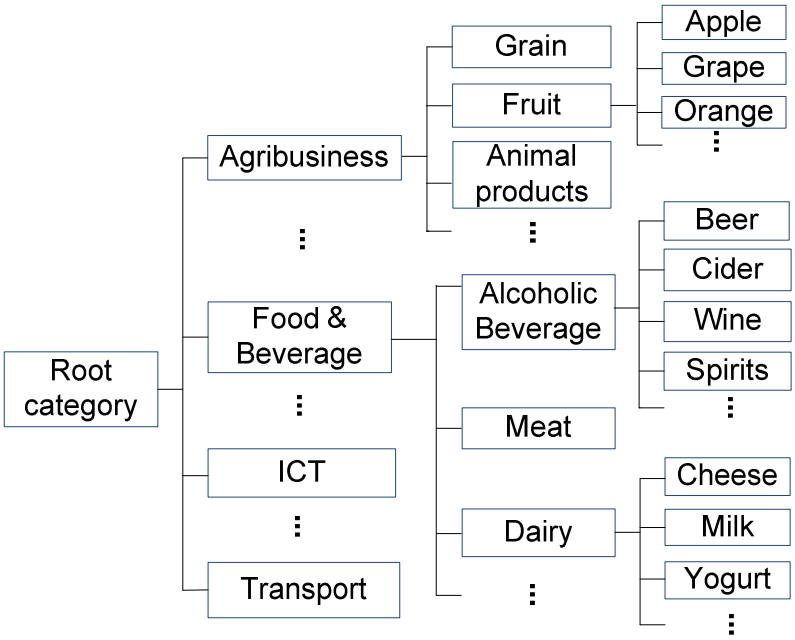


Figure 4-2. A product category tree

2) Node concept similarity

The node concept similarity between two nodes is calculated based on the *category* and *label* attributes. If two nodes are assigned a *category*, the category similarity will be taken as the node concept similarity. Otherwise, their labels are compared. For any two nodes *u* and *v*, the node concept similarity *sc*(*u*,*v*) is defined as:

$$sc(u,v) = \begin{cases} sim_{category}\big(category(u), category(v)\big), & category(u) \neq \emptyset \text{ and } category(v) \neq \emptyset \\ sim_{label}\big(label(u), label(v)\big), & category(u) = \emptyset \text{ or } category(v) = \emptyset \end{cases}. \quad (4\text{-}1)$$

The category similarity is calculated based on the product category tree. The methods for computing category similarities within a category tree can be grouped into two basic types: the edge-based methods and the node-based methods (Leacock & Chodorow 1998; Schickel-Zuber & Faltings 2007). Schickel-Zuber and Faltings (2007) compared various existing methods and concluded that there are only marginal differences between different methods in terms of mean absolute error measure. This conclusion encourages this study to adopt a relatively simple similarity evaluation method. When comparing two sub-categories, the category similarity between them should be higher if the depth of their least super-category is deeper with respect to the depth of these two sub-categories. Based on the basic idea, the category similarity is defined as follows.

Definition 4-3. Given a category tree, for two categories c_1 and c_2, their category similarity is defined as:

$$sim_{category}(c_1, c_2) = \frac{depth(lca(c_1,c_2))}{\min\{depth(c_1),depth(c_2)\}}, \quad (4\text{-}2)$$

where $lca(c_1, c_2)$ is the least common ancestor of c_1 and c_2 in the category tree, and *depth*() is the depth of the node in the category tree.

Take the categories in Figure 4-2 as examples: $sim_{category}(Cheese, Milk) = depth(Dairy)/\min\{depth(Cheese), depth(Milk)\} = 2/3$; $sim_{category}(Beer, Milk) = depth(Food\ \&\ Beverage)/\min\{depth(Beer), depth(Milk)\} = 1/3$; $sim_{category}(Meat, Dairy) = depth(Food\ \&\ Beverage)/\min\{depth(Meat), depth(Dairy)\} = 1/2$.

The label similarity compares two labels. For two labels l_1 and l_2, $sim_{label}(l_1, l_2) = 1$ if l_1 and l_2 are the same; $sim_{label}(l_1, l_2) = 0$ otherwise.

3) Node value

A value range of the supply or demand amount for the product/category is defined.

For two values v_1 and v_2 in the value range, the value similarity, $sv(\cdot)$ is defined to evaluate their similarity: $sv(v_1, v_2) = 1 - |v_1 - v_2|/diam$, where $diam$ is the maximum distance of values in the value range.

4) Node weights

The weights of nodes in the product tree and buying request tree are specified by business users.

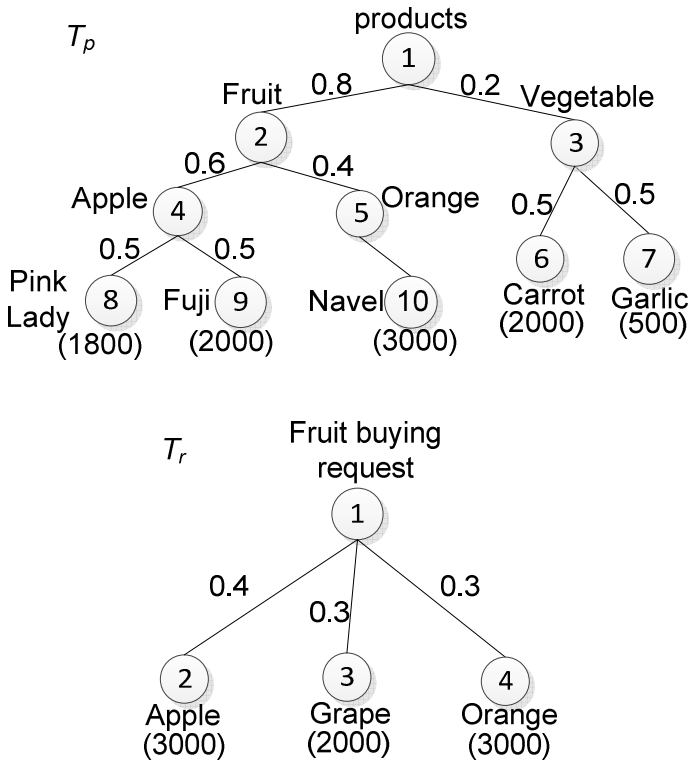


Figure 4-3. Product tree and buying request tree examples

A product tree T_p and a buying request tree T_r are shown in Figure 4-3. T_p represents the product tree of a fruit and vegetable supply company. The company supplies two

kinds of apple products, one kind of orange product, carrot and garlic. The weights of the products and product categories are expressed as the node weights, which are indicated by the numbers beside the edges. The supply ability of each product is described as the node value, which is represented by the number under the leaf node. T_r is the buying request tree of another company, which requires three kinds of fruit. The numbers in the bracket under the nodes represent the demand of the products.

Based on the similarity measure on tree-structured data presented in Section 3.3, the matching degree of T_p to T_r can be computed by $sim_{asym}(T_r, T_p)$. The conceptual similarity between T_r and T_p is first calculated by Formula (3-2) as $sc_T(T_r, T_p) = 0.7$. The maximum conceptual similarity tree mapping is constructed during the computation, and the mapping tree between T_r and T_p is then constructed, which is shown as T_M in Figure 4-4. Let the value range of the supply or demand amount be [0, 5000]. Based on T_M, the value similarity between T_r and T_p is calculated by Formula (3-8) as $sv_T(T_r, T_p) = 0.64$. Finally, the matching degree of T_p to T_r is be computed by weighted aggregating $sc_T(T_r, T_p)$ and $sv_T(T_r, T_p)$ using Formula (3-1).

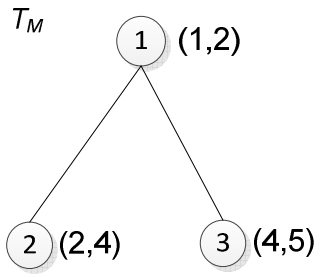


Figure 4-4. The matching tree between T_r and T_p in Figure 4-3

4.3 AN ITEM TREE AND USER REQUEST TREE-BASED HYBRID RECOMMENDATION APPROACH

In this section, an item tree and user request tree-based hybrid (IUTH) recommendation approach is presented. As depicted in Figure 4-5, the IUTH algorithm takes three kinds of information resources as inputs: the user-item rating matrix, the item trees and the user request trees, and produces a user-item prediction matrix as outputs. In Figure 4-5, the request matching similarity module considers the semantics of both users and items and computes the matching degree of an item to a user. The item-based semantic similarity module considers the semantic information of items and computes the item-based semantic similarity for any pair of items. Both the request matching similarity module and the item-based semantic similarity module utilize the proposed similarity measure on tree-structured data. The item-based CF similarity module computes the item-based CF similarity between any pair of items. The total weighted similarity module uses the item-item semantic similarity matrix and the item-item CF similarity matrix, to combine both similarity values to obtain the total weighted similarity value for any pair of items. The neighbour selection module selects a set of nearest neighbours of items that are most similar to the target items. The item-based CF-based predicted rating generation module computes the predicted ratings by use of ratings of the selected neighbours. The requirement matching-based predicted rating generation module computes the predicted ratings to unrated items based on the matching degrees of the items to the requirements of users. The total weighted predicted rating generation module hybridizes the above two kinds of prediction values, generates the final predicted user-item rating matrix, and then produces the final ranked recommendations list for the active user.

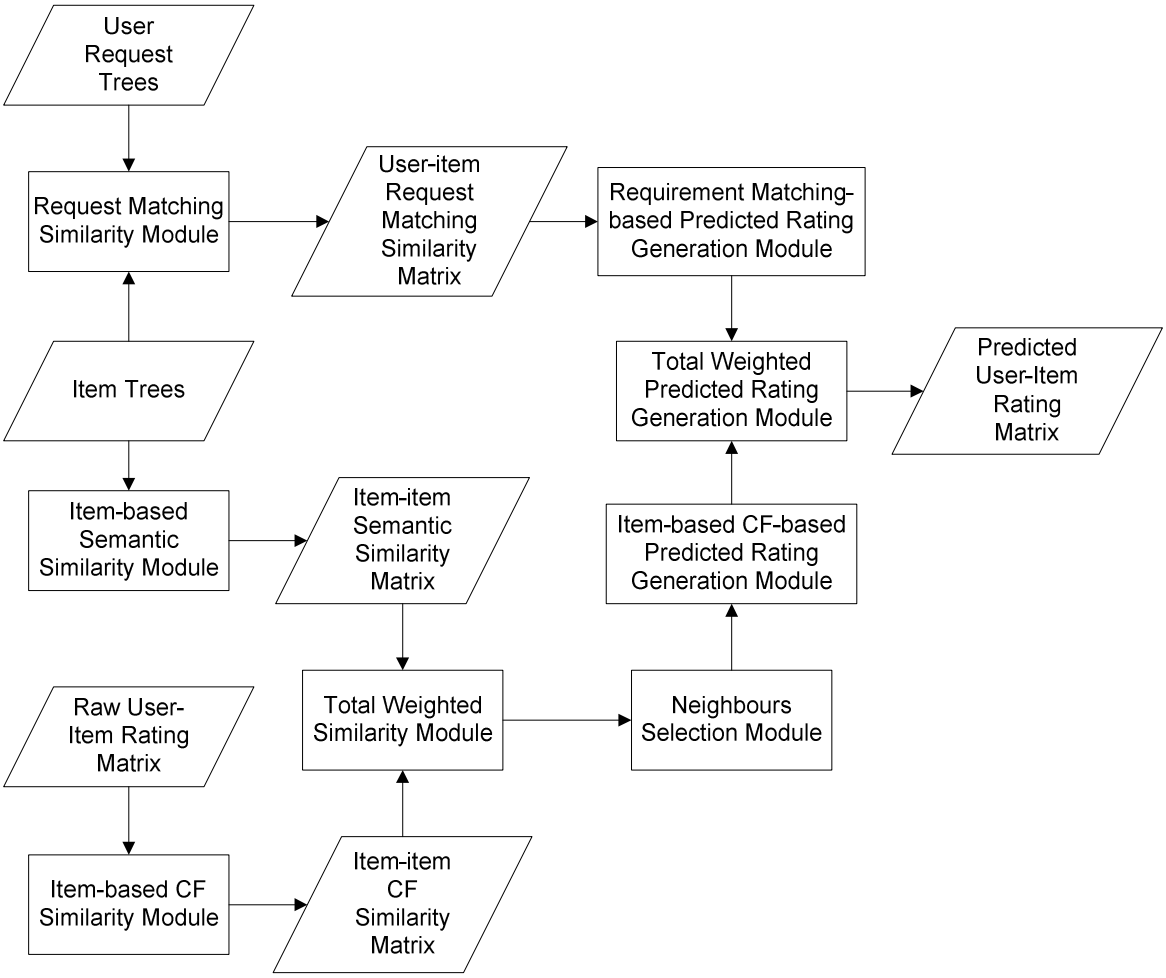


Figure 4-5. An item tree and user request tree-based hybrid recommendation approach

Let the user request tree of an active business user u be $T_{req,u}$, and the item tree of a target item t be $T_{item,t}$. The predicted rating of the user u to the item t is calculated by the IUTH recommendation algorithm, which consists of six steps and is described as follows.

Algorithm 4-1. An item tree and user request tree-based hybrid recommendation algorithm

input: the user-item rating matrix, the user request tree $T_{req,u}$ of the active business user u, the item tree $T_{item,t}$ of the target item t, and the item trees of the items rated by u

output: the predicted rating of u to target item t

Step 1: calculate the request matching degree of item t to user u

The request matching degree of item *t* to user *u* is assessed in this step by calculating the similarity between the user request tree of *u* and the item tree of *t*. During the matching degree computation, the node weights of the user request tree should be the main consideration. For this reason, the asymmetric (sim_{asym}) similarity measure is used in this step.

$$s_m(u,t) = sim_{asym}(T_{req,u}, T_{item,t}). \quad (4\text{-}3)$$

Step 2: calculate the semantic similarity between the target item *t* and the items rated by user *u*

Let user *u* have rated *m* items, denoted as $\{b_1, b_2, \ldots, b_m\}$. Let the item tree of b_i be T_{item,b_i}, $i = 1, 2, \ldots, m$. The semantic information of items is expressed by item trees, and the semantic similarity between items is assessed by calculating the similarity between their item trees. To compare two item trees objectively, the node weights of both item trees should be considered. As a consequence, the symmetric (sim_{sym}) similarity measure is used in this step. The semantic similarity between *t* and b_i is calculated as

$$s_{sem}(b_i,t) = sim_{sym}(T_{item,b_i}, T_{item,t}). \quad (4\text{-}4)$$

Step 3: calculate the item-based CF similarity between the target item *t* and the items rated by user *u*

Based on the user-item rating matrix, this step computes the item-based CF similarity between item *t* and the items that have been rated by user *u*, $\{b_1, b_2, \ldots, b_m\}$. The item-based CF similarity between *t* and b_i is calculated as

$$s_{CF}(b_i,t) = \frac{\sum_{k \in U_{b_i,t}} rs_{k,b_i} \times rs_{k,t}}{\sqrt{\sum_{k \in U_{b_i,t}} {rs_{k,b_i}}^2 \times \sum_{k \in K} {rs_{k,t}}^2}}, \quad (4\text{-}5)$$

where $U_{b_i,t}$ is the set of users that have rated both items b_i and *t*, rs_{k,b_i} and $rs_{k,t}$ are the ratings of user *k* to items b_i and *t* respectively.

Step 4: calculate the total similarity

The total weighted similarity module computes the total similarity between two items by integrating the item-based CF similarity and the semantic similarity values obtained in the last two steps. In this module, the weighted hybridization method (Burke 2007) is applied to obtain the total weighted similarity value between the item *t* and item b_i, *i* = 1, 2, …, *m*.

$$s(b_i, t) = \beta \times s_{sem}(b_i, t) + (1 - \beta) \times s_{CF}(b_i, t), \quad (4\text{-}6)$$

where $\beta \in [0, 1]$ represents the relative importance of the item-based semantic similarity, and $1 - \beta$ is the relative importance for the item-based CF similarity. The relative weighting is adopted to adjust the importance of the item-based semantic similarity and the item-based CF similarity. When $\beta = 1$, the item-based semantic similarity value is used as the final similarity value for predictions, while if $\beta = 0$, then only the item-based CF similarity value is used for predictions. β can be determined according to the data provision situations of the specific applications and the application requirements.

Step 5: select neighbours

The neighbour selection module selects a set of nearest neighbours that contains items that are most similar to the target item. In this module, the items most similar to the target item in terms of the total weighted similarity values are selected for generating the prediction. Currently, two methods have been employed in recommender systems: the Top-*N* method (e.g., a predefined number of neighbours with greatest similarity are selected), and the similarity threshold (e.g., all neighbours with similarity exceeding a certain threshold are selected). In this approach, the Top-*N* method is used as recommended by Herlocker et al. (2002).

Step 6: calculate the predicted rating

Let the rating of u to b_i be rs_{u,b_i}. The predicted rating of u to target item t is calculated as:

$$rs_{u,t} = \theta \times s_m(u, t) \times r_{\max} + (1 - \theta) \times \frac{\sum_{i=1}^{m} s(b_i,t) \times rs_{u,b_i}}{\sum_{i=1}^{m} s(b_i,t)}, \quad (4\text{-}7)$$

where $\theta \in [0,1]$, $r_{\max}$ represents the maximum value of ratings. The formula contains two parts. $s_m(u,t) \times r_{\max}$ is the requirement matching-based predicted rating. If the target item is exactly matched to the user's requirement, the target item should obtain the highest rating. $\sum_{i=1}^{m} s(b_i,t) \times rs_{u,b_i} / \sum_{i=1}^{m} s(b_i,t)$ is the traditional item-based CF-based predicted rating. θ represents the relative importance of the requirement matching-based predicted rating, and $1-\theta$ is the relative importance for the item-based CF-based predicted rating. The relative weighting is adopted to adjust the importance of the two parts. When $\theta = 1$, the requirement matching-based predicted rating is used as the final predicted rating, while if $\theta = 0$, then the item-based CF-based predicted rating is used as the final predicted rating.

The predicted ratings for all the potential items are calculated and ranked accordingly. *K* potential items with the highest predicted ratings are selected as the final recommendations.

The proposed recommendation approach draws strength from the CB, CF and KB recommendation approaches. The requirement matching in the predicted rating calculation formula fully utilizes the semantic information of user requirements and item descriptions and the matching knowledge. This makes the recommendation easier to explain. Because it does not require any rating information, the proposed approach can deal with the new user and new item problems. By using the item-based CF similarity in traditional item-based CF predicted rating, the merits of the CF recommendation approach can be exploited. The semantic similarity between items fully utilizes the semantic information of items, and can also deal with the new item problem.

4.4 A CASE STUDY

In this section, a case study is given to illustrative the proposed IUTH recommendation approach. In the case study, the proposed recommendation approach is applied into a business partner recommender system.

Assume there are five business users (U_1 to U_5) seeking business partners in the recommender system. U_1 and U_2 are two bottle shops that want to buy wine and soft

drink, and cider and wine, respectively. U_3 is a hotel, which wants to buy wine, cider, soft drink and spirit. U_4 is a pub, which wants to buy spirit, wine and juice. These four users are seeking supplier recommendations. U_5 is a fruit supplier company which supplies apple, orange and grape. U_5 is seeking buyer recommendations. The buying request trees of U_1 to U_4 and the product tree of U_5 are illustrated in Figure 4-6. The features of tree nodes are defined as the definitions in Section 4.2.2.

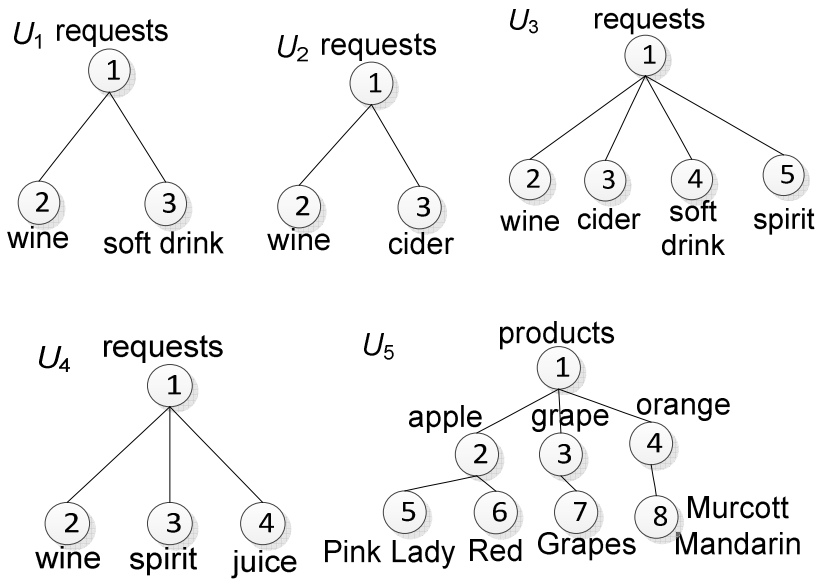


Figure 4-6. The product tree and buying request trees of business users

Table 4-1. Products of the businesses

Businesses	Product Category	Product
B_1	wine	Red Label; Jim Beam
B_2	cider	Ciders; Breeze
B_3	wine	Mitchelton; Petaluma
	beer	Beers
B_4	cider	Rtd`s
	wine	2010 Rouge
B_5	spirit	Spirits
	wine	Chardonnay 2009

There are five winery businesses (B_1 to B_5) available in the system. Their products and buying requests are shown in Table 4-1 and Table 4-2 respectively, which construct

tree structures. The product trees of B_1 to B_5 are illustrated in Figure 4-7, and their buying request trees are illustrated in Figure 4-8. The features of tree nodes are defined as the definitions in Section 4.2.2.

Table 4-2. Buying requests of the businesses

Businesses	Requested Product Category
B_1	Grape
B_2	Apple
B_3	Grape
	Vegetable
B_4	Grape
	Apple
B_5	Grape
	Apple

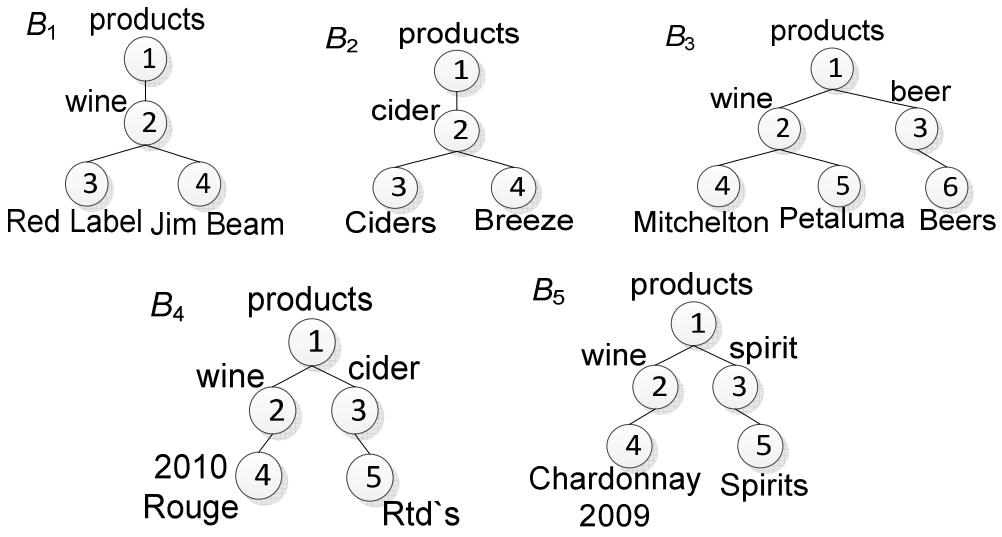


Figure 4-7. The product trees of the businesses

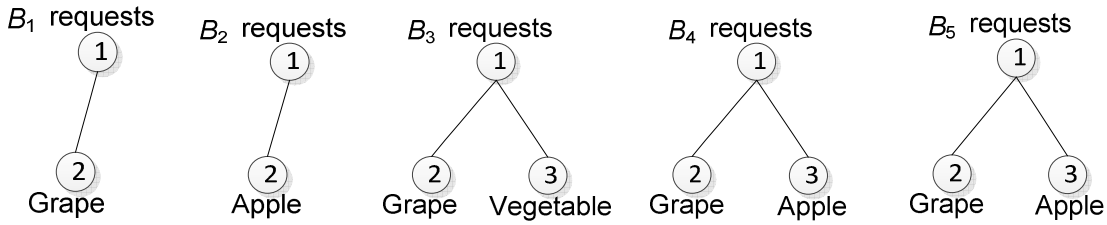


Figure 4-8. The buying request trees of the businesses

To recommend suppliers to users U_1 to U_4, their request trees are taken as the user request trees, and the product trees of B_1 to B_5 are taken as the item trees in the IUTH recommendation algorithm. The existing user supplier rating matrix is illustrated in Table 4-3. It can be seen from the table that U_4 is a new user, and B_5 is a new item. The recommendation process is given in detail as follows.

Table 4-3. User supplier rating matrix

	B_1	B_2	B_3	B_4	B_5
U_1	4		2	3	
U_2	3	4	5		
U_3	5	5		4	
U_4					

Step 1: calculate the request matching degrees.

Let the product tree of business B_i be denoted as T_{prod,B_i}, the buying request tree of business user U_j be denoted as T_{req,U_j}. For any unrated supplier B_i of business user U_j, $i,j \in \{1,2,3,4,5\}$, the matching degree of the product tree of B_i to the buying request tree of U_j is computed by Formula (4-3) as $s_m(U_j, B_i) = sim_{asym}(T_{req,U_j}, T_{prod,B_i})$. The calculated request matching degrees are illustrated in Table 4-4.

Table 4-4. Request matching degrees

	B_1	B_2	B_3	B_4	B_5
U_1		0.25			0.75
U_2				1.0	0.75
U_3			0.375		0.5
U_4	0.333	0.167	0.5	0.5	0.667
U_5	0.333	0.333	0.333	0.667	0.667

Step 2: calculate the semantic similarity between the businesses.

In this step, the semantic similarities between the target item and the rated items are calculated. For any two businesses B_i and B_j, the semantic similarity between them is computed by Formula (4-4) as $s_{sem}(B_i, B_j) = sim_{sym}(T_{prod,B_i}, T_{prod,B_j})$. The semantic similarities between the businesses are illustrated in Table 4-5.

Table 4-5. Semantic similarities between businesses

	B_2	B_3	B_4	B_5
B_1	0.5	0.75	0.75	0.75
B_2		0.375	0.75	0.375
B_3			0.75	0.75
B_4				0.75

Step 3: calculate the item-based CF similarity between the businesses.

In this step, the item-based CF similarities between the target item and the rated items are calculated. Based on the user supplier rating matrix illustrated in Table 4-3, the item-based CF similarities between the businesses are computed by Formula (4-5), as shown in Table 4-6.

Table 4-6. Item-based CF similarities between businesses

	B_2	B_3	B_4	B_5
B_1	0.991	0.8542	0.9995	N/A
B_2		N/A	N/A	N/A
B_3			N/A	N/A
B_4				N/A

Step 4: calculate the total similarity.

Let $\beta = 0.5$, the total similarity between businesses is calculated by Formula (4-6), as illustrated in Table 4-7.

Table 4-7. Total similarities between businesses

	B_2	B_3	B_4	B_5
B_1	0.7455	0.8021	0.8747	0.75
B_2		0.375	0.75	0.375
B_3			0.75	0.75
B_4				0.75

Step 5: select neighbours.

For each unrated supplier of a business user, the most similar rated suppliers of the business user are selected in this step. As the number of potential suppliers is very small in this case study, all the rated suppliers of a user are selected.

Step 6: calculate the predicted rating.

The predicted ratings are calculated by Formula (4-7). The results are illustrated in Table 4-8.

Table 4-8. Predicted ratings to suppliers

	B_1	B_2	B_3	B_4	B_5
U_1		2.224			3.375
U_2				4.474	3.875
U_3			3.243		3.55
U_4	1.667	0.833	2.5	2.5	3.333

The predicted ratings for users U_1 to U_3 are calculated as the weighted sum of two parts, the requirement matching part and the traditional item-based CF part. As U_4 is a new user, only the requirement matching part is calculated and taken as the final predicted rating. Based on the predicted ratings, the businesses can be ranked, and the most interested businesses for users are recommended.

U_5 needs a set of buyer recommendations. The IUTH recommendation algorithm can be applied in the buyer recommendations, in which the buying request trees of B_1 to B_5 are taken as the item trees, and the product tree of U_5 is taken as the user request tree. The predicted ratings of U_5 to potential buyers B_1 to B_5 are then computed, as shown in Table 4-9.

Table 4-9. Predicted ratings to buyers

	B_1	B_2	B_3	B_4	B_5
U_5	1.67	1.67	1.67	3.33	3.33

From the case study, it can be seen that the semantic information of businesses and the requirement matching knowledge are fully utilized in the recommendation process. Both new item (B_5) and new user (U_4 and U_5) problems are solved.

4.5 EXPERIMENTAL EVALUATION

This section evaluates the performance of the proposed item tree and user request tree-based hybrid recommendation approach by using an Australian business dataset and the MovieLens dataset. The evaluation is made by comparing the proposed approach with typical alternative recommendation approaches. This section includes the data sets used for evaluation, the evaluation metrics and the evaluation results.

4.5.1 EVALUATION DATA SETS

Two datasets are used in the experiments.

1) Australian business dataset

This dataset was collected from business data in Australia. It contains the ratings of businesses to their business partners (buyers or suppliers). The basic business information, product information and buying request information was collected in the dataset for each business. Basic information includes for example a company's business type, scale and contact information. Product information describes the products supplied by the business in detail, which contains the product categories and detailed features of the products. There is also 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 a business. The products and buying requests of a business are both presented as tree structures and described as tree-structured data. The structures of the product tree and the buying request tree are illustrated in Figure 4-1, and the tree node features are defined in Section 4.2.2. There are 130 business users in the dataset, which includes 363 ratings to suppliers and 360 ratings to buyers. The sparsity level[1] of the supplier rating matrix is 97.84% ($sparsity\ level = 1 - Density = 1 - 363/(130 \times 129) = 0.9784$), and the

[1] The sparsity level of any dataset is defined as $1 - Density$. $Density\ (Dataset) = number\ of\ nonzero\ entries/total\ number\ of\ entries$, where the number of total entries is calculated by multiplying the number of users by the number of items; the number of nonzero entries is the total number of overall ratings in the dataset.

sparsity level of the buyer rating matrix is 97.85% ($sparsity\ level = 1 - Density = 1 - 360/(130 \times 129) = 0.9785$).

In the experiment, 20% of all the ratings, which contains the ratings to suppliers and the ratings to buyers, are randomly selected as the testing set. The proposed IUTH recommendation approach is tested to recommend both buyer and supplier partners.

2) The HetRec 2011 MovieLens data set

This HetRec 2011 MovieLens data set (http://ir.ii.uam.es/hetrec2011) is extended from the MovieLens10M dataset (http://www.grouplens.org/node/73). It includes the users' ratings to movies, tags, movie genres, directors, actors and so on. The MovieLens dataset has commonly been used in recommender system research, and many other recommendation approaches have been tested using this data set. This dataset is used in the evaluation to enable a fair comparison of the performance of this approach with other recommender systems. In this experiment, each movie is represented as a tree-structured object. The structure is shown in Figure 4-9. There are 855598 explicit ratings of 10109 movies from 2113 users in the dataset. The sparsity level of the dataset is 95.99% ($sparsity\ level = 1 - Density = 1 - 855598/(2113 \times 10109) = 0.9599$).

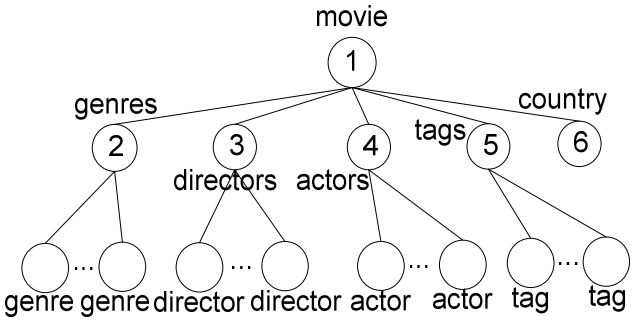


Figure 4-9. The movie tree structure

In the experiment, the ratings of each user are split into two parts, the training set and the testing set, and the latest 10 ratings of each user make up the testing sets. In the experiment, each movie is seen as a tree-structured item, and represented by an item tree. The predicted rating of the target user to the target movie is calculated by the IUTH

recommendation algorithm. Since there is no requirement information of the users, $\theta = 0$ in the Formula (4-7).

4.5.2 EVALUATION METRICS

In this study, the proposed recommendation approach is evaluated by experiments from both the accuracy metric and coverage rate through comparing with three existing approaches.

1) Accuracy metric

The Mean Absolute Error (MAE) is the most widely used accuracy metric in recommendation research (Resnick et al. 1994), and is adopted in this study. MAE measures the accuracy by computing the average absolute deviation between the recommender system's predicted rating against the actual rating assigned by the user. Note that a lower MAE value represents a higher recommendation accuracy. Let r_{a_i} and r_{p_i} be the actual and predicted ratings for item *i* respectively. Given the set of the actual/predicted rating pair (r_{a_i}, r_{p_i}) for all the *n* items in the test set, the MAE is computed by

$$MAE = \frac{\sum_{i=1}^{n} \left| r_{a_i} - r_{p_i} \right|}{n}. \quad (4\text{-}8)$$

2) Coverage rate

The coverage measure evaluates the ability of a given recommender system to provide recommendations. The coverage is computed as the percentage of items for which a prediction is requested and for which the recommender system is able to make a prediction (Herlocker, Konstan & Riedl 2002). Let *n* be the number of available items and let I_p denote the number of items for which a prediction can be made; the coverage rate is computed by

$$coverage = \frac{I_p}{n}. \quad (4\text{-}9)$$

4.5.3 BENCHMARK RECOMMENDATION APPROACHES

To compare the performance of the proposed IUTH recommendation approach, three benchmark recommendation approaches were selected. Taking into consideration that the IUTH recommendation approach hybridizes the item-based recommendation algorithm, Sarwar's item-based CF approach, which employs Pearson correlation-based similarity (denoted as SarwarCF) (Sarwar et al. 2001), and Deshpande and Karypis's item-based CF approach, which employs cosine-based similarity (denoted as DeshpandeCF) (Deshpande & Karypis 2004) were implemented. These approaches have been widely utilised as benchmarks for evaluating recently proposed recommendation approaches (Kim et al. 2010; Shambour & Lu 2012). Considering that the semantic information of items and users is fully used in the IUTH recommendation approach, a typical hybrid recommendation approach which hybridizes item-based semantic similarity and item-based CF similarity techniques (denoted as HSR) (Lu et al. 2010, 2013) was also implemented.

4.5.4 EVALUATION RESULTS ON THE AUSTRALIAN BUSINESS DATASET

The evaluation results of accuracy metric and coverage rate on the Australian business dataset are shown in this sub-section.

1) Accuracy metric

Using the Australian business dataset, experiments have been performed to compare the recommendation accuracy performance of the proposed approach (IUTH) with respect to the benchmark approaches, and the MAE of these approaches at different neighbourhood sizes was recorded. The MAE comparison between the IUTH recommendation approach and other benchmark approaches is shown in Figure 4-10. It can be seen from the results that the accuracy performances of the IUTH recommendation approach are better than the other three approaches, which indicates that

the IUTH recommendation approach is well-suited to deal with the tree-structured business data in recommender systems.

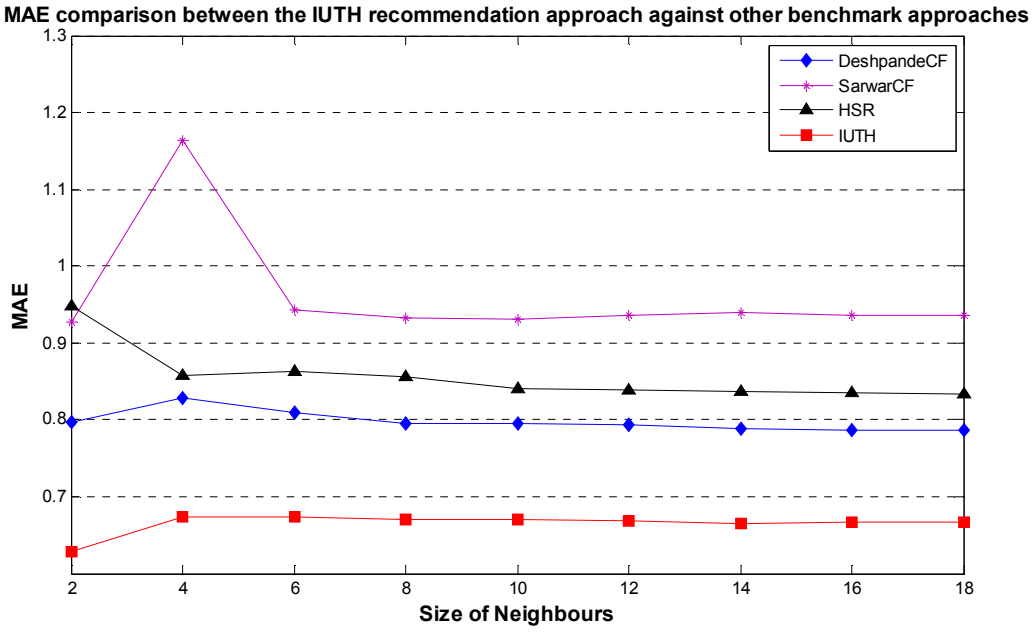


Figure 4-10. Recommendation accuracy comparison between the IUTH recommendation approach and other benchmark approaches on different numbers of neighbours with the Australian business dataset

2) Coverage rate

The coverage rates of the four recommendation approaches on the Australian business dataset are shown in Table 4-10. Because the sparsity level is very high in the dataset, the coverage rates of DeshpandeCF and SarwarCF are strongly influenced. Since the HSR approach combines the item semantic similarity, it is able to recommend new items, but it still suffers from the new user problem. Consequently, the coverage rate of HSR is still influenced. Because the IUTH recommendation approach fully utilizes the semantic information of items and combines the KB recommendation technique, it is able to recommend all the items.

Table 4-10. Coverage rate of the recommendation algorithms for the Australian business dataset

	DeshpandeCF	SarwarCF	HSR	IUTH
coverage	0.8552	0.7517	0.8897	1

4.5.5 EVALUATION RESULTS ON THE MOVIELENS DATASET

The evaluation results of accuracy metric and coverage rate on the Movielens dataset are shown in this sub-section.

1) Accuracy metric

Experiments with the Movielens dataset were conducted, and the MAE of the IUTH recommendation approach and the benchmark approaches at different neighbourhood sizes were recorded. Since different items in the dataset receive different numbers of ratings, the accuracy performance is analysed separately for different situations.

Taking all the items in the dataset into consideration, the MAE comparison between the IUTH recommendation approach and other benchmark approaches on different numbers of neighbours is shown in Figure 4-11. It can be seen from the result that the accuracy performances of IUTH and SarwarCF are equally matched and are better than DeshpandeCF and HSR.

The proposed IUTH recommendation approach and HSR approach are both capable of recommending new items. For new items, only the item-based semantic similarity is used in the recommendation. Taking only new items in the Movielens dataset into consideration, the MAE comparison between the IUTH recommendation approach and the HSR approach on different numbers of neighbours is shown in Figure 4-12. It can be seen from the results that the accuracy of IUTH is better than HSR on the whole, which reflects that IUTH uses the semantic information effectively.

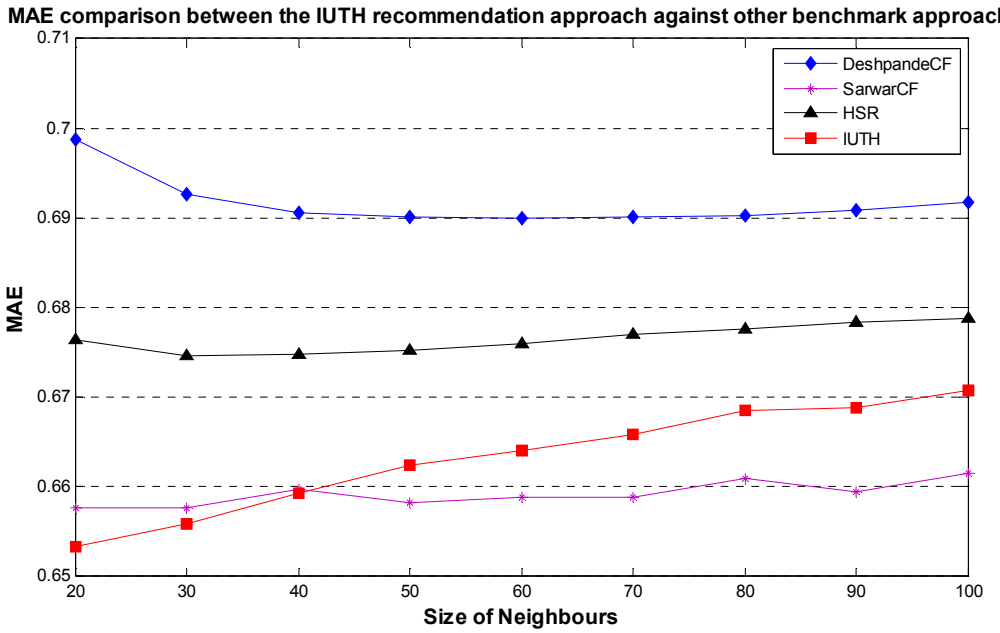


Figure 4-11. Recommendation accuracy comparison between the IUTH recommendation approach and other benchmark approaches on different numbers of neighbours with all items in Movielens dataset

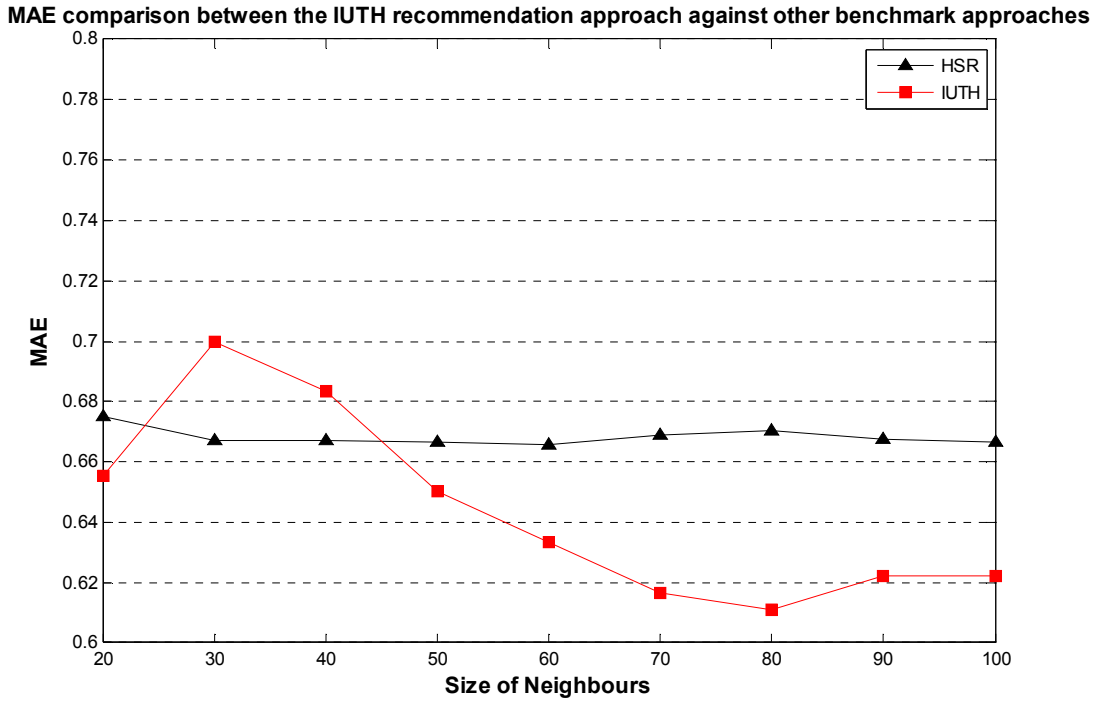


Figure 4-12. Recommendation accuracy comparison between the IUTH recommendation approach and HSR approach on different numbers of neighbours with new items in Movielens dataset

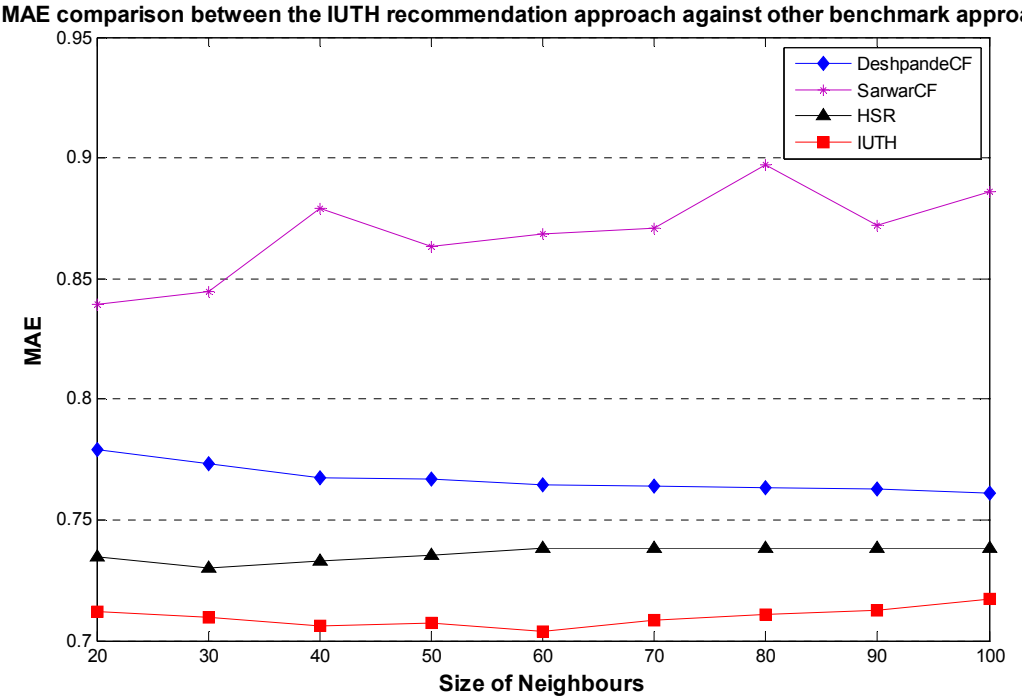


Figure 4-13. Recommendation accuracy comparison between the IUTH recommendation approach and other benchmark approaches on different numbers of neighbours with items rated less than twenty times in Movielens dataset

For those items that are not rated by a sufficient number of users, the performances of CF-based recommendation approaches are not guaranteed. However, as the semantic features are fully considered, the recommendation performance is not influenced in the IUTH approach. Taking the items that are rated less than twenty times in the Movielens dataset into consideration, the MAE comparison between the IUTH recommendation approach and other benchmark approaches on different numbers of neighbours is shown in Figure 4-13. It can be seen from the results that the accuracy of IUTH is better than the CF-based approaches (DeshpandeCF and SarwarCF) and the hybrid approach (HSR).

2) Coverage metric

The coverage rates of the four recommendation approaches are shown in Table 4-11. Because the sparsity level is not high in the Movielens dataset, all the coverage rates are very high in the results. Because the IUTH approach fully utilizes the semantic information of items and combines the KB recommendation techniques, it is able to recommend all the items.

Table 4-11. Coverage rate of the recommendation algorithms

	DeshpandeCF	SarwarCF	HSR	IUTH
coverage	0.9957	0.9886	1	1

The evaluation results demonstrate that the proposed item tree and user request tree-based hybrid recommendation approach has a good performance. It can utilize the semantic information of tree-structured data effectively in recommender systems, and is well-suited in recommending tree-structured items, such as business partners.

4.6 SUMMARY

This chapter outlined the development of an item tree and user request tree-based hybrid (IUTH) recommendation approach to deal with the tree-structured data in recommender systems. To model tree-structured items and users, an item tree and a user request tree are defined. The similarity measure model developed in Chapter 3 is utilized to evaluate the semantic similarity between item trees or user request trees and to evaluate the matching degree of the item trees to user request trees. In the proposed IUTH recommendation approach, the item similarity combines the semantic similarity and the item-based CF similarity, and the final predicted rating hybridizes both the requirement matching-based prediction and the traditional item-based CF prediction. The proposed recommendation approach deals well with the tree-structured data which commonly appear in many real applications, such as business partner recommendations. The effectiveness of the approach is shown in a case study. Experiments on two datasets have been conducted, and the results show that the proposed recommendation approach has good performance and is well-suited in recommending tree-structured items.

CHAPTER 5
A FUZZY PREFERENCE TREE-BASED RECOMMENDER SYSTEM

5.1 INTRODUCTION

In real situations of recommender system applications, the features of items and user behaviours are often subjective, vague and imprecise (Zenebe & Norcio 2009), and users' preferences to items 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 (Chen & Duh 2008; Yager 2003; Zenebe & Norcio 2009; Zenebe, Zhou & Norcio 2010), and recommendations to customers for the selection of the most suitable items are made with incomplete and uncertain information (Cornelis et al. 2007; Porcel, López-Herrera & Herrera-Viedma 2009). The previous research in recommender systems in handling users' uncertainty preferences and items' fuzzy representations only focuses on the single value or vector representations. However, in many application domains, such as business-to-business (B2B) e-services as mentioned before, items or user profiles are so complex that they can often be presented as tree structures. The single value or vector representations for items or user preferences in previous research are not able to deal with the complicated tree-structured data. The fuzzy preference models proposed in previous research, which are represented as vectors, are not suitable for dealing with the tree-structured data in these applications. Consequently, tree-structured

data modelling for users' fuzzy preferences and relevant tree matching methods are needed.

To solve these challenges in personalization of recommendations – namely, tree-structured items, tree-structured user preferences, and vague values of user preferences, this chapter proposes a fuzzy preference tree definition and its construction method to model fuzzy tree-structured user preferences, and, based on the fuzzy preference tree model and the similarity measure on tree-structured data developed in Chapter 3, this chapter develops an innovative fuzzy preference tree-based recommendation approach.

The remainder of the chapter is organized as follows. Section 5.2 presents the fuzzy tree-structured preference model and defines a fuzzy preference tree. The fuzzy preference tree construction algorithm is proposed in Section 5.3. A fuzzy preference tree-based recommendation approach for tree-structured items is presented in Section 5.4. The approach has been tested using the Australian business data set and MovieLens data set. The experimental evaluations and results are given in Section 5.5. Finally, the summary is given in Section 5.6.

5.2 FUZZY TREE-STRUCTURED PREFERENCE MODEL

This section describes the representation of fuzzy tree-structured user preferences. Users' fuzzy preferences are first modelled by use of fuzzy set techniques; and a fuzzy tree-structured user preference model is then presented by use of the tree-structured data model proposed in Chapter 3.

5.2.1 USERS' FUZZY PREFERENCES

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

Information about user preferences can essentially be obtained in two different ways: extensionally and intentionally (Yager 2003). 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 study, the user preference model covers both kinds of information.

In the practice of recommender systems, a user's preferences are usually complex and vague. It might be difficult to require a 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' preferences.

To express users' preferences, it is assumed that a totally ordered set R={1, 2, …, r} is predefined to represent the crisp values of ratings. Based on the fuzzy set definition (Definition 2-1), 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 $f_{i,uj} \in [0,1]$, $i \in R$, 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 sub-set 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" and also 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.

5.2.2 FUZZY TREE-STRUCTURED USER PREFERENCE

The items considered in this study are presented as tree structures, and the features of items form a hierarchical structure. They are modelled as item trees which are defined in

Section 4.2. A user's preferences concern a set of products/features, and user preference is therefore described as a tree structure which has fuzzy preference values.

5.2.2.1 FUZZY PREFERENCE TREE

To formally describe the tree-structured user preferences, the tree-structured data model defined in Chapter 3 is utilized here. To express the fuzzy preference values, a fuzzy tree-structured data model is defined by introducing the fuzzy features.

Definition 5-1. A fuzzy tree-structured data model is a tree-structured data whose node features, i.e. the node attributes, node values, node concept similarity and value similarity measures, or node weights, are represented as fuzzy sets.

A fuzzy preference tree is then defined as follows.

Definition 5-2: The fuzzy preference tree of a user is a tree-structured data whose node values are the user's fuzzy preferences for the corresponding attributes.

A user's preference is represented as a fuzzy preference tree. Each sub-tree in a user's fuzzy preference tree represents the user's preference for one aspect of the features, and the sub-trees 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 quantity or quality 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 sources used to calculate the value. The *count* is used to incrementally update the user's fuzzy preference tree, as shown below.

5.2.2.2 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.

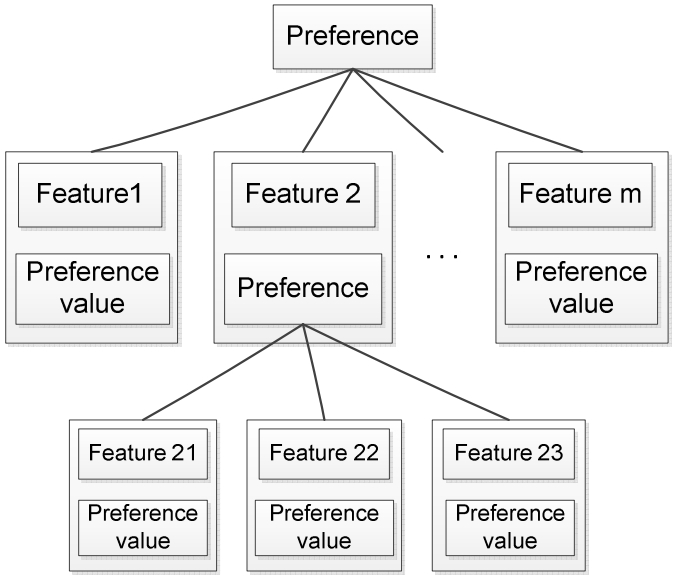


Figure 5-1. Intentionally expressed user preference

Because the item features present tree structures, the preferences given by users are in tree structures, which is shown in Figure 5-1. To express preferences, a user selects several features. For example, *Feature* 1, *Feature* 2, … , *Feature m* are selected in Figure 5-1. For each feature, there are two situations. First, the user can assign a preference value, such as *Feature* 1 in Figure 5-1. 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 Figure 5-1. The tree node attributes are the relevant features. If the user expresses the preference value for a feature, the value will be assigned to the relevant node accordingly. The node weights are also set according to the user's input.

5.2.2.3 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 $EI_u = \{i_1, i_2, \dots, i_m\}$. Each item i_j (j=1, 2, …, m) corresponds to an item tree $T_{item,j}$ and a preference value given by the user $\tilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \dots, f_{r,uj}\}$. 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 $T_{item,j}$ 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 the next section. During the process, the conceptual corresponding parts between two trees considering tree structures, node attributes and weights comprehensively must be identified. Therefore, the conceptual similarity computation algorithm developed in Section 3.3.3 is applied to construct the maximum conceptual similarity tree mapping between two trees.

5.3 A 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 discussed above, the intentionally expressed preference is acquired directly, while the extensionally expressed preference is constructed from the experienced items of the user. The construction algorithm is presented in detail in this section.

5.3.1 ALGORITHM DESCRIPTION

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 is processed by 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. When constructing the mapping, the node weights of both T_u and T_i should be treated equally, i.e. the symmetric similarity measure computation algorithm should be called. Therefore, $M_{u,i}$ is constructed by calling $sc_{T\,sym}(T_u, T_i, M_{u,i})$ illustrated in Algorithm 3-2.

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 sub-tree 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 sub-tree. The sub-tree under n_i is copied and inserted under the new root of T_u. Each leaf of the copied sub-tree 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_p are not identical. In this case, the sub-tree under n_i is copied and inserted under the parent node of n_p. Each leaf of the copied sub-tree 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 $\tilde{p}_{un_p}$ by

$$f_{k,un_p} = (f_{k,un_p} \cdot count + f_{k,ui})/(count + 1),\ k = 1, 2, \ldots, r;$$

$$count = count + 1. \tag{5-1}$$

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 sub-tree under n_i is copied and inserted under n_p. Each leaf of the copied sub-tree 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 sub-tree 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 sub-tree under n_t is copied and taken as tree T'_u. Each leaf of the copied sub-tree 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 5-1, which takes the reference of the fuzzy preference tree as input, and obtains the merged fuzzy preference tree following the operation.

Algorithm 5-1. Fuzzy preference tree merging algorithm

$merge(T_u, n_i, \tilde{p}_{ui}, M_{u,i})$

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

1 if $M_{u,i} = \emptyset$

create a tree node *r*
insert(*r*, T_u)
tree T_{n_i} ←*CopyTree*(n_i)
SetLeafValues(T_{n_i}, $\tilde{p}_{ui}$)
insert(*r*, T_{n_i})
$T_u \leftarrow r$
else
n_p ←*GetMappedNode*(n_i, $M_{u,i}$)
if $n_p \neq null$
if $attributes(n_p) \neq attributes(n_i)$
tree T_{n_i} ←*CopyTree*(n_i)
SetLeafValues(T_{n_i}, $\tilde{p}_{ui}$)
insert(*parent*(n_p), T_{n_i})
else
if n_i has no children
let $n_p.\tilde{p}_u$ be $(f_{1,un_p}, f_{2,un_p}, \ldots, f_{r,un_p})$, $n_p.count$ be c,
$\tilde{p}_{ui}$ be $(f_{1,ui}, f_{2,ui}, \ldots, f_{r,ui})$
$f_{k,un_p} \leftarrow (f_{k,un_p} \cdot c + f_{k,ui})/(c+1)$, k=1, 2, …, r
$n_p.count \leftarrow c + 1$
else
for each child node n_{ic} of n_i
$merge(T_u, n_{ic}, \tilde{p}_{ui}, M_{u,i})$
else
n_p ←*GetMappedNode*(*parent*(n_i), $M_{u,i}$)
if $n_p \neq null$
tree T_{n_i} ←*CopyTree*(n_i)
SetLeafValues(T_{n_i}, $\tilde{p}_{ui}$)
insert(n_p, T_{n_i})
else

29 $n_t \leftarrow$*SearchMappedDescendant*(n_i, $M_{u,i}$)

30 $n_p \leftarrow$*parent*(n_t)

31 *remove*(n_i,n_t)

32 tree $T_{n_i} \leftarrow$*CopyTree*(n_i)

33 *insert*(n_p, T_u)

34 $T_u \leftarrow T_{n_i}$

35 $merge(T_u, n_t, \tilde{p}_{ui}, M_{u,i})$

36 END

In Algorithm 5-1, lines 1-7, lines 11-14, lines 16-21, lines 24-27, and lines 29-35 deal with the five cases of the merge operation respectively. In the algorithm, the procedure *insert*(r, T_{n_i}) inserts T_{n_i} under the tree node r. *CopyTree*(n_i) copies the sub-tree 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 sub-tree 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.

5.3.2 AN EXAMPLE

An example is given in this sub-section to show the construction of a business user's preference tree by use of the algorithm proposed in this study above.

Let T_1 in Figure 5-2 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 Figure 5-2 be the item tree of an item 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). To integrate the user's preferences for T_2

into the user's fuzzy preference tree, the fuzzy preference tree construction algorithm developed above is applied.

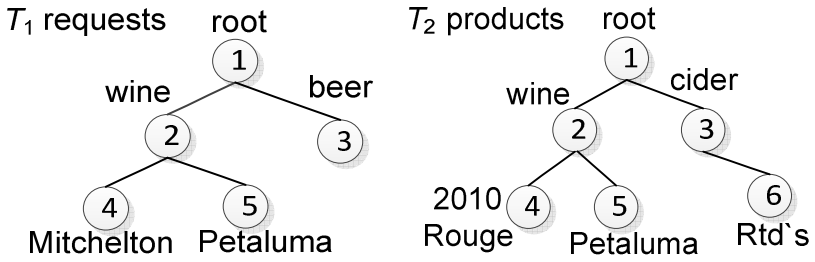


Figure 5-2. Two tree-structured data examples

Step 1: generate the maximum conceptual similarity tree mapping between T_1 and T_2

By calling $sc_{T_{sym}}(T_1, T_2, M_{u,i})$ illustrated in Algorithm 3-2, the maximum conceptual similarity tree mapping between T_1 and T_2, $M_{u,i}$, is constructed, which is shown in Figure 5-3. The matched nodes in the maximum conceptual similarity tree mapping in Figure 5-3 are connected by dashed lines.

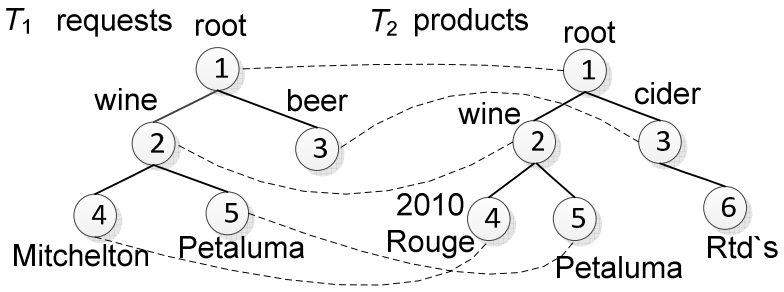


Figure 5-3. The maximum conceptual similarity tree mapping between T_1 and T_2

Step 2: merge T_2 into T_1

Based on $M_{u,i}$, T_2 is merged into T_1 by calling $merge(T_1, t_2[1], \tilde{p}_i, M_{u,i})$ illustrated in Algorithm 5-1. Because $t_2[1]$ is mapped to $t_1[1]$, and $t_2[1]$ and $t_1[1]$ have the same attribute, $t_2[1]$'s child nodes, $t_2[2]$ and $t_2[3]$, are merged recursively. In this way, all the nodes in T_2 are merged into T_1. Some nodes are chosen as examples to show how the preference values are calculated for the fuzzy preference tree nodes. $t_2[4]$ is mapped to

$t_1[4]$, but they have different attributes. According to lines 11-14 in Algorithm 5-1, the sub-tree under $t_2[4]$ is copied and inserted under the parent node of $t_1[4]$. Each leaf of the copied sub-tree is assigned a value whose preference is $\tilde{p}_i$ and *count* is 1. The leaf node $t_2[5]$ is mapped to $t_1[5]$, and they have the same attribute. According to lines 16-18 in Algorithm 5-1, the $\tilde{p}_i$ is integrated into the preference value of node $t_1[5]$ by Formula (5-1): $f_{1,t_1[5]} = 0$, $f_{2,t_1[5]} = 0$, $f_{3,t_1[5]} = (0 + 0.6)/(1 + 1) = 0.3$, $f_{4,t_1[5]} = (0.8 + 1)/(1 + 1) = 0.9$, $f_{4,t_1[5]} = (1 + 0.6)/(1 + 1) = 0.8$.

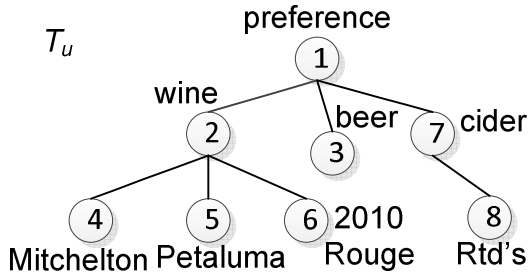


Figure 5-4. The constructed fuzzy preference tree

The final constructed fuzzy preference tree is shown as T_u in Figure 5-4. 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).

5.4 A FUZZY PREFERENCE TREE-BASED RECOMMENDATION APPROACH

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

5.4.1 APPROACH DESCRIPTION

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.

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 $sc_{T\,asym}(T_u, T_i, M_{u,i})$ illustrated in Algorithm 3-2.

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 fuzzy preference value of node $t_u[j]$ in T_u. $v(t_u[j]) = null$ 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 sub-tree $T_u[j]$ makes no contribution to the predicted rating.

$$pr\big(t_u[j], M_{u,i}\big) = 0. \tag{5-2}$$

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 $\tilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \dots, f_{r,uj}\}$.

$$pr\left(t_u[j], M_{u,i}\right) = \begin{cases} 0, & \sum_{k=1}^{r} f_{k,uj} = 0 \\ \sum_{k=1}^{r} k \cdot f_{k,uj} \Big/ \sum_{k=1}^{r} f_{k,uj}, & \sum_{k=1}^{r} f_{k,uj} \neq 0 \end{cases}. \quad (5\text{-}3)$$

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.

$$pr\left(t_u[j], M_{u,i}\right) = \sum_{t_u[j_x] \in mc(t_u[j])} w_x \cdot pr(t_u[j_x], M_{u,i}), \quad (5\text{-}4)$$

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 Case 2 and Case 3. Both the values of node $t_u[j]$ and its children should be considered.

$$pr(t_u[j], M_{u,i}) = \beta_j \cdot pr_{uj} + (1 - \beta_j) \cdot \sum_{t_u[j_x] \in mc(t_u[j])} w_x \cdot pr(t_u[j_x], M_{u,i}), \quad (5\text{-}5)$$

where $pr_{uj} = 0$ if $\sum_{k=1}^{r} f_{k,uj} = 0$; $pr_{uj} = \sum_{k=1}^{r} k \cdot f_{k,uj} / \sum_{k=1}^{r} f_{k,uj}$ if $\sum_{k=1}^{r} f_{k,uj} \neq 0$, $w_x = w(t_u[j_x]) / \sum_{t_u[j_s] \in mc(t_u[j])} w(t_u[j_s])$, 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 5-2.

Algorithm 5-2. Rating prediction algorithm

$pr\left(t_u[j], M_{u,i}\right)$

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 $mc = null$

let the preference value be $\tilde{p}_{uj} = \{f_{1,uj}, f_{2,uj}, \dots, f_{r,uj}\}$

5 if $\sum_{k=1}^{r} f_{k,uj} = 0$

6 return 0;

7 else

8 return $\sum_{k=1}^{r} k \cdot f_{k,uj} / \sum_{k=1}^{r} f_{k,uj}$

9 else if $v(t_u[j]) = null$ and $mc \neq null$

10 return $\sum_{t_u[j_x] \in mc} w_x \cdot pr(t_u[j_x], M_{u,i})$

11 else if $v(t_u[j]) \neq null$ and $mc \neq null$

12 if $\sum_{k=1}^{r} f_{k,uj} \neq 0$

13 return $\beta_j \cdot \left(\sum_{k=1}^{r} k \cdot f_{k,uj} / \sum_{k=1}^{r} f_{k,uj}\right)$

$+(1-\beta_j) \cdot \sum_{t_u[j_x] \in mc} w_x \cdot pr\left(t_u[j_x], M_{u,i}\right)$

14 else

15 return $(1-\beta_j) \cdot \sum_{t_u[j_x] \in mc} w_x \cdot pr\left(t_u[j_x], M_{u,i}\right)$

16 END

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

The proposed recommendation approach overcomes the cold start (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. The approach in this study 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.

5.4.2 AN EXAMPLE

A numerical example is given to show the computation process of the proposed recommendation approach. Let T_u in Figure 5-4 represent the fuzzy preference tree of a user who wants to find supplier partners. T_3 in Figure 5-5 represents an item tree of a wine company. The predicted rating of the user to the wine company is computed by use of the developed fuzzy preference tree-based recommendation approach.

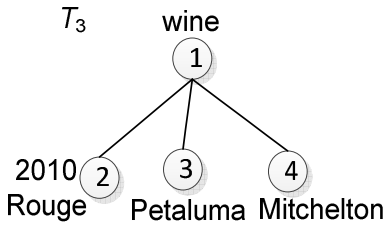


Figure 5-5. An item tree

Step 1: Identifying the corresponding parts of T_u and T_3

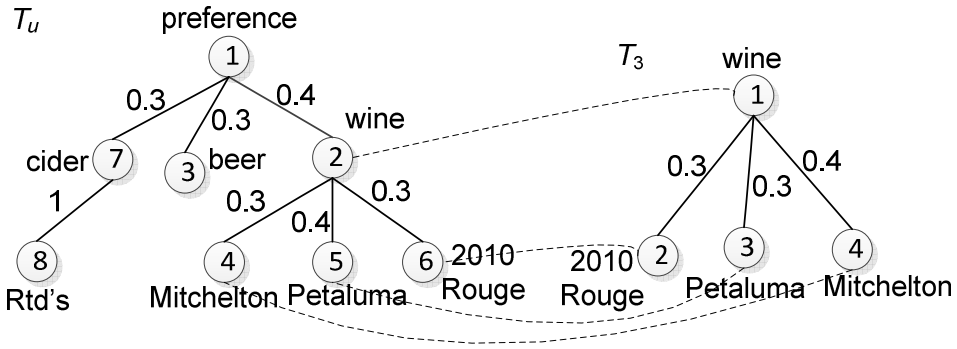


Figure 5-6. The maximum conceptual similarity tree mapping between T_u and T_3

By calling $sc_{T\,asym}(T_u, T_3, M_{u,i})$ illustrated in Algorithm 3-2, the maximum conceptual similarity tree mapping between T_u and T_3, $M_{u,i}$, is constructed, which is

shown in Figure 5-6. The matched nodes in the maximum conceptual similarity tree mapping in Figure 5-6 are connected by dashed lines.

Step 2: Compute the predicted rating to T_3

Based on $M_{u,i}$, the predicted rating to T_3 is computed by calling $pr(t_u[1], M_{u,i})$ illustrated in Algorithm 5-2. Let the numbers beside the edges denote the weights of the relevant child nodes in Figure 5-6. According to Algorithm 5-2, $pr(t_u[1], M_{u,i}) = w_2 \cdot pr(t_u[2], M_{u,i})$, where $w_2 = 1$ and $pr(t_u[2], M_{u,i}) = \sum_{k=4}^{6} w_k \cdot pr(t_u[k], M_{u,i})$. $w_4 = 0.3$, and $pr(t_u[4], M_{u,i}) = (1 \times 0 + 2 \times 0 + 3 \times 0 + 4 \times 0.8 + 5 \times 1)/(0.8 + 1) = 4.56$. In the same way, $w_5 = 0.4$, $pr(t_u[5], M_{u,i}) = 4.25$, $w_6 = 0.3$, $pr(t_u[6], M_{u,i}) = 4$. Thus, the predicted rating to T_3 is finally calculated as 4.27.

5.5 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 data sets used for evaluation, the evaluation metrics and the evaluation results.

5.5.1 EVALUATION DATA SETS

Two data sets, the Australian business dataset and the HetRec 2011 MovieLens data set (http://ir.ii.uam.es/hetrec2011), are used to validate the performance of the proposed recommendation approach. The details of these two data sets are described in Section 4.5.1.

There are 130 business users in the Australian business 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. The products and buying requests of the businesses present tree structures, which are illustrated in Figure 4-1. Because making buyer and supplier recommendations requires different information, the applications for

recommending 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 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.

There are 2113 users in the HetRec 2011 MovieLens data set. In this experiment, the ratings of each user are split into two parts, the training set and the testing set, and the 10 most recent ratings of each user make up the testing set. Each movie is represented as a tree-structured object, which is shown in Figure 4-9. 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 data set are on a scale of 1 to 5. In these experiments, each rating is transformed into a fuzzy sub-set on {1, 2, 3, 4, 5}.

5.5.2 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 against the user ratings for the user-item pairs in the test set. The Mean Absolute Error (MAE), which is described in detail in Section 4.5.2, is the most widely used statistical accuracy metric in recommendation research (Resnick et al. 1994), and is utilized in this experiment.

2) Recall, precision and F1 metrics

Besides MAE, recall, precision and F1-measure are also widely used to evaluate the accuracy of recommender systems. 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.

$$recall = \frac{\left|\{preferred\} \cap \{recommended\}\right|}{\left|\{preferred\}\right|}, \tag{5-6}$$

$$precision = \frac{\left|\{preferred\} \cap \{recommended\}\right|}{\left|\{recommended\}\right|}, \tag{5-7}$$

$$F1 = \frac{2 \times recall \times precision}{recall + precision}. \tag{5-8}$$

5.5.3 BENCHMARK RECOMMENDATION APPROACHES

The application of fuzzy modelling for a content-based recommender system was initially presented in (Yager 2003). A fuzzy set theoretic method (FTM) for recommender systems was proposed in (Zenebe & Norcio 2009). 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 this experiment. A crisp set-based method was also implemented. A fuzzy user preference model was presented in (Zenebe, Zhou & Norcio 2010) and was also implemented in this experiment for comparison.

5.5.4 EVALUATION RESULTS

In the experiments, the MAE, precision, recall and F1 of the proposed approach and the benchmark approaches are recorded. Figures 5-7 to 5-14 show the MAE, precision, recall and F1 comparisons of each recommendation approach with two data sets

respectively. In these figures, the first approach is the proposed fuzzy preference tree-based recommendation approach. Approaches 2 to 5 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 (Zenebe, Zhou & Norcio 2010).

1) Evaluation results on the Australian business data set

Figures 5-7 to 5-10 show the evaluation results on the Australian business data set. It can be seen that the proposed fuzzy preference tree-based 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 reflects business users' preferences, and the proposed recommendation approach is well-suited to deal with the tree-structured fuzzy user preferences in recommender systems.

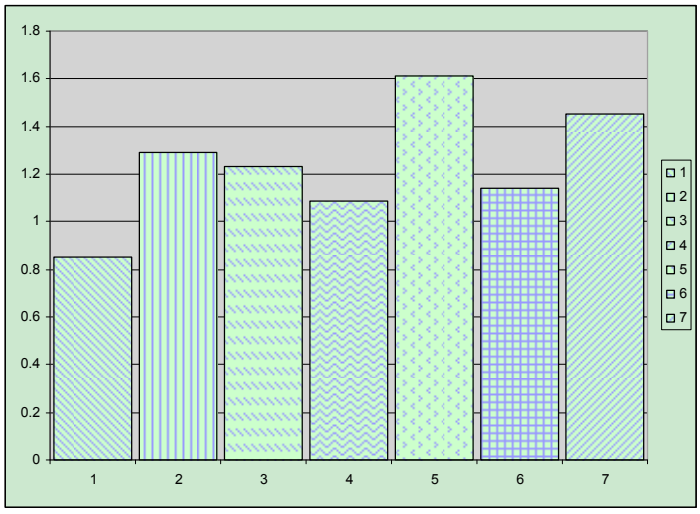


Figure 5-7. The MAE comparison with the Australian business data set

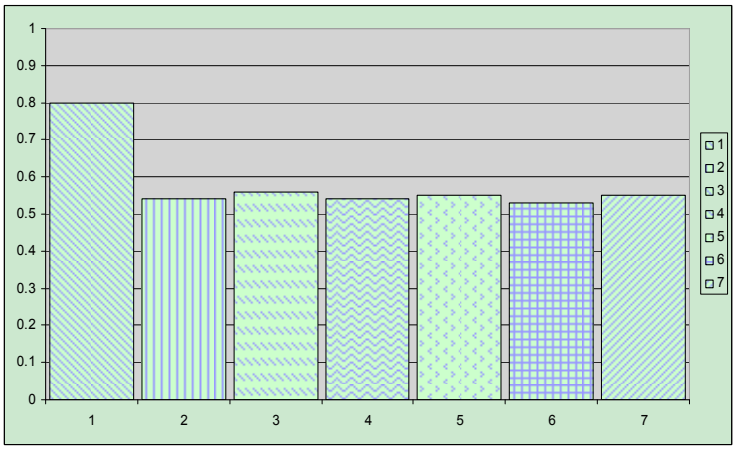


Figure 5-8. The precision comparison with the Australian business data set

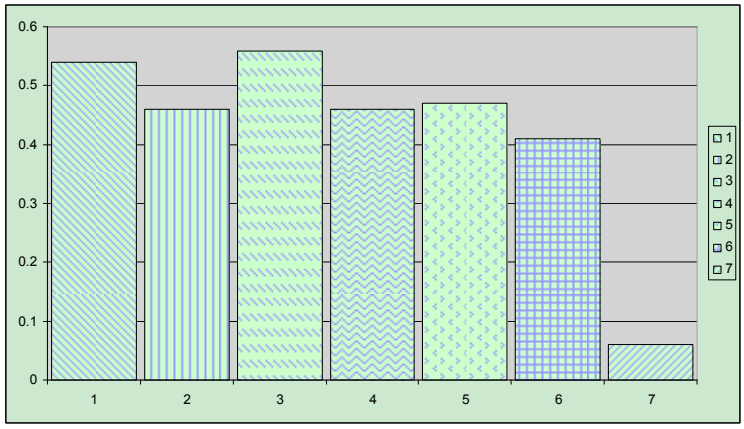


Figure 5-9. The recall comparison with the Australian business data set

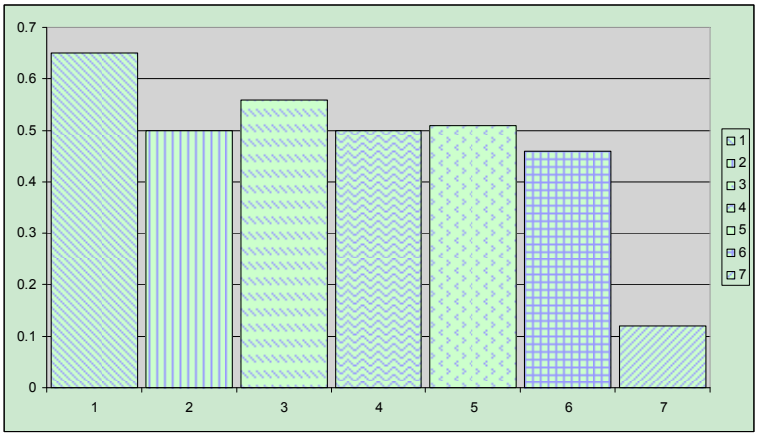


Figure 5-10. The F1 comparison with the Australian business data set

2) Evaluation results on the MovieLens data set

Figures 5-11 to 5-14 show the evaluation results on the MovieLens data set. It can be seen that the proposed fuzzy preference tree-based recommendation approach in this study has the lowest MAE and the highest precision, which indicates the accuracy of this approach. Even though the recall of the proposed approach is a little lower, the F1 measure is comparable with others.

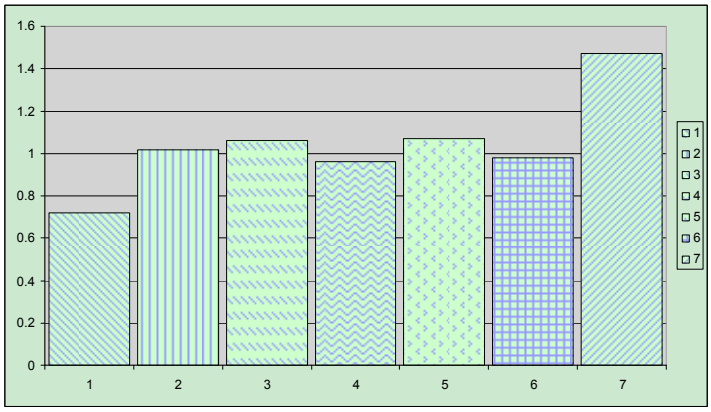


Figure 5-11. The MAE comparison with the MovieLens data set

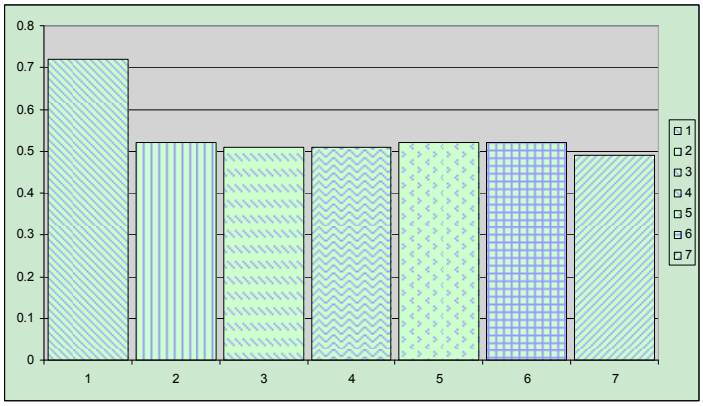


Figure 5-12. The precision comparison with the MovieLens data set

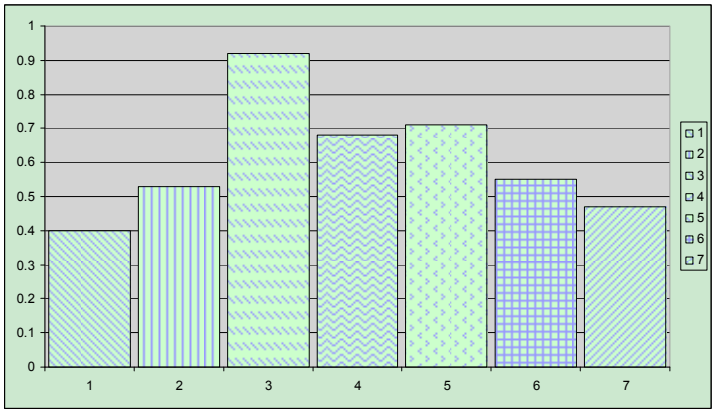


Figure 5-13. The recall comparison with the MovieLens data set

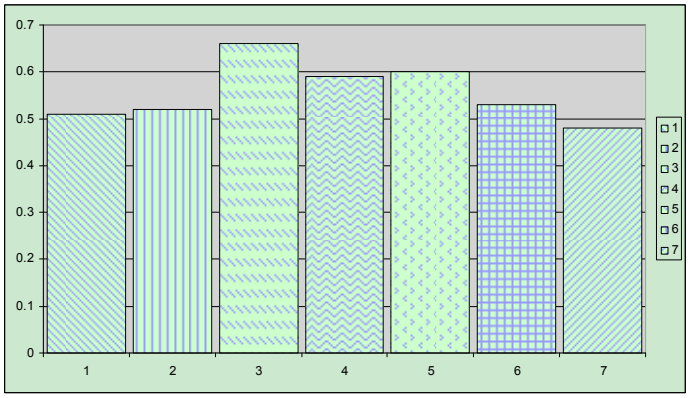


Figure 5-14. The F1 comparison with the MovieLens data set

Experimental results on both data sets show that high recommendation accuracy is obtained by representing the user preferences with this proposed fuzzy tree-structured preference model, especially on the Australian business data set which has tree-structured features. This reflects the effectiveness of the fuzzy tree-structured user preference model and the proposed recommendation approach based on it.

5.6 SUMMARY

This chapter proposes a method for modelling and presenting tree-structured user preferences with uncertainty and develops a new recommendation approach which can recommend tree-structured items. The fuzzy tree-structured user preference modelling 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, the similarity measure on tree-structured data developed in Chapter 3 which comprehensively considers tree structures, node attributes and node weights is applied. Two experiments on an Australian business data set and the MovieLens data set respectively are conducted to evaluate the performance of the proposed recommendation approach. Both results show that the proposed approach makes accurate recommendations and demonstrate that the fuzzy tree-structured user preference profile reflects user preferences effectively. The experiment on the Australian business data set shows that it is well-suited to the business application environment. This approach provides a new solution for improving recommender systems in general and can be used in e-government, e-business, e-learning and other application fields where the data is described in a tree structure.

CHAPTER 6
SMART BIZSEEKER – A BUSINESS PARTNER RECOMMENDER SYSTEM

6.1 INTRODUCTION

Finding qualified business partners (buyers or suppliers) is important for companies that are expanding their business, and is especially important for new companies that are setting up a business. The increasing popularity of the Internet has led to an abundance of information being created and delivered via the web, which provides excellent opportunities for developing business-to-business (B2B) e-services to help businesses find partners (Lu et al. 2010). A number of public online business directories and portals support the searching for and retrieving potential business partners. For example, the Australian Trade Commission government agency, Austrade (www.austrade.gov.au), supports Australian export companies in buyer/partner identification and selection by searching overseas markets for relevant buyers/partners. The United States (US) Commercial Service (www.export.gov) helps the US export companies to find partners in overseas markets through the International Partner Search (IPS) service. However, due to its simplicity, low recall and imprecision, keyword query is not efficient and cannot satisfy users' particular needs (Zhang, Zhu & Huang 2009). In addition, the amount of information available on the web is overwhelming, and searching for qualified business partners therefore requires excessive time and effort, which sometimes turns out to be too costly, unreliable and risky (Bivainis 2006). Therefore, businesses cannot directly use the partner information obtained online.

To solve this problem, this study develops a recommender system – Smart BizSeeker to provide personalized business partner finding e-services. There is a big challenge in the development: business product and buying request data, which are the main attributes of "items" (potential supplier partners supplying products and potential buyer partners having buying requests) and "users" (businesses searching partners) in the business partner recommender system, are often described in complicated tree structures. For example, a business may supply several product categories, and each product category may contain several sub-categories. Under most sub-categories, there are several specific products that the business can supply which form a tree structure. To handle this challenge, the item tree and user request tree-based hybrid (IUTH) recommendation approach proposed in Chapter 4 is applied in the Smart BizSeeker. To deal with users' fuzzy preferences, the fuzzy preference tree-based recommendation approach developed in Chapter 5 is also applied. A case study demonstrates that the developed system can effectively deal with tree-structured business data and users' fuzzy preferences, and recommend suitable business partners to business users.

The remainder of the chapter is organised as follows. The requirements of e-business recommender systems are described in detail in Section 6.2. The system architecture of Smart BizSeeker is presented in Section 6.3 followed by Section 6.4 which describes the implementation of Smart BizSeeker. A case study is then given in Section 6.5 to illustrate the Smart BizSeeker recommender system. Finally, a summary of this chapter is given in Section 6.6.

6.2 REQUIREMENTS OF E-BUSINESS RECOMMENDER SYSTEMS

In comparison with other application domains, the items and user profiles in e-business recommender systems have their special features, which involve special requirements for recommendation approaches and similarity measures.

1) The items and user preferences in e-business recommender systems are usually very complex, which contain abundant of semantic information. To make effective and accurate recommendations, the semantic information of users or items must be used and the semantic similarities need to be developed.

2) The semantic information of items or users in e-business recommender systems is often presented in tree structures. These tree-structured data are different from several aspects, including tree structures, nodes concepts, nodes values and nodes weights.

3) In a real life situation, the preferences of business users are usually vague and uncertain.

4) When making business partner recommendations to a business user, both supplier partners and buyer partners need to be recommended.

This study develops a business partner recommender system which deals with the above special requirements in the e-business environment.

6.3 SYSTEM ARCHITECTURE

In this section, the system architecture of Smart BizSeeker is presented. The architecture of Smart BizSeeker is depicted in Figure 6-1. As a web-based online system, the Smart BizSeeker recommender system has a standard multi-tier architecture, which includes web browser, web server and database server. The web browser is the user interface for users to actively access the system. When a user visits the website of Smart BizSeeker, the web browser will send requests to the web server every time the user performs an action, such as login or visiting a new page. When the web server receives the requests, it retrieves the requested resources and sends them back to the web browser. The web site and application are hosted in the web server, which provide business partner recommendation services and other relevant services. The Smart BizSeeker web site can

be divided into three layers: the presentation layer, business logic layer and data access layer. The databases of the system are maintained in the database server.

This business partner recommender system, as depicted in Figure 6-1, is designed to have two types of users: system administrators and business users. The roles of the users are described as follows.

- *System administrator*. The role of the system administrator is: 1) to maintain the product categories and business categories, which are used to infer the semantic similarity between products or businesses; 2) to manage the recommendation algorithms and related parameters, which are used to support the operation of the system.
- *Business user*. The role of a business user is: 1) to register a business into the system; 2) to manage the profile of the business and input its products and buying requests into the system; 3) to seek and search for potential business partners (buyers and suppliers) and receive recommendations on their suitability; 4) to give feedback and rate the recommended partners.

The main components of the Smart BizSeeker recommender system are described in detail as follows.

6.3.1 DATABASES

The database stores all the data of the recommender system, which includes the following main components:

- *Business profiles*: stores the basic information, such as business name, business category, scale and contact information, of businesses registered in the recommender system.
- *Business products*: stores the products and their detailed features provided by each business.

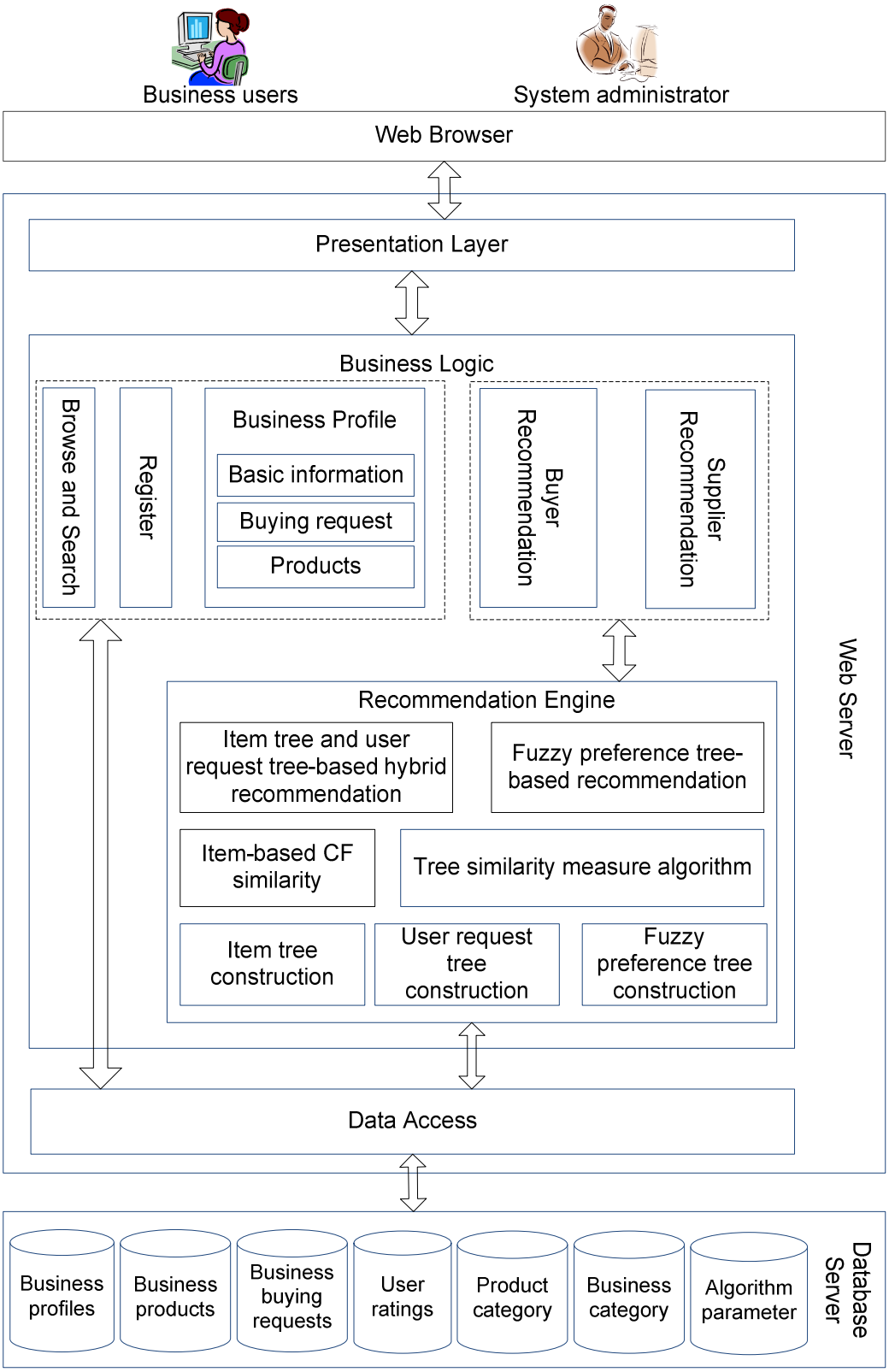


Figure 6-1. The architecture of Smart BizSeeker

- *Business buying requests*: stores the product categories and products required by each business.
- *User ratings*: stores the business users' ratings to their business partners, which contain two types, the ratings to buyer partners and the ratings to supplier partners.
- *Product category*: maintains the hierarchical structured product categories.
- *Business category*: maintains the hierarchical structured business categories.
- *Algorithm parameter*: maintains the parameters used in the recommendation algorithms.

6.3.2 WEB APPLICATION COMPONENTS

The application in the web server contains three layers: the presentation layer, the business logic layer and the data access layer.

1) Presentation layer

This layer is responsible for generating the requested web pages and handling the user interface logics and events. When a user requests to view a new page, the presentation layer will invoke corresponding methods in the business logic layer, extract the request data, transform the data into an HTML page and send it back to the client.

2) Business logic layer

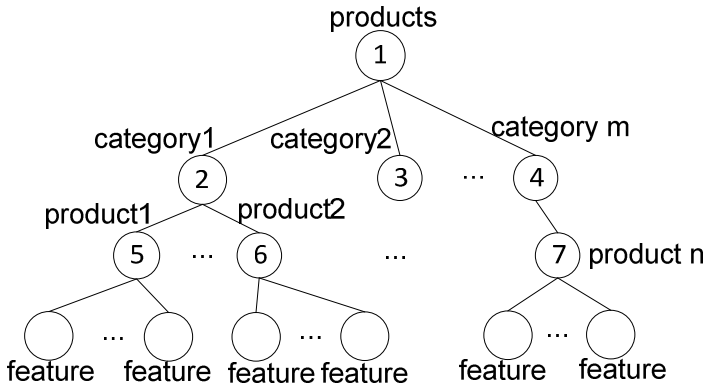
The business logic layer defines the business processes and functions of the application, and serves as a mediator between the presentation layer and the recommendation engine and the data access layer. It provides the following main functionalities as follows:

- *Browse and search*: This function concerns the user's ability to browse the potential business partners' profiles and their products/buying requests, or search

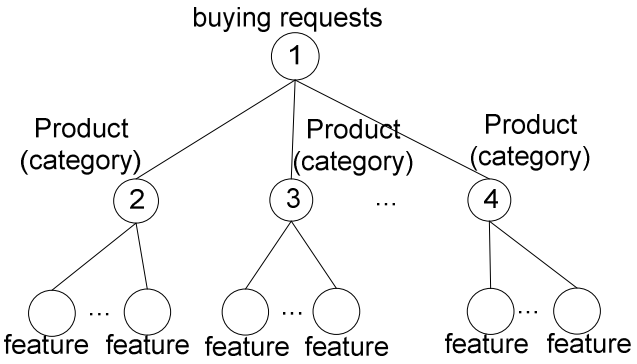
potential business partners and products/buying requests based on the keyword-based search engine, and to view the matching results.

- *Register*: This function supports new business users to register into the system.
- *Business profile management*: The business profile management module collects the basic information, the supplied product information, and the buying request information of the business user. Users can modify, add and delete these kinds of information through this module.
- *Buyer recommendation*: The module calls the recommendation engine to generate buyer partner recommendations to the business user and passes the recommendation list to the user. Users' comments and ratings to the buyer partner recommendation results are also collected through the module.
- *Supplier recommendation*: The module calls the recommendation engine to generate supplier partner recommendations to the business user and passes the recommendation list to the user. Users' comments and ratings to the supplier partner recommendation results are also collected through the module.
- *Recommendation engine*: The recommendation engine implements the proposed item tree and user request tree-based hybrid (IUTH) recommendation approach in Chapter 4 and the fuzzy preference tree-based recommendation approach in Chapter 5. It generates a recommendation list of potential business partners, buyers or suppliers, according to the user's requirement. The following main components in the recommendation engine are described as follows:
  - *Tree similarity measure algorithm*: This component implements the tree similarity measure computation algorithms developed in Section 3.3. It provides not only the interface to calculate the similarity measure between two tree-structured data, but also the interface to generate the maximum conceptual similarity tree mapping between tree-structured data.

– *Item-based CF similarity*: This component computes the item-based CF similarity between two items based on the user-item rating matrix, which is used in Step 3 in the IUTH recommendation approach proposed in Section 4.3.



(a)



(b)

Figure 6-2. The product tree structure (a) and the buying request tree structure (b) in Smart BizSeeker

– *Item tree construction*: The products or buying requests of a business construct tree structures, which are depicted in Figure 6-2. To handle these tree-structured data comprehensively, the item tree and user request tree definitions proposed in Section 4.2 are applied. This component constructs item trees that are illustrated in Section 4.2. The tree structures are in accordance with that in Figure 6-2. The tree node attributes and values are

assigned according to the definition in Section 4.2. When a user wants to get recommendations of supplier partners, the potential suppliers' product trees will be constructed as the item trees. When a user wants to get recommendations of buyer partners, the potential buyers' buying request trees will be constructed as the item trees.

- *User request tree construction*: This component constructs user request trees that are illustrated in Section 4.2. The tree structures are in accordance with that in Figure 6-2. The tree node attributes and values are assigned according to the definition in Section 4.2. When a user wants to get recommendations of supplier partners, the user's buying request tree will be constructed as the user request tree. When a user wants to get recommendations of buyer partners, the user's product tree will be constructed as the user request tree.

- *Fuzzy preference tree construction*: This component implements the fuzzy preference tree construction algorithm proposed in Section 5.3 and constructs the fuzzy preference tree for the user. In Smart BizSeeker, the users' ratings are on a scale of 1 to 5, with 1 being the lowest rating and 5 the highest rating, a user's fuzzy preference value of a fuzzy preference tree node is then represented as a fuzzy sub-set on {1, 2, 3, 4, 5} by membership degree [0, 1]. When recommending suppliers, the user's fuzzy preference tree is constructed by merging the user's buying request and the product trees of businesses rated by the user. When recommending buyers, the user's fuzzy preference tree is constructed by merging the user's product tree and the buying requirement trees of businesses rated by the user.

- *Item tree and user request tree-based hybrid recommendation*: This component implements the proposed recommendation approach in Section 4.3.

– *Fuzzy preference tree-based recommendation*: This component implements the proposed recommendation approach in Section 5.4.

3) Data access layer

The data access layer provides the interfaces to access the data in the database. It deals with the data operations of the database and transfers data with the business logic layer.

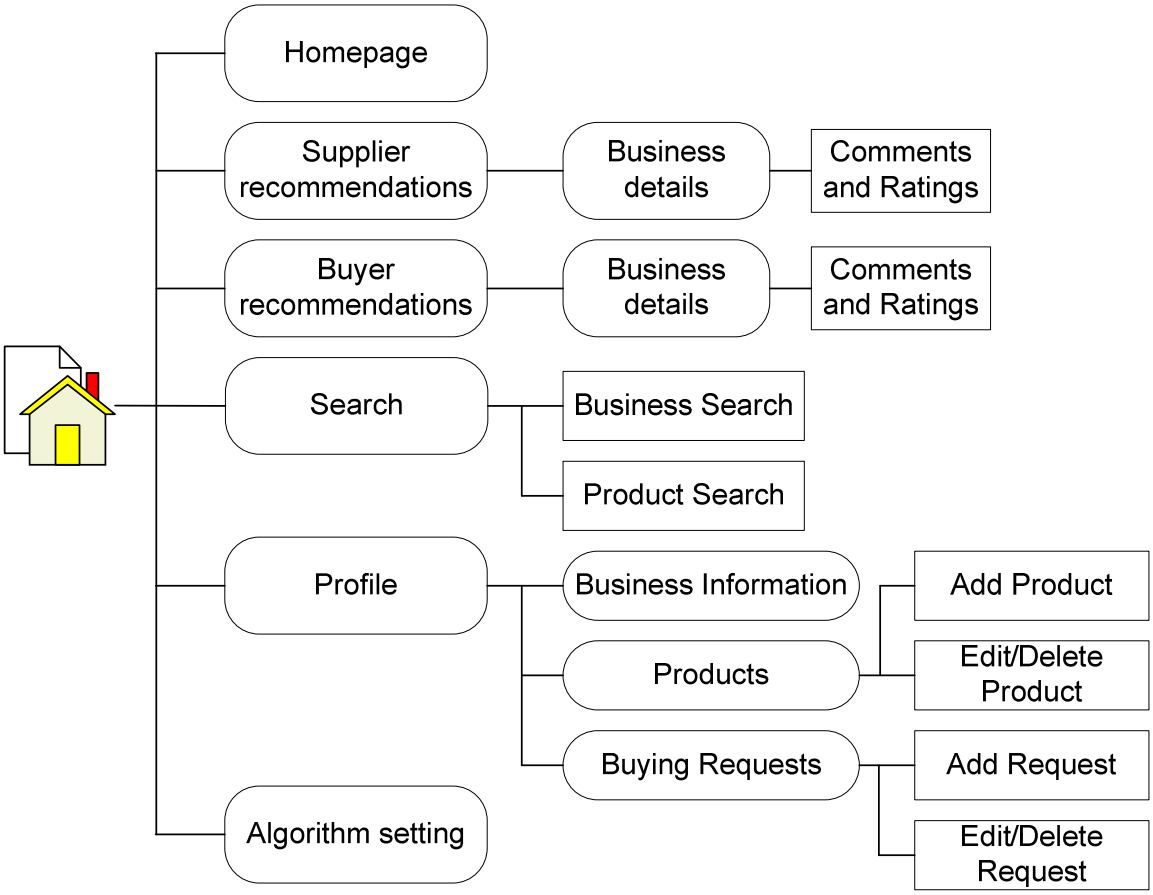


Figure 6-3. The Smart BizSeeker site map

6.4 SYSTEM IMPLEMENTATION

The system is developed and implemented 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 respectively. The site map of Smart BizSeeker is shown in Figure 6-3. All the functionalities described in Section 6.2.2 and Section 6.2.3 are implemented.

The database is designed and implemented in the PostgreSQL database server. Tables are designed and created to store the entities described in Section 6.2.1. To evaluate the semantic similarity between products effectively, the product category must be maintained in the system. To infer the semantic relations between businesses, the business category must be provided. In the Smart BizSeeker recommender system, the business category follows the existing categorization by Austrade (www.austrade.gov.au), depicted in Table 6-1. A two-level product category is also constructed according to the classification of industry classes by Austrade. Part of the product category is shown in Table 6-2.

Table 6-1. Business categories in Smart BizSeeker

No	Business Category
1	Agribusiness
2	Arts & Recreation
3	Building & Construction
4	Business & Other Services
5	Consumer Goods, Non-Food
6	Defence, Security & Safety
7	Education & Training
8	Environment & Energy
9	Finance & Insurance
10	Food & Beverage
11	Government

12	Health, Biotechnology & Wellbeing
13	ICT
14	Manufacturing (Other)
15	Mining
16	Transport
17	Tourism & Hospitality

Table 6-2. Product categories in Smart BizSeeker

Product Category	**Product subcategory**	**Product Category**	**Product subcategory**
Food	Pastry	Drink	Wine
	Roll		Beer
	Sandwich		Spirit
	Pizza		Cider
	Quiche		Soft drink
	Muffin		Juice
	Cake		Coffee
	Tart		Milk
	Cookie	Fruit & Vegetable	Apple
	Noodles		Pear
	Bread		Grape
	Pie		Vegetable
Meat	Beef		Orange
	Chicken		Banana
	Lamb		Avocado
	Sausage		Watermelon
	Pork	…	…

To test the recommender system, it is deployed in the Glassfish web server. Figure 6-4 shows the login page of Smart BizSeeker.

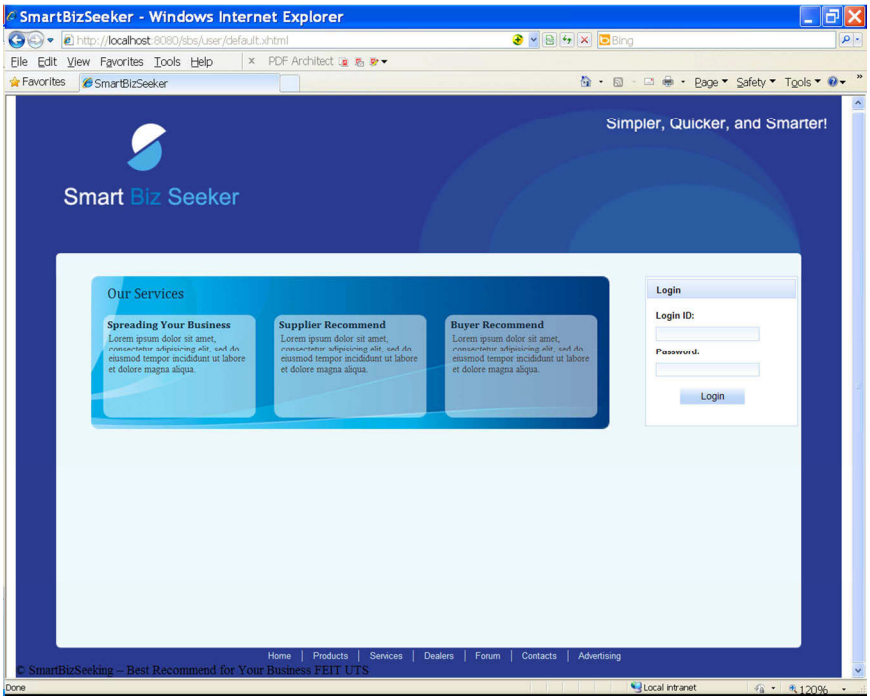


Figure 6-4. The login page of Smart BizSeeker

6.5 A CASE STUDY

This section demonstrates a case study to illustrate the Smart BizSeeker recommender system. In this case, Jack, a business administrator for a wine company called "Laurance Wines Company" would like to find buyer partners and material suppliers by use of the Smart BizSeeker recommender system.

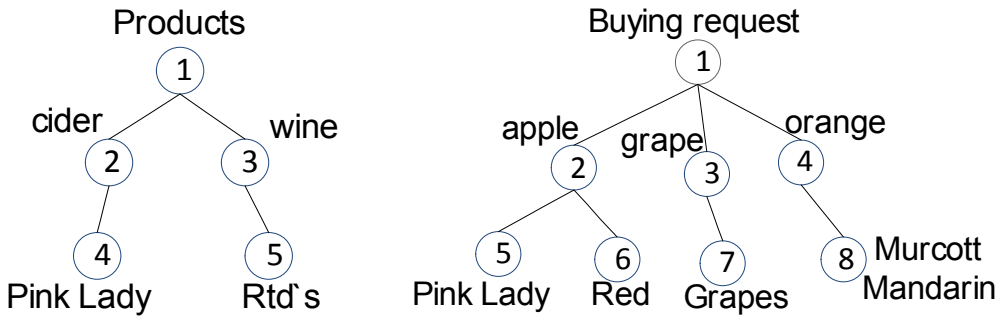


Figure 6-5. The product tree and buying request tree

The company produces ciders and red wines, and needs apples, grapes and some other fruits as materials. The product tree and buying request trees are illustrated in

Figure 6-5. After login to the system, Jack can manage the business profile of his company through the profile management module, as shown in Figure 6-6.

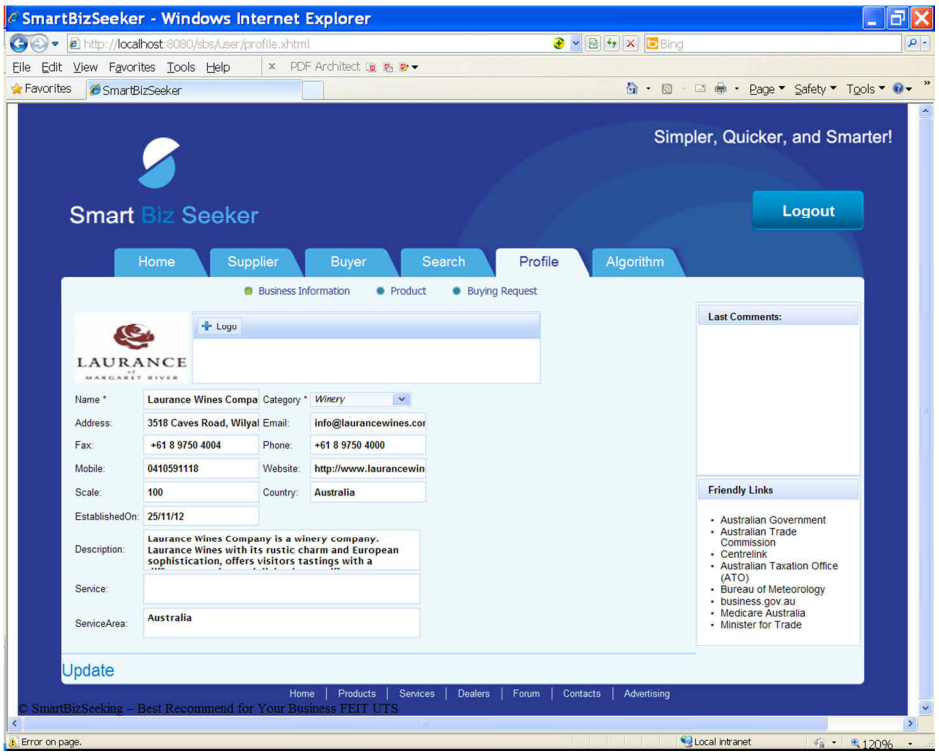


Figure 6-6. Profile management page

The products produced by the company can be input into the system through the product management page, as shown in Figure 6-7. Jack can specify the buying requests for these products or product categories if he wants to find suitable supplier partners, as shown in Figure 6-8. The input information of a user can help the user get partner recommendations. At the same time, the information is also used to make the user be recommended to other users, which is very helpful to expand the user's businesses. In the supplier recommendation page, a list of fruit providers is recommended, as shown in Figure 6-9. In the buyer recommendation page, a list of bottle shops and hotels that want to buy his products are recommended to the user, as shown in Figure 6-10. Users can write comments or give ratings to the recommended partner, as shown in Figure 6-11.

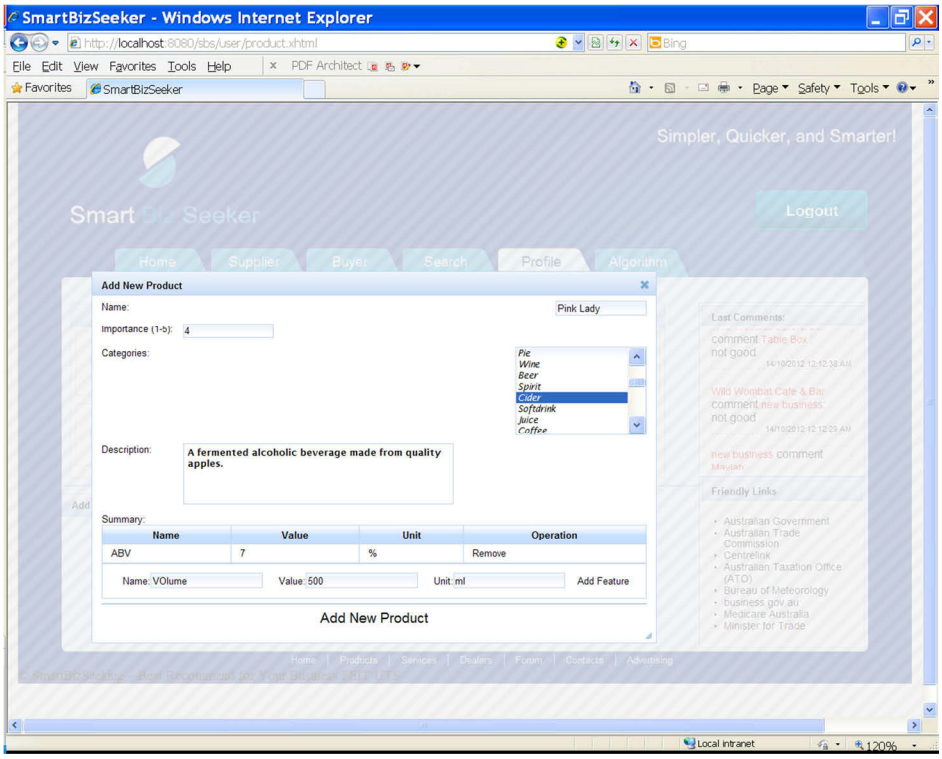


Figure 6-7. Product management page

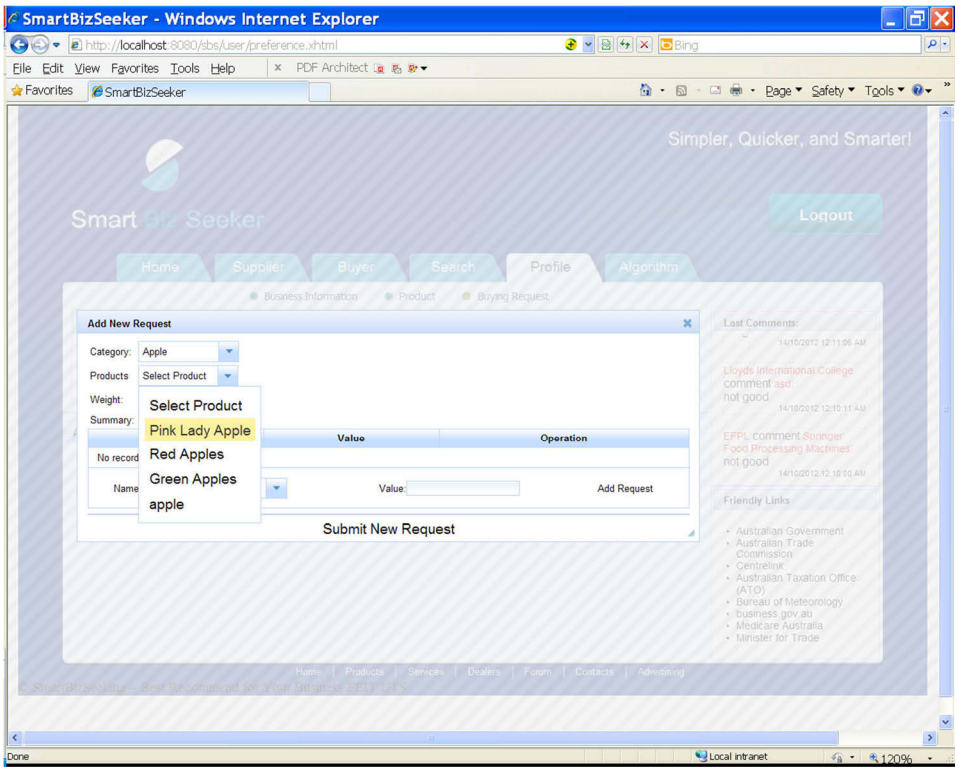


Figure 6-8. Buying request management page

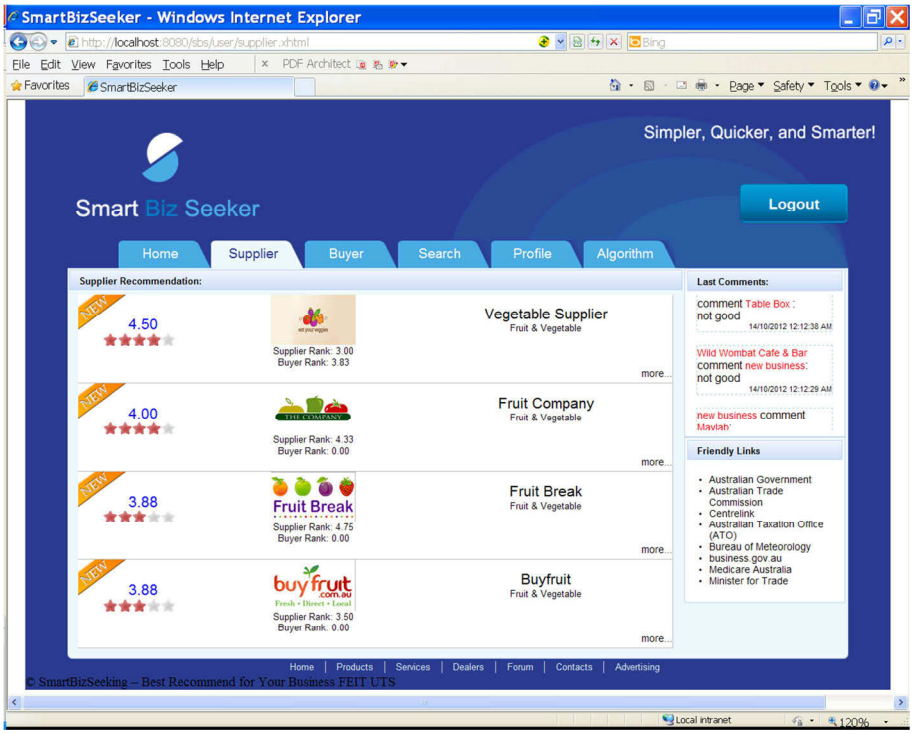


Figure 6-9. The supplier recommendation results

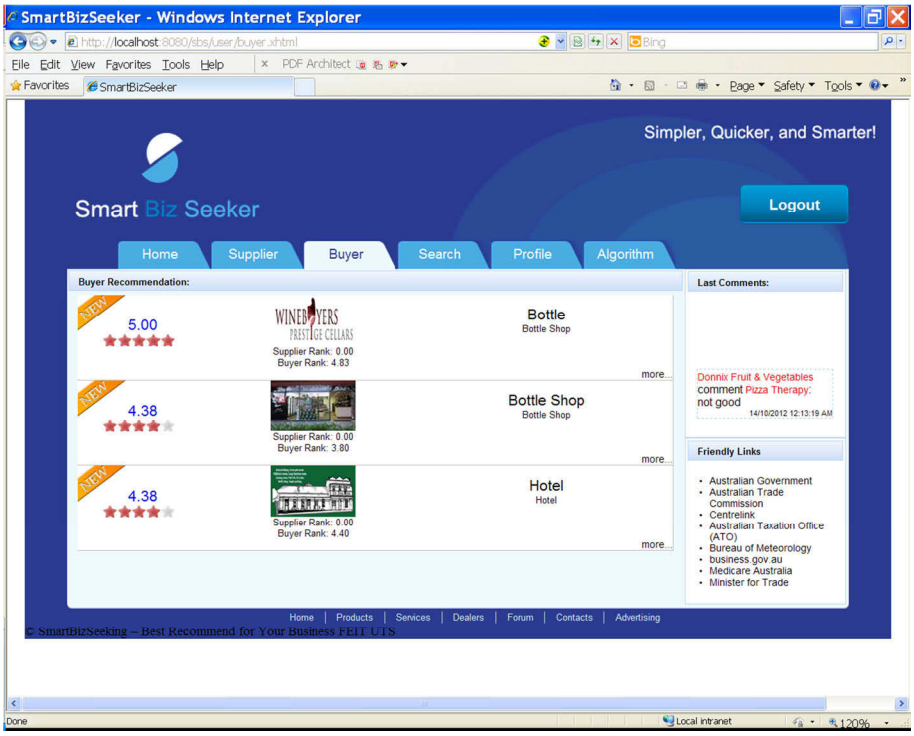


Figure 6-10. The buyer recommendation results

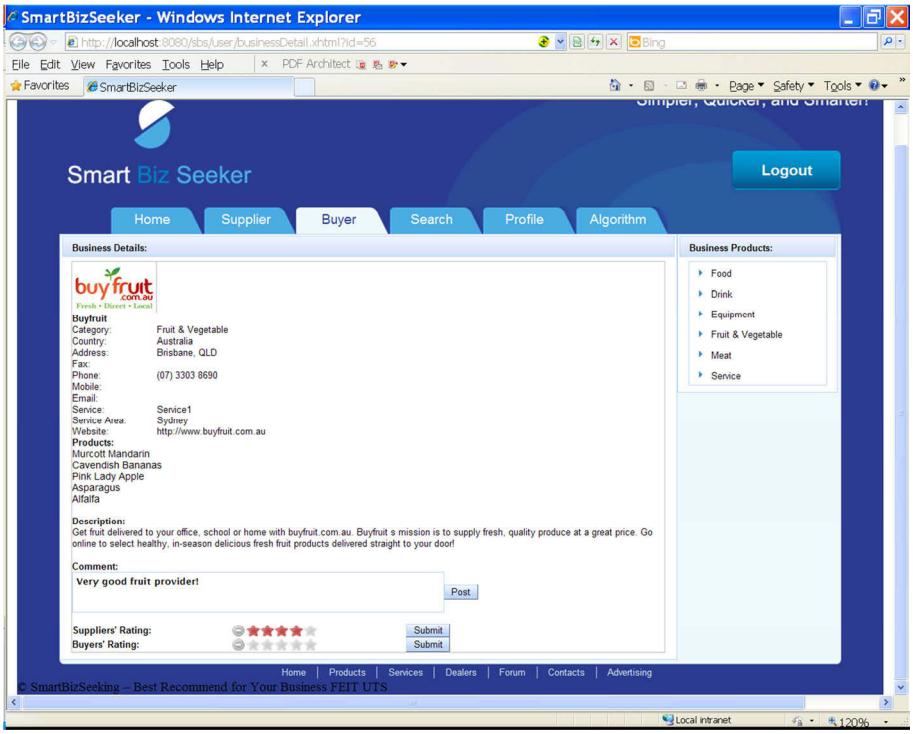


Figure 6-11. Comment and rating page

6.6 SUMMARY

This chapter develops a business partner recommender system, called Smart BizSeeker, which aims to assist business users to effectively select the right business partners (buyers and suppliers) that match their personal business needs and preferences. The proposed Smart BizSeeker recommender system utilizes the IUTH recommendation approach developed in Chapter 4 and the fuzzy preference tree-based recommendation approach developed in Chapter 5 to deal with the tree-structured data and users' fuzzy preferences in business partner recommendation applications. A case study demonstrates the usage of the system.

CHAPTER 7
ELRS – AN E-LEARNING RECOMMENDER SYSTEM

7.1 INTRODUCTION

E-learning systems are becoming increasingly popular in educational establishments due to the development of web-based information and communication technologies. The rapid growth of e-learning systems has changed traditional learning behaviour and presents a new situation to learners (students), which greatly supports and enhances learning practices online. Learning activities in e-learning systems, which are called the units of learning (Drachsler, Hummel & Koper 2008b), can be subjects, learning materials, resources and so on. Due to the emergence of numerous kinds of learning activities in the e-learning environment, learners find it difficult to select the learning activities that best meet their criteria. It is imperative for an e-learning system to automatically generate personalised recommendations to guide a learner's activities (Zaıane & Luo 2001), and as demonstrated by Lu (2004), an e-learning recommender system is necessary to make personalised recommendations. The motivation of this chapter is to develop a recommender system to support learners in the selection of the most appropriate learning activities in an e-learning environment.

E-learning recommender systems need to be able to handle certain special requirements: 1) leaning activities and learners' profiles often present tree structures; 2) learning activities contain vague and uncertain data, such as the uncertain categories that the learning activities belong to; 3) there are pedagogical issues, such as the precedence

relations between learning activities. To deal with the above special requirements in e-learning recommender systems, the tree similarity measure proposed in Chapter 3 and the recommendation approach developed in Chapter 4 are applied, and improved taking the specific requirements of e-learning systems into consideration. The learning activities (items) and learners (users) are modelled as learning activity (item) trees and learner profile (user) trees respectively. To handle the uncertain issues in the data, fuzzy set techniques are applied. The methodology of the IUTH recommendation approach in Chapter 4 is then adopted. To effectively recommend different or new learning activities to learners, the CF recommendation part of IUTH recommendation approach employs the user-based CF technique in this system.

The remainder of the chapter is organised as follows. The special requirements of e-learning recommender systems are described in detail in Section 7.2. The learning activity tree and learner profile tree models are presented in Sections 7.3 and 7.4, respectively. The modified IUTH recommendation approach for learning activities is presented in Section 7.5. The system architecture of ELRS is presented in Section 7.6 followed by Section 7.7 which describes the implementation of ELRS. A case study is then given in Section 7.8 to illustrate the e-learning recommender system. Finally, a summary of this chapter is given in Section 7.9.

7.2 REQUIREMENTS OF E-LEARNING RECOMMENDER SYSTEMS

E-learning systems can be divided into two types according to their application environments: a formal setting and an informal setting (Salehi & Kamalabadi 2013). A formal setting e-learning system includes learning offers from educational institutions (e.g. universities, schools) within a curriculum or syllabus framework. An informal setting is described in the literature as a learning phase of so-called lifelong learners who are not participating in any formal learning and are responsible for their own learning pace and path (Salehi & Kmalabadi 2012). The learning process depends, to a large extent, on individual preferences or choices, and is often self-directed (Maâtallah &

Seridi 2012). Different to the formal setting, the informal setting may provide numerous learning activities from different providers, where learners are also from different backgrounds. There is not usually a curriculum or syllabus framework. Therefore, it is very difficult for students to choose proper learning activities in the informal setting. This can cause high drop-out rates and low completion rates (Clarke 2013; Watters 2013). This study focuses on supporting learners in the informal setting.

In comparison with other application domains, e-learning activities have their special features and demands (Drachsler, Hummel & Koper 2008b), which involve special requirements for recommendation approaches and similarity measures.

1) Both learning activities and learner profiles have complex descriptions and features. A learning activity contains several aspects of information, such as the content description, lecture information and prerequisite information, while a learner profile contains the learner's background, learning goals, prior knowledge, learner characteristics and so on. Each aspect of information can be described in detail with several sub-aspects. Thus, the data in the e-learning environment presents a tree structure.

2) In a real life situation, learning activities and learner profiles always contain vague and uncertain data. One learning activity may be under several categories with different degrees. For example, the subject *Business Intelligence* is mainly in information technology area but also involves business. A learner's requirements are usually described in linguistic terms such as "highly required" or "very important".

3) The pedagogical issues must be considered in the learning activity recommendation. Some learning activities require prerequisite courses. For example, studying the subject *Data Mining* requires the pre-knowledge about database and algorithms. Additionally, learners always want to learn something new or with higher (more advanced) difficulty levels, so these types of precedence relations among learning activities must be considered.

4) It is not feasible to differentiate between two learning activities just from their IDs or names, because learning activities provided from different schools may have different

names, such as one subject is called *Java* and another is called *Program Fundamental*, but the same or similar content.

This study develops an e-learning recommender system to deal with the above special requirements in the e-learning environment.

7.3 LEARNING ACTIVITY TREE

In this section, a learning activity tree is defined based on the fuzzy tree-structured data model (Definition 5-1) to describe the learning activities in the e-learning recommender system. The structure of a learning activity tree is illustrated in Figure 7-1. A learning activity is described from several perspectives, such as the prerequisite courses, the categories, the content and the lecture, and some features may be described from several sub-features, which construct a tree structure.

7.3.1 NODE CONCEPT OF THE LEARNING ACTIVITY TREE

To represent the concept of a node in the learning activity tree, each node is assigned a *label* attribute, as shown in Figure 7-1. Some nodes are assigned a *category* attribute. The node concept similarity is calculated based on the two attributes. If two nodes are both assigned *category*, the category similarity will be taken as the node concept similarity. Otherwise, their labels are compared.

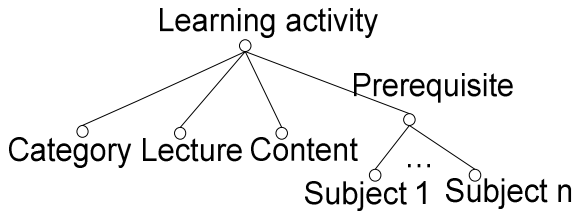


Figure 7-1. The structure of a learning activity tree

The *category* of a learning activity is an important attribute to infer the semantic relations between different learning activities. In a real life situation, one learning activity may belong to several categories with different degrees. Therefore, the value of a

category is a fuzzy category tree in our system. The fuzzy category trees and their similarity measure are presented in detail as follows.

7.3.2 FUZZY CATEGORY TREE AND THE FUZZY CATEGORY SIMILARITY

The fuzzy category tree definition and similarity measure are presented in this sub-section.

7.3.2.1 FUZZY CATEGORY TREE DEFINITION

To divide the learning activities, a learning activity category is usually introduced in e-learning systems. The learning activity category defined in our system is shown in Figure 7-2. It has three levels, which construct a tree structure. There are six general categories, which are "IT/Computer Science", "Nature Science", "Humanities/Social Sciences", "Business", "Engineering/Technology", and "Medicine/Health". Each general category is divided into several sub-categories. For example, the "IT/Computer Science" category can be divided into four sub-categories, which are "Internet", "Software", "Hardware", and "Business Intelligence".

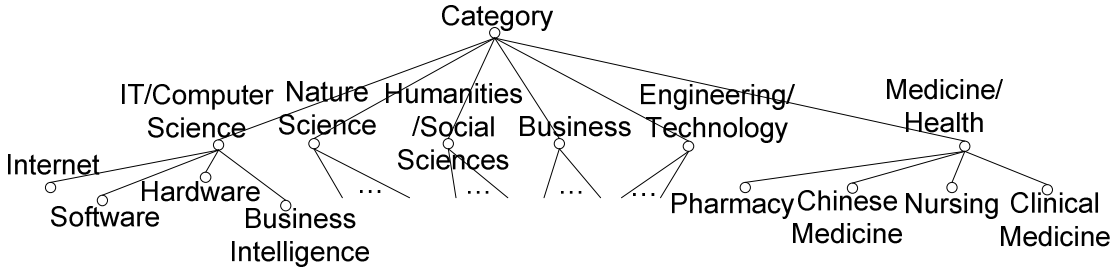


Figure 7-2. The learning activity category tree

In real applications, each learning activity may belong to several categories with different degrees. For example, the subject *Business Intelligence* is under the categories "Business Intelligence", "Software", "Marketing", and "Management" with different membership degrees, as shown in Figure 7-3 (a), in which the number under each sub-

category represents the membership degree of the subject that belongs to the sub-category. The sub-categories and corresponding membership degrees are specified by the learning activity providers when they insert the learning activities into the system. To represent the categories of a learning activity, a fuzzy category tree is defined.

Definition 7-1. A fuzzy category tree of a learning activity represents the categories the learning activity belongs to, which is a sub-tree of the learning activity category tree. The nodes of the fuzzy category tree are assigned category values, which represent the membership degrees of the learning activity belonging to the relevant sub-categories.

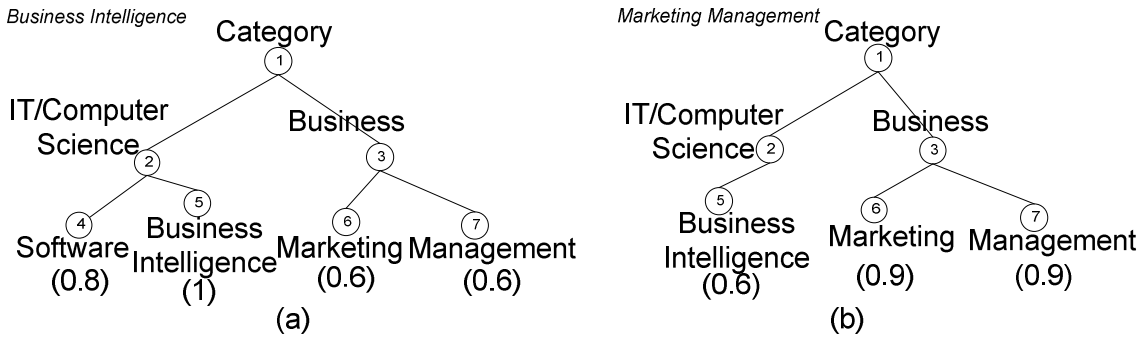


Figure 7-3. The fuzzy category trees of two learning activities: (a) is the fuzzy category tree of the subject *Business Intelligence*. (b) is the fuzzy category tree of the subject *Marketing Management*.

Two examples of fuzzy category tree are shown in Figure 7-3.

Let $v_c(t[i])$ represent the category value of node $t[i]$. If a learning activity does not belong to the sub-category represented by node $t[i]$, $v_c(t[i]) = 0$. The category value of $T[i]$, the sub-tree under the node $t[i]$, can be inferred from the category values of nodes in the sub-tree $T[i]$, which is calculated by Formula (7-1).

$$v_c(T[i]) = \begin{cases} v_c(t[i]), & F[i] = \emptyset \\ \left(\bigvee_{j=1}^{n_i} v_c\left(T[i_j]\right)\right) \vee v_c(t[i]), & F[i] \neq \emptyset \end{cases} \quad (7\text{-}1)$$

Similarly, the category value of the forest $F[i]$ can be defined, and calculated by

$$v_c(F[i]) = \begin{cases} 0, & F[i] = \emptyset \\ \left(\bigvee_{j=1}^{n_i} v_c(T[i_j])\right), & F[i] \neq \emptyset \end{cases} \quad (7\text{-}2)$$

The category value of the sub-tree $T[i]$ or the forest $F[i]$ will be 0, if the learning activity is not relevant to the categories under the sub-tree $T[i]$ or the forest $F[i]$.

7.3.2.2 FUZZY CATEGORY SIMILARITY

The similarity measure between categories is necessary to evaluate the semantic similarity between learning activities, which is vital to make recommendations. Because the category for each learning activity is represented as a fuzzy category tree, the traditional node distance based method cannot be used. A fuzzy category tree similarity measure is developed in this sub-section.

As the fuzzy category trees are all based on the learning activity category tree shown in Figure 7-2, the numbering of the learning activity category tree is used to represent tree nodes. Let $T_1[i]$ and $T_2[i]$ represent two fuzzy category trees of two learning activities a_1 and a_2, respectively. To evaluate the similarity between two fuzzy category trees, the values of all nodes must be taken into account. According to the fuzzy category tree definition, four properties of the fuzzy category trees can be discovered: 1) the structures of $T_1[i]$ and $T_2[i]$ are the same as they are based on the same category tree; 2) only the sub-trees with positive category values need to be considered when calculating the similarity as the sub-trees with zero category values are not relevant; 3) the category values may be assigned to nodes at different levels; 4) category values in different levels present different weights. When calculating the similarity between two category trees, all these properties must be considered. According to the conditions whether the children of $t_1[i]$ and $t_2[i]$ are assigned positive values or zero, four situations are considered in the similarity measure formula. The fuzzy category similarity between $T_1[i]$ and $T_2[i]$ is calculated as:

$$s_{fc}(T_1[i], T_2[i]) =$$

$$\begin{cases} v_c(t_1[i]) \wedge v_c(t_2[i]), & v_c(F_1[i]) = 0, v_c(F_2[i]) = 0 \\ v_c(t_1[i]) \wedge v_c(T_2[i]), & v_c(F_1[i]) = 0, v_c(F_2[i]) \neq 0 \\ v_c(T_1[i]) \wedge v_c(t_2[i]), & v_c(F_1[i]) \neq 0, v_c(F_2[i]) = 0 \\ (\alpha^{h-d_i} - \alpha^h) \cdot (v_c(T_1[i]) \wedge v_c(T_2[i])) & \\ + (1 - \alpha^{h-d_i} + \alpha^h) \cdot (\bigvee_{j=1}^{n_i} s_{fc}(T_1[i_j], T_2[i_j])), & v_c(F_1[i]) \neq 0, v_c(F_2[i]) \neq 0 \end{cases}, \quad (7\text{-}3)$$

where α is the influence factor of the parent node, h is the height of the learning category tree, and d_i is the depth of node i in the category tree. In the first situation, $v_c(F_1[i]) = 0$ and $v_c(F_2[i]) = 0$, which means that $t_1[i]$ and $t_2[i]$ have no children nodes or their children nodes are not assigned positive values. Therefore, only the values of $t_1[i]$ and $t_2[i]$ are considered. In the second situation, $t_1[i]$ has no children or its children nodes are not assigned positive values. Thus, the two trees $T_1[i]$ and $T_2[i]$ can only be compared at the level of $t_1[i]$. The third situation is similar to the second one. In the fourth situation, the children of both $t_1[i]$ and $t_2[i]$ are assigned positive values. Therefore, the lower levels of $t_1[i]$ and $t_2[i]$ should also be compared. As the categories in the lower level are more specific, the lower level should gain more weight in the similarity measure. The coefficient α^{h-d_i} in Formula (7-3) reflects the point. To guarantee that the similarity between different general categories be 0, α^h is subtracted from α^{h-d_i} in the formula.

Take two subjects, *Business Intelligence* and *Marketing Management*, which are illustrated in Figure 7-3, as examples. Let α be 0.5. In the example, h = 2. $s_{fc}(T_a[4], T_b[4]) = 0$; $s_{fc}(T_a[5], T_b[5]) = 0.6$; $v_c(T_a[2]) = v_c(T_a[4]) \vee v_c(T_a[5]) = 1$, $v_c(T_b[2]) = 0.6$, $s_{fc}(T_a[2], T_b[2]) = (\alpha^{h-1} - \alpha^h) \cdot \left(v_c(T_a[2]) \wedge v_c(T_b[2])\right) + (1 - \alpha^{h-1} + \alpha^h) \cdot \left(s_{fc}(T_a[4], T_b[4]) \vee s_{fc}(T_a[5], T_b[5])\right) = 0.6$; similarly, $s_{fc}(T_a[3], T_b[3]) = 0.6$; the fuzzy category similarity between these two subjects is calculated as $s_{fc}(T_a[1], T_b[1]) = s_{fc}(T_a[2], T_b[2]) \vee s_{fc}(T_a[3], T_b[3]) = 0.6$.

7.3.2.3 FUZZY CATEGORY TREE COMBINATION

In practice, there are times when the fuzzy category trees need to be combined. For example, a learner has completed several learning activities. To examine the categories learned by the learner comprehensively, the categories of all the learning activities learned by the user should be combined. A fuzzy category tree combination procedure $combine(\cdot)$ is presented in this sub-section.

Definition 7-2. Let $S_{T_c} = \{T_1[i], T_2[i], \dots, T_m[i]\}$ represent a set of fuzzy category trees. The combination of the fuzzy category trees in S_{T_c} is denoted as $T_c[i] = combine(S_{T_c})$. For each node $t_c[j]$ in $T_c[i]$, $v_c(t_c[j]) = \bigvee_{k=1}^{m} v_c(t_k[j])$.

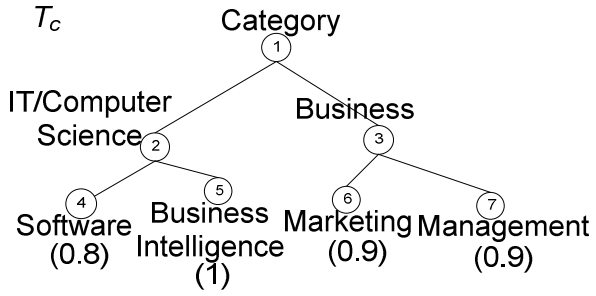


Figure 7-4. The combination of two fuzzy category trees in Figure 7-3

For example, the combination of two fuzzy category trees in Figure 7-3 is shown in Figure 7-4.

7.3.3 THE PEDAGOGICAL RELATIONS BETWEEN LEARNING ACTIVITIES

In the learning activity recommendation, the learning process, which is concerned with repeatability, periodicity and some dependency relations, must be considered (Salehi & Kamalabadi 2013). Recommended learning activities must be new, or have a level slightly above the learners' current competence level (Salehi & Kamalabadi 2013; Vygotskiĭ 1978). For some learning activities with similar content, or under the similar categories, it is not reasonable to recommend the elementary activities to a learner if

he/she has already learned some advanced activities. Our system considers two kinds of precedence relations between learning activities.

The first kind of precedence relations are derived from the prerequisites of learning activities. Many learning activities have prerequisite courses. These prerequisite learning activities are specified for the learning activity and described in the learning activity trees. The second kind of precedence relations are derived from learning sequences in learners' learning history. These learning sequences can be used to infer the advanced levels of learning activities, which are difficult to identify due to the open environment in the informal learning setting. Some sequential feature factors are defined as follows to identify the sequential relations between learning activities from the learning sequences.

For a learning activity *a* learned by a learner, there is a starting time t_s and a finishing time t_f. Obviously, $t_s(a) < t_f(a)$. Let a_1 and a_2 be two learning activities which are both learned by a learner. According to Allen's interval algebra (Allen 1983), there are thirteen temporal relations between a_1 and a_2. In this study, only the precedence relations are concerned. The following three sequential relations are considered: 1) a_1 is prior to a_2, denoted as $a_1 \rightarrow a_2$, if $t_f(a_1) \leq t_s(a_2)$; 2) a_2 is prior to a_1, denoted as $a_2 \rightarrow a_1$, if $t_f(a_2) \leq t_s(a_1)$; 3) a_1 and a_2 are concurrent, if $t_s(a_1) < t_f(a_2)$ and $t_s(a_2) < t_f(a_1)$. In the learning history, the relevant learning times of the learning activities for each learner are recorded.

To analyse the sequential relations between learning activities from the whole learners' learning histories, the following coefficients are defined. Let the support of a learning activity set *L*, *support*(*L*), be defined as the percentage of the learners who learned all the activities in *L* in all learners. $support(\{a_1, a_2\})$ represents the proportion of learners who learned both a_1 and a_2. $support(\{a_1 \rightarrow a_2\})$ represents the proportion of learners who learned a_1 before a_2. A prior relation confidence coefficient is defined as:

$$priorc(a_1 \rightarrow a_2) = \frac{support(\{a_1 \rightarrow a_2\})}{support(\{a_1, a_2\})}. \quad (7\text{-}4)$$

When learning activities a_1 and a_2 satisfy a minimum prior relation confidence and a minimum support threshold, i.e., $priorc(a_1 \rightarrow a_2) > priorc_{thres}$ and $support(\{a_1, a_2\}) > support_{thres}$, it indicates that there is a dependency relation between a_1 and a_2, and a_1 is usually learned before a_2. If a learner has learned a_2, it will not be suitable to recommend a_1 to him/her. A sequence set S_{prior} is used to record these relations. S_{prior} is constructed offline periodically.

7.4 LEARNER PROFILE TREE

This section defines a learner profile tree to model fuzzy tree-structured learner profiles. When a learner wants to select a learning activity, various kinds of information, such as the learner's background, learning goal, required learning categories, and learned learning activities, will influence the learner to make the decision. All the information should be contained in the learner profile tree. The structure of a learner profile tree is illustrated in Figure 7-5.

7.4.1 NODE CONCEPT OF THE LEARNER PROFILE TREE

Similar to the learning activity tree, the learner profile tree nodes are assigned a *label* attribute and a *category* attribute, which are used to calculate the node concept similarity.

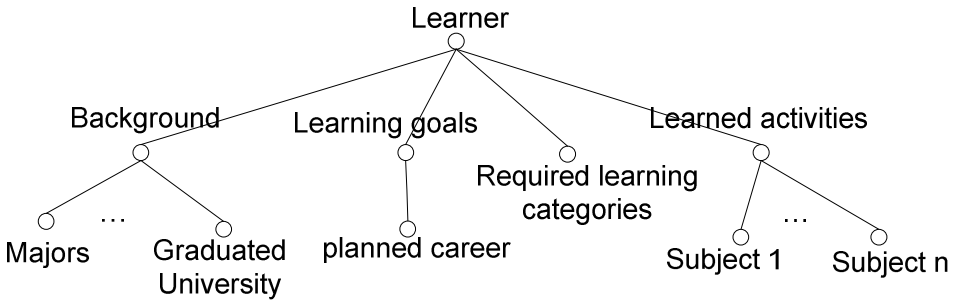


Figure 7-5. The structure of a learner profile tree

In the learner profile, the learned activities are recorded by the system during the learning process. Other information, such as the learner's background, planned career and required learning categories, is specified by the learner when the learner registered. In particular, the planned career is selected from a predefined career list, such as *Software*

Engineer, *Developer Programmer*, *Accountant*, etc. The required learning categories are selected by learners from the learning activity category tree, which is illustrated in Section 7.3.2, and the requirement levels of the sub-categories are specified by the learners. In a real life situation, the requirements are usually uncertain and described by linguistic terms. Thus, the category requirements are represented as fuzzy required category trees.

Our recommender system defines a linguistic term set *R*={*Very low required* (*VLR*), *Low required* (*LR*), *Medium required* (*MR*), *High required* (*HR*), *Very high required* (*HR*)} for learners to express their requirements for a specific learning category. To handle these linguistic terms in the recommendation calculation process, fuzzy set technology is, therefore, used (Zhang & Lu 2004). A set of triangular fuzzy numbers is applied to deal with these linguistic terms (Zhang & Lu 2003, 2009). The related fuzzy numbers to these linguistic terms are shown in Table 7-1.

Table 7-1. Linguistic terms and related fuzzy numbers for learner requirement

Linguistic terms	*VLR*	*LR*	*MR*	*HR*	*VHR*
Triangular Fuzzy Numbers	(0,0,0.25)	(0,0.25,0.5)	(0.25,0.5,0.75)	(0.5,0.75,1)	(0.75,1,1)

Two examples of fuzzy required category trees are shown in Figure 7-6. The linguistic terms under the nodes represent the learner's requirement. It can be seen from the examples that learners' requirements can be specified at different levels. For each branch of the tree, only one node is assigned to the user's requirement.

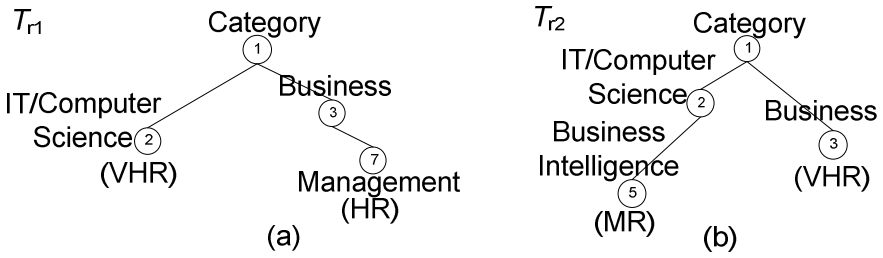


Figure 7-6. Two fuzzy required category trees

7.4.2 THE SIMILARITY MEASURES RELATED TO THE FUZZY REQUIRED CATEGORY TREE

In the recommendation generation process, the similarity measure is necessary to find similar users or items and to match suitable items to users' requirements. In this section, the similarity between two learners' fuzzy required category trees is presented to assist to compare the two learners, and the matching similarity of a learning activity's fuzzy category tree to a learner's fuzzy required category tree is given to help select proper learning activities.

7.4.2.1 FUZZY REQUIRED CATEGORY SIMILARITY

Let T_{r1} and T_{r2} be two fuzzy required category trees. The similarity measure between T_{r1} and T_{r2} is given in this sub-section. As learners' fuzzy required category trees are based on the learning activity category tree, T_{r1} and T_{r2} have the same base structure and labels. We use the numbering of the learning activity category tree to represent the nodes in T_{r1} and T_{r2}. The fuzzy required category similarity between T_{r1} and T_{r2} is calculated by

$$s_{frc}(T_{r1}[i], T_{r2}[i]) =$$

$$\begin{cases} s_v(v_r(t_{r1}[i]), v_r(t_{r2}[i])), & v_r(F_{r1}[i]) = 0, v_r(F_{r2}[i]) = 0 \\ s_v(v_r(t_{r1}[i]), v_r(T_{r2}[i])), & v_r(F_{r1}[i]) = 0, v_r(F_{r2}[i]) \neq 0 \\ s_v(v_r(T_{r1}[i]), v_r(t_{r2}[i])), & v_r(F_{r1}[i]) \neq 0, v_r(F_{r2}[i]) = 0 \\ (\alpha^{h-d_i} - \alpha^h) \cdot s_v(v_r(T_{r1}[i]), v_r(T_{r2}[i])) & \\ + (1 - \alpha^{h-d_i} + \alpha^h) \cdot \sum_{j=1}^{n_i} w_j \cdot s_{frc}(T_{r1}[i_j], T_{r2}[i_j]), & v_r(F_{r1}[i]) \neq 0, v_r(F_{r2}[i]) \neq 0 \end{cases} \quad (7\text{-}5)$$

where α is the influence factor of the parent node, h is the height of the learning category tree, and d_i is the depth of node i in the category tree; w_j is the weight of $T_{r1}[i_j]$ and $T_{r2}[i_j]$, which is calculated as $w_j = (w(t_{r1}[i_j]) + w(t_{r2}[i_j]))/2$; $v_r(t_{r1}[i])$ represents the value of node $t_{r1}[i]$, which is a fuzzy number; $v_r(F_{r1}[i])$ represents the value of forest $F_{r1}[i]$, which is 0 if $F_{r1}[i]$ is *null* or none of its nodes are assigned values;

$v_r(T_{r1}[i])$ represents the value of the sub-tree $T_{r1}[i]$, which is calculated by Formula (7-6); $s_v(\cdot)$ is the similarity measure for two fuzzy numbers.

$$v_r(T_r[i]) = \begin{cases} v_r(t_r[i]), & v_r(F_r[i]) = 0 \\ \sum_{j=1}^{n_i} w(t_r[i_j]) \cdot v_r(T_r[i_j]), & v_r(F_r[i]) \neq 0 \end{cases}. \quad (7\text{-}6)$$

For two fuzzy numbers $\tilde{a}$, $\tilde{b}$,

$$s_v(\tilde{a}, \tilde{b}) = 1 - d(\tilde{a}, \tilde{b})/d_{\max}, \quad (7\text{-}7)$$

where $d(\tilde{a}, \tilde{b})$ is the distance between fuzzy numbers $\tilde{a}$ and $\tilde{b}$ as defined by Formula (2-6), $d_{\max}$ is the maximum distance between fuzzy numbers in the domain.

Let α be 0.5. Taking the two learner requirement trees in Figure 7-6 as an example, the fuzzy required category similarity between them is computed by $s_{frc}(T_{r1}[1], T_{r2}[1]) = w_2 \cdot s_v(v_r(t_{r1}[2]), v_r(T_{r2}[2])) + w_3 \cdot s_v(v_r(T_{r1}[3]), v_r(t_{r2}[3]))$, and calculated as 0.675.

7.4.2.2 FUZZY CATEGORY MATCHING SIMILARITY

Let T_r be a learner's fuzzy required category tree, and T_c represent the fuzzy category tree of a learning activity. The fuzzy category matching similarity measure of T_c to T_r is calculated by Formula (7-8).

$$s_{fcm}(T_r[i], T_c[i]) = \begin{cases} s_v(v_r(t_r[i]), v_c(t_c[i])), & v_r(F_r[i]) = 0, v_c(F_c[i]) = 0 \\ s_v(v_r(t_r[i]), v_c(T_c[i])), & v_r(F_r[i]) = 0, v_c(F_c[i]) \neq 0 \\ s_v(v_r(T_r[i]), v_c(t_c[i])), & v_r(F_r[i]) \neq 0, v_c(F_c[i]) = 0 \\ (\alpha^{h-d_i} - \alpha^h) \cdot s_v(v_r(T_r[i]), v_c(T_c[i])) & \\ + (1 - \alpha^{h-d_i} + \alpha^h) \cdot \sum_{j=1}^{n_i} w(t_r[i_j]) \cdot s_{fcm}(T_r[i_j], T_c[i_j]), & v_r(F_r[i]) \neq 0, v_c(F_c[i]) \neq 0 \end{cases}. \quad (7\text{-}8)$$

In respect of the fuzzy category matching similarity measure, first, the category values of nodes in T_c are real numbers, which will be seen as special fuzzy numbers in

the similarity measure $s_v(\cdot)$. Second, the similarity between T_r and T_c is asymmetric, and only the weights of T_r are considered.

Taking the fuzzy required category tree T_{r1} in Figure 7-6 and the two fuzzy category trees of *Business Intelligence* and *Marketing Management* illustrated in Figure 7-3 as examples, the matching similarity of *Business Intelligence* to T_{r1} is computed by $s_{fcm}(T_{r1}[1], T_a[1]) = w_2 \cdot s_v\big(v_r(t_{r1}[2]), v_c(T_a[2])\big) + w_3 \cdot s_{fcm}(T_{r1}[3], T_a[3])$, and calculated as 0.845. Similarly, the matching similarity of *Marketing Management* to T_{r1} is calculated as 0.722. Because T_{r1} expresses very high requirement on "IT" category, and the degree of *Business Intelligence* belonging to "IT" is higher than that of *Marketing Management*, the calculated matching similarity degrees reflect the requirement.

7.5 A MODIFIED IUTH RECOMMENDATION APPROACH FOR LEARNING ACTIVITIES

This section outlines the development of the recommendation approach for learning activities by use of the methodology of IUTH recommendation approach in Chapter 4, which is modified for the specific requirements of e-learning systems. In e-learning systems, learners always want to learn something new or with higher (more advanced) difficulty levels. To effectively recommend different or new learning activities to learners, the CF recommendation part of IUTH employs the user-based CF technique in this system. For a target learner u_t, the recommendation process is described in seven steps, as follows.

Step 1: Determine the recommendation alternatives

There are numerous learning activities under various categories in an e-learning system, but for a specific target learner, only the learning activities under certain relevant categories are suitable for recommendation. To improve the recommendation efficiency, the relevant categories of the target learner are first identified, and the learning activities under the categories are then selected.

The relevant learning categories of the target learner u_t are identified in two ways: the learning activities that have been learned by u_t and other learners with the same learning goals; and the fuzzy required category tree T_{frc} of u_t. Let the learning goal of u_t be g_t. The learners whose learning goal is g_t are selected to constitute a set U_{g_t}. For each learner $u_i \in U_{g_t}$, the learned activities are $\{a_{i,1}, a_{i,2}, \dots, a_{i,n_i}\}$, and the corresponding fuzzy category trees are $\{T_{i,1}, T_{i,2}, \dots, T_{i,n_i}\}$. The learned category tree of u_i, denoted as T_i, can be calculated as $T_i = combine(\{T_{i,1}, T_{i,2}, \dots, T_{i,n_i}\})$. The learned category trees of all the users in U_{g_t} are combined, and the learned category tree for the learning goal g_t is obtained and denoted as T_{g_t}. A fuzzy category tree T_{cr} is derived from the learner's fuzzy required category tree by setting the membership degrees of leaf nodes in T_{frc} as 1. The relevant learning category tree is obtained by combing T_{g_t} and T_{cr}, as $T_{cr} = combine(\{T_{cr}, T_{g_t}\})$. For any learning activity a with fuzzy category tree T_{ca}, if $s_{fc}(T_{ca}, T_{cr}) > 0$, it is preselected.

The pedagogical constraints are considered when preselecting the learning activities. Let the profile tree of the target learner u_t be denoted as T_t. The sub-tree of T_t which represents the learned learning activities, is denoted as $T_{t,l}$. The learned activities are $\{a_{t,1}, a_{t,2}, \dots, a_{t,n_t}\}$. For a learning activity a, the sequential and prerequisite constraints are verified separately. For the sequential constraints, if $\exists(a \rightarrow a_{t,i}) \in S_{prior}$, $1 \leq i \leq n_t$, a will not be suitable for recommendation. For the prerequisite constraints, let the learning activity's prerequisite sub-tree be denoted as $T_{a,p}$. As mentioned before, it is usually impossible to match two learning activities just from their IDs or names. The proposed tree matching method is used to check if a learning activity is suitable for the learner. A sub-tree match is calculated by use of the conceptual similarity between two tree-structured data proposed in Section 3.3, as $s_a = sc_{T_{asym}}(T_{a,p}, T_{t,l})$. A matching similarity threshold s_{thres} is predefined. If $s_a > s_{thres}$, then learning activity a can be selected as a recommendation alternative.

By using this step, a set of recommendation alternatives are chosen. For each alternative learning activity a, the following steps are taken to predict its rating.

Step 2: Calculate the matching degree of the learning activity a to the learner's requirement

The learner u_t's fuzzy required category tree is T_{req}, and the learning activity a's fuzzy category tree is T_{ca}. The matching degree of a to u_t is calculated by Formula (7-9):

$$s_m(u_t, a) = s_{fcm}(T_{req}, T_{ca}). \tag{7-9}$$

Step 3: Calculate the semantic similarity between users

The users who have rated a are selected, denoted as $U_a = \{u_1, u_2, \dots, u_m\}$. For each user $u_i \in U_a$, let the profile tree be T_i. The semantic similarity between u_t and u_i is calculated by use of the conceptual similarity between two tree-structured data proposed in Section 3.3, as:

$$s_{sem}(u_t, u_i) = sc_{T_{sym}}(T_t, T_i, M_{t,i}). \tag{7-10}$$

During the calculation process of $s_{sem}(u_t, u_i)$, a maximum conceptual similarity tree mapping between the profile trees of u_t and u_i is constructed. Their most similar learned activities can be matched. Let the matched learning activities be recorded in $M_{t,i}$. For any $(p, q) \in M_{t,i}$, p and q are the learning activities rated by u_t and u_i, respectively.

Step 4: Calculate the CF similarity between users

A learning activity similarity threshold as_t is predefined. For any learning activity pair (p, q), T_p and T_q are their learning activity trees respectively. p and q will be shown to be irrelevant if the similarity between T_p and T_q, $s_{sem}(p, q) = sc_{T_{sym}}(T_p, T_q)$ is less than as_t. Given the matched learning activity set $M_{t,i}$ of u_t and u_i, a sub-set $M'_{t,i} = \{(p, q): (p, q) \in M_{t,i}, s_{sem}(p, q) > as_t\}$ is selected. Based on $M'_{t,i}$, the CF similarity between u_t and u_i is calculated as:

$$s_{CF}(u_t, u_i) = \frac{\sum_{(p,q)\in M'_{t,i}} r_{t,p} \times r_{i,q}}{\sqrt{\sum_{(p,q)\in M'_{t,i}} r_{t,p}^2} \times \sqrt{\sum_{(p,q)\in M'_{t,i}} r_{i,q}^2}}, \tag{7-11}$$

where $r_{t,p}$ is the rating of item p from user u_t.

Step 5: Select top-N similar users

The total similarity between users u_t and u_i is computed by integrating the two similarity measures computed in the last two steps.

$$s_u(u_t, u_i) = \beta \times s_{sem}(u_t, u_i) + (1 - \beta) \times s_{CF}(u_t, u_i), \quad (7\text{-}12)$$

where $\beta \in [0,1]$ is a semantic combination parameter specifying the weight of similarity in the integrating measure. The users in U_a are sorted according to the total similarity. The top-N most similar users are selected as neighbours to predict ratings.

Step 6: Calculate the predicted rating

The predicted rating to learning activity a of learner u_t is calculated as:

$$pr_{t,a} = \theta \times s_m(u_t, a) \times r_{\max} + (1-\theta) \times \frac{\sum_{i=1}^{N} r_{i,a} \times s_u(u_t, u_i)}{\sum_{i=1}^{N} s_u(u_t, u_i)}, \quad (7\text{-}13)$$

where $\theta \in [0,1]$, $r_{\max}$ represents the maximum value of ratings. The formula contains two parts. $s_m(u_t, a) \times r_{\max}$ is the requirement matching-based predicted rating. If the target learning activity is exactly matched to the user's requirement, the target item should achieve the highest rating. $\sum_{i=1}^{N} r_{i,a} \times s_u(u_t, u_i) / \sum_{i=1}^{N} s_u(u_t, u_i)$ is the traditional user-based CF-based predicted rating. θ is a parameter that combines the two parts.

Step 7: Generate the recommendations:

The predicted ratings of all the alternative learning activities of learner u_t are calculated. The alternatives are ranked according to the predicted rating, and the top-K of them are recommended to the learner.

7.6 SYSTEM ARCHITECTURE

In this section, the architecture of the e-learning recommender system is presented. It is designed to have three types of users: system administrators, teachers and students. The roles of the users are described as follows.

- *System administrator*. The role of the system administrator is to maintain the learning activity category and the career list of learners, which are used to support the operation of the system.
- *Teacher*. The teachers are responsible for managing the learning activities. They input the learning activities with detailed descriptions into the system. When a learning activity is input, its categories and the related membership degrees are specified by the teacher. During the operation of the system, teachers obtain feedback from students about their learning activities and interact with the students.
- *Student*. The students are searching the appropriate learning activities and want to receive recommendations. They provide their background information and learning requirements when registering in the system. After finishing a learning activity, the student can provide feedback and rate the learning activity.

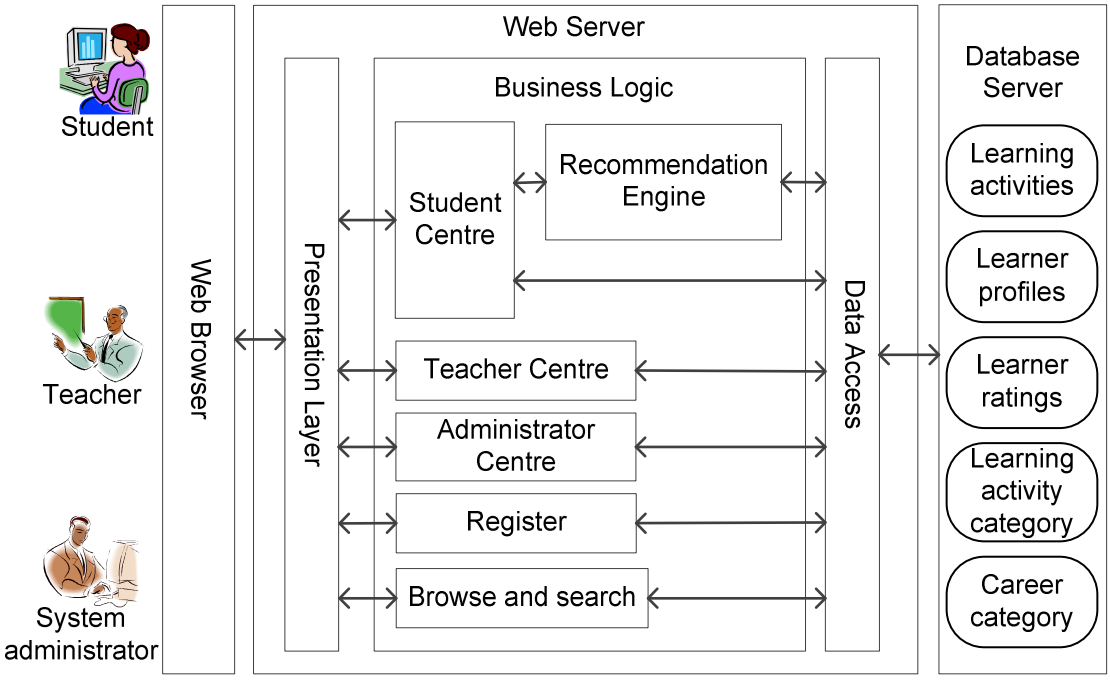


Figure 7-7. The architecture of the e-learning recommender system

The architecture of the e-learning activity recommender system is depicted in Figure 7-7. As a web-based online system, the e-learning recommender system has a standard multi-tier architecture, which includes web browser, web server, and database server. The main components of the system are described as follows.

7.6.1 DATABASES

The database stores all the data of the system, which includes the following main components:

- *Learning activity*: stores the information of each learning activity, which is used to construct the learning activity tree.
- *Learner profile*: stores the profile of each learner, which is used to construct the learner profile tree.
- *Learner ratings*: stores the learners' ratings to their learned learning activities.
- *Learner activity category*: maintains the tree-structured learning activity categories.
- *Career category*: maintains the career list of learners.

7.6.2 WEB APPLICATION COMPONENTS

The application in the web server contains three layers: the presentation layer, business logic layer and data access layer.

1) Presentation layer

The presentation layer is responsible for generating the requested web pages and handling the user interface logic and events for the three kinds of users.

2) Business logic layer

The business logic layer realizes the business services and the core recommendation algorithm. It provides the following main functionalities as follows:

- *Browse and search*: supports learners to browse the learning activities, or search the learning activities based on keyword-based search engine, and to view the matching results.
- *Register*: supports new users to register into the system.
- *Administrator centre*: is used by administrators to manage the users and common data.
- *Teacher centre*: supports teachers to input and manage the learning activities.
- *Student centre*: collects the learner's profile and requirements, tracks the user's learning behaviour, and calls the recommendation engine to generate learning activity recommendations to the learner. Learners' comments and ratings to the recommendation results are also collected through the module.
- *Recommendation engine*: implements the proposed recommendation approach and generates recommendations for student users. It implements the following main sub-components: (i) *learning activity tree construction*: constructs the learning activity trees that are illustrated in Section 7.3; (ii) *learner profile tree construction*: constructs the learner profile trees that are illustrated in Section 7.4; (iii) *tree similarity measure algorithm*: implements the tree similarity measure computation algorithms developed in Section 3.3. It provides not only the interface to calculate the conceptual similarity measure between two tree-structured data, but also the interface to generate the maximum conceptual similarity tree mapping between tree-structured data; (iv) *user-based CF similarity*: computes the user-based CF similarity between two users based on the user-item rating matrix, which is used in Step 4 in the proposed recommendation approach in Section 7.5; (v) *fuzzy category similarity*: computes the fuzzy category similarity between two learning activities as defined in Section 7.3.2; (vi) *fuzzy required category similarity*: computes the fuzzy required category similarity between two learners as defined in Section 7.4.2; (vii) *fuzzy category matching similarity*: calculates the matching degree of a learning activity to a learner's requirement; (viii) *modified IUTH recommendation approach for*

learning activities: implements the proposed recommendation approach in Section 7.5.

3) Data access layer

The data access layer deals with the data operations of the database.

7.7 SYSTEM IMPLEMENTATION

The system is developed and implemented 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 respectively. All the functionalities of the web application components described in last sub-section are implemented. The database is designed and implemented in the PostgreSQL database server. Tables are designed and created to store the entities described in the system architecture. To test the recommender system, it is deployed in the Glassfish web server. Figure 7-8 shows the home page of the e-learning recommender system.

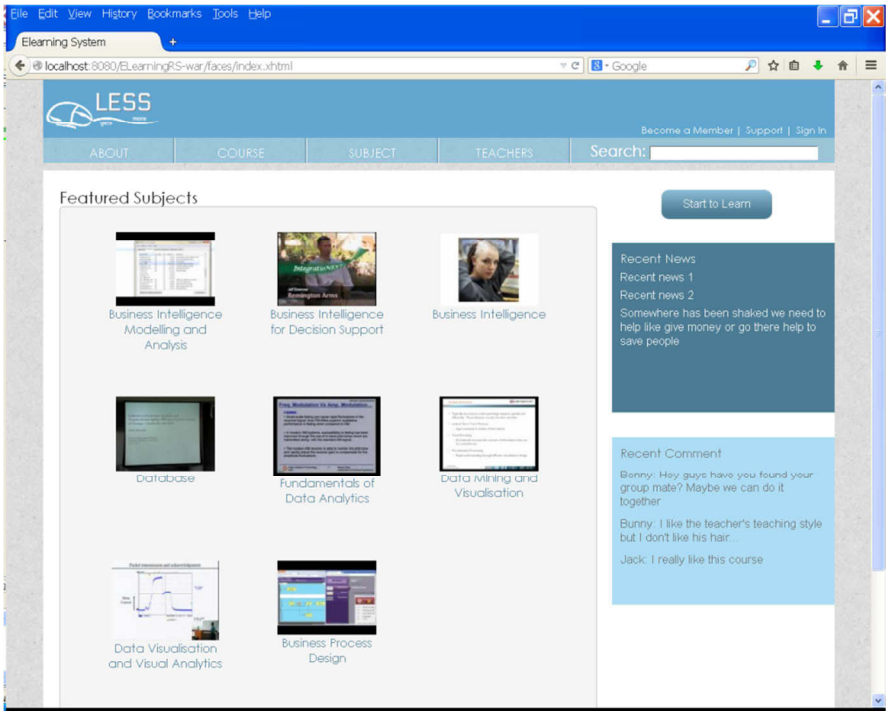
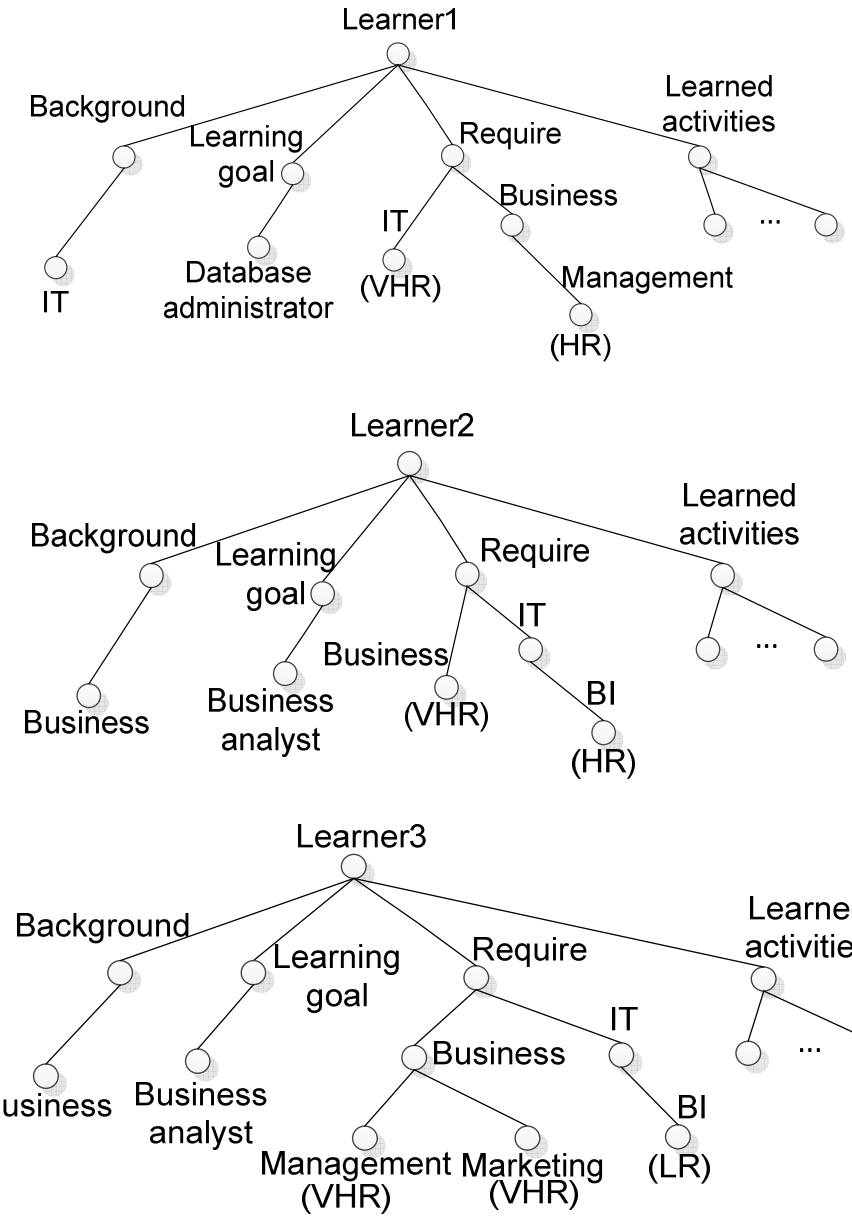


Figure 7-8. The homepage of the e-learning recommender system

7.8 A CASE STUDY

In the e-learning recommender system, it is supposed that there are five learners (Leaner 1, …, Leaner 5) and eight subjects (S1-*Business Intelligence*, …, S8-*Business Process Design*) available. The fuzzy tree-structured learner profiles are described in Figure 7-9, and the fuzzy category trees of the subjects are shown in Figure 7-10.



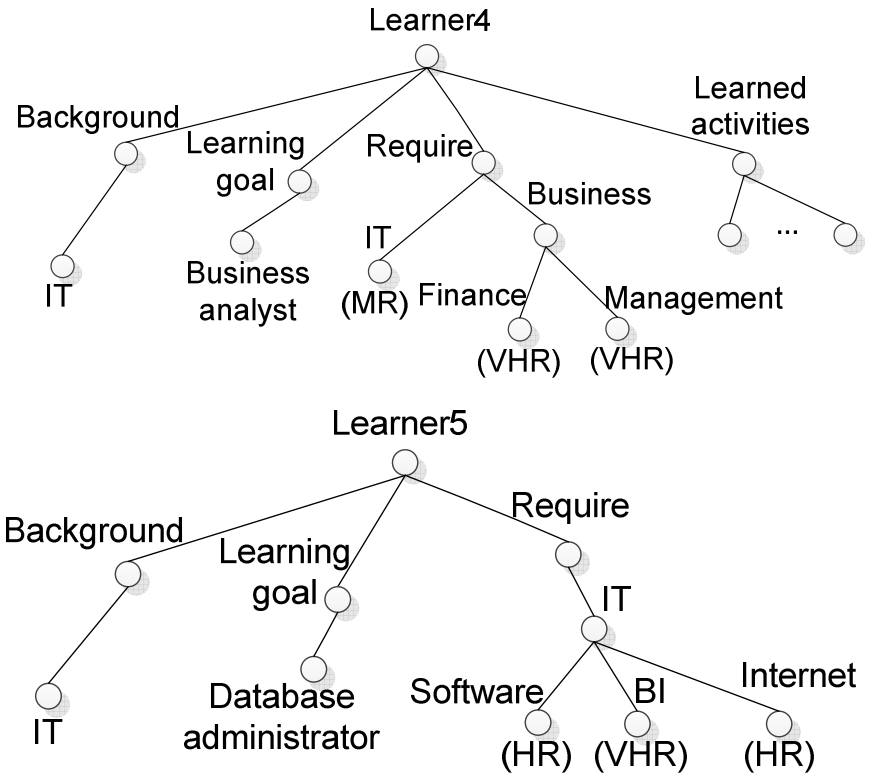
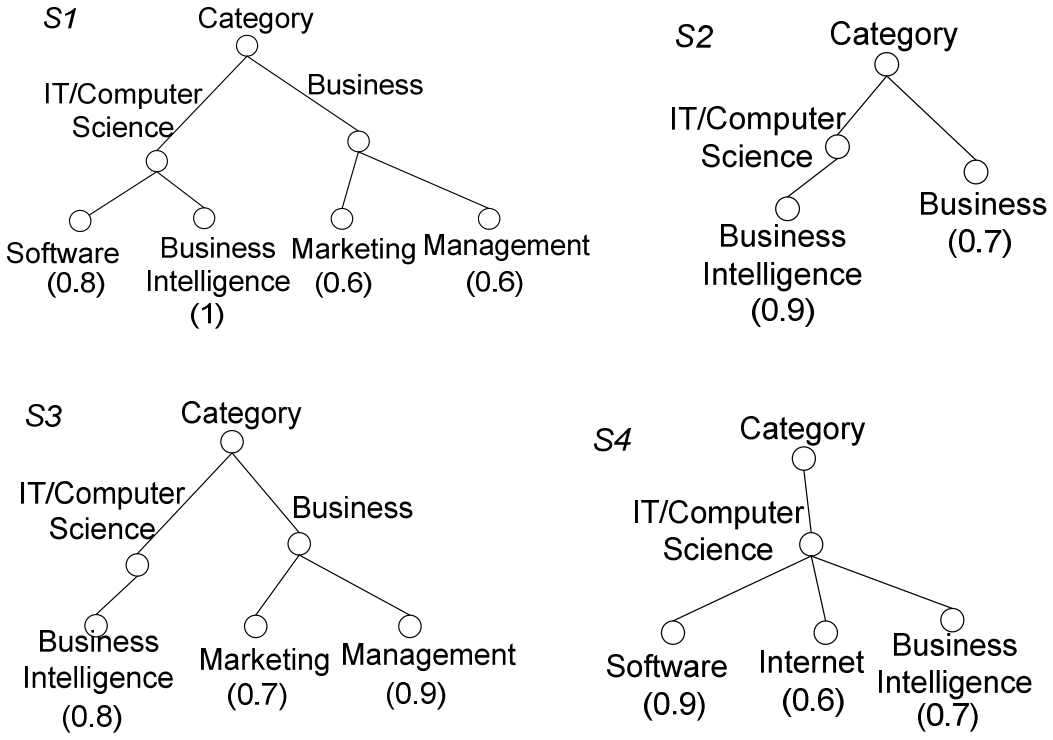


Figure 7-9. Five learner profiles



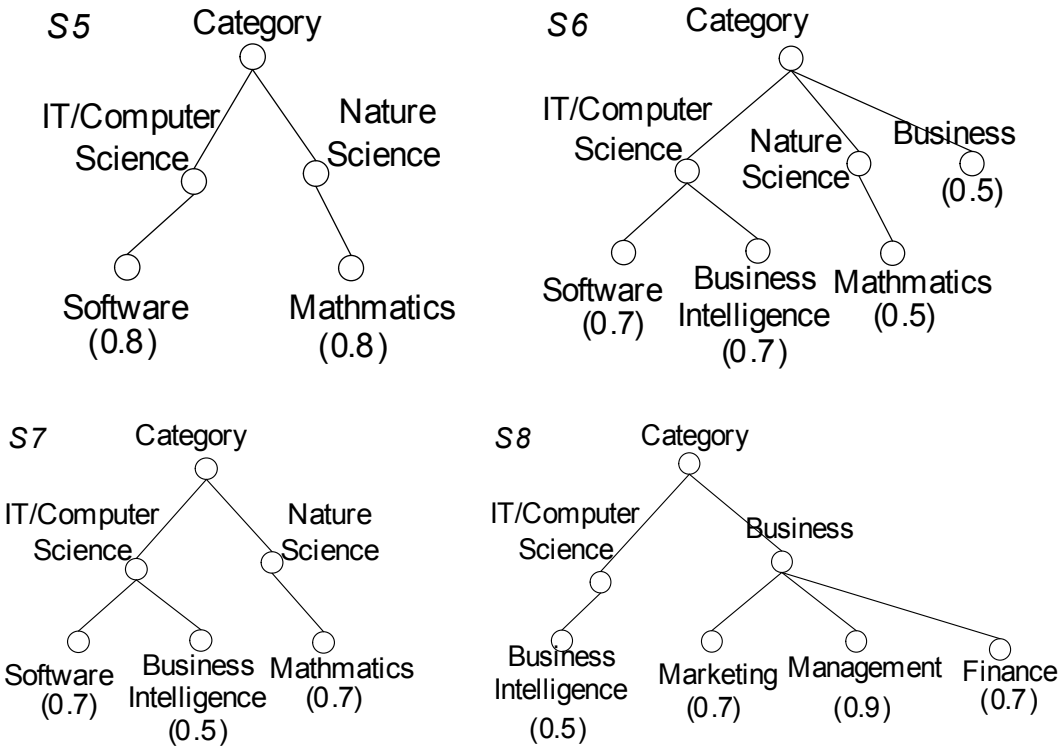


Figure 7-10. The fuzzy category trees of the subjects

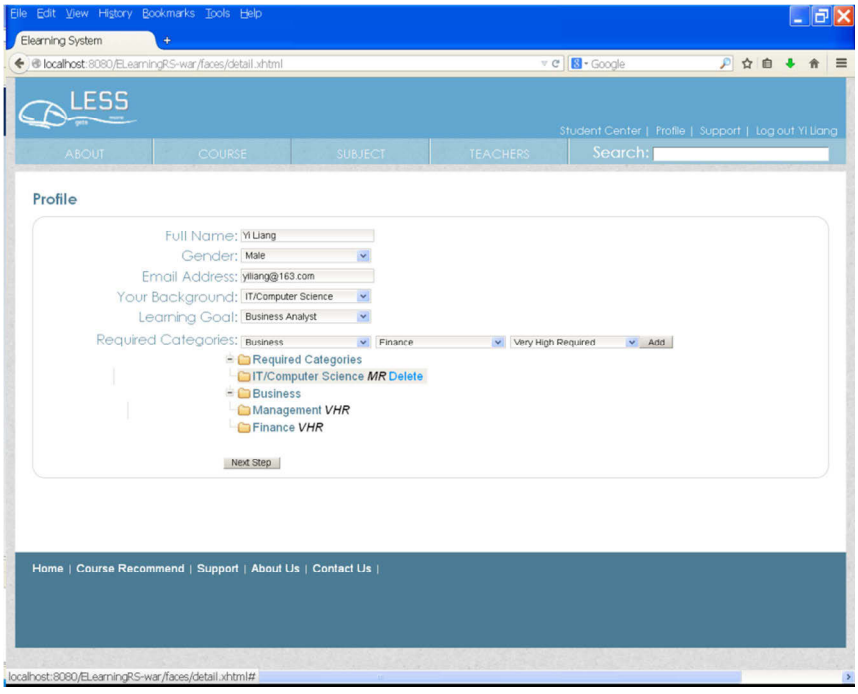


Figure 7-11. The student profile page

When a learner signs into the system, he/she can edit the profile. The learner's background, learning goal, and preferred learning categories can be specified. For example, Figure 7-11 shows the profile editing page of Learner 4, in which the learner's required categories construct a tree structure and the required levels are expressed by linguistic terms.

The study room in the student centre presents the learner's learned activities and current learning progress, which is used for learners to manage their current learning activities. For example, the study room of Learner 4 is shown in Figure 7-12. A learner can also provide ratings and comments for a learning activity. This recommender system provides ratings on a scale of 1 to 5. Figure 7-13 provides an example.

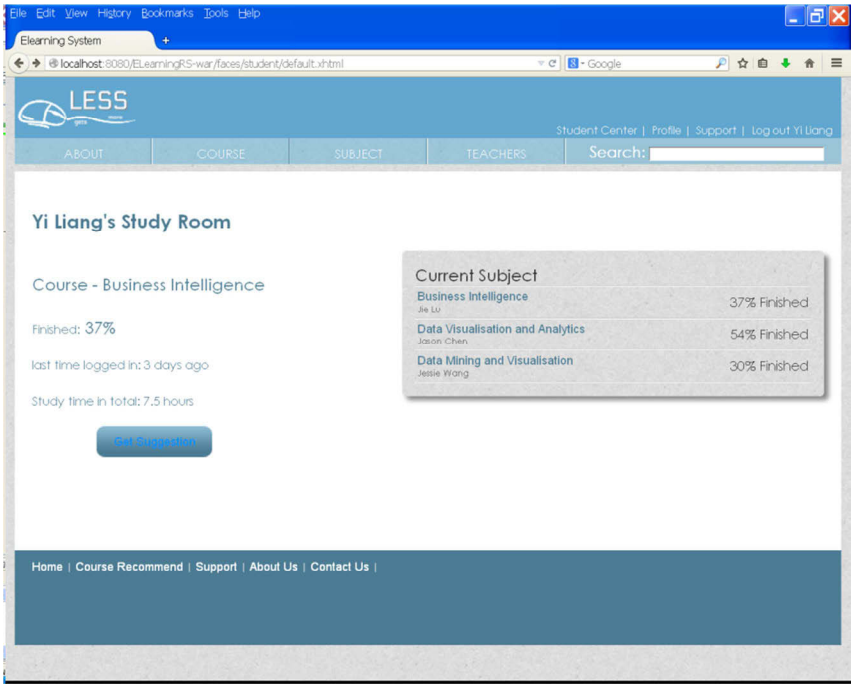


Figure 7-12. The student's study room

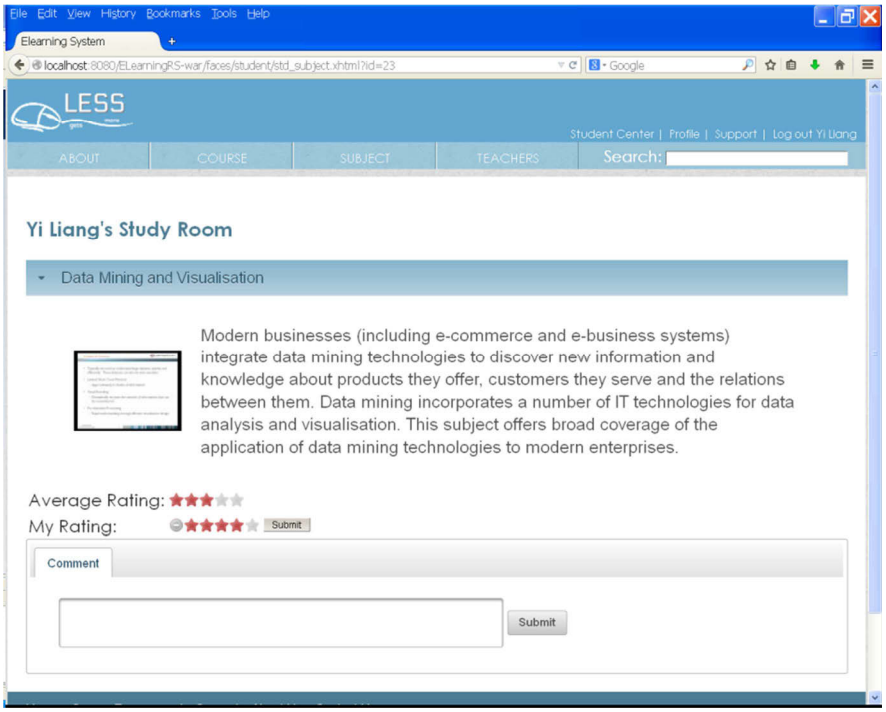


Figure 7-13. Student rating and comment input page

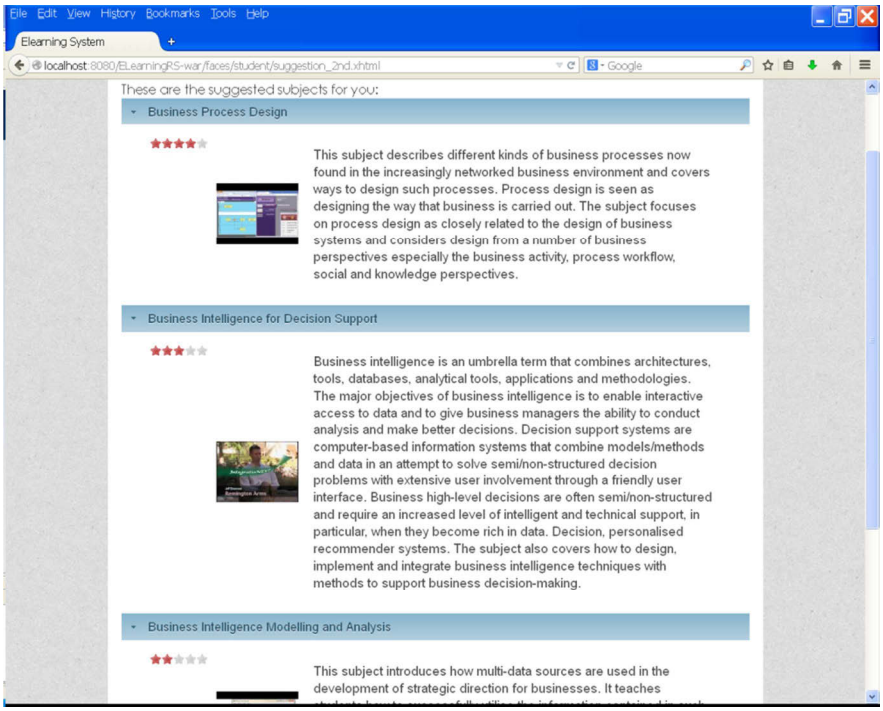


Figure 7-14. Learning activity recommendation results

The existing learner-subject rating matrix in the case study is depicted in Table 7-2. It can be seen that Learner 5 is a new registered learner, and the subject S8 (*Business*

Process Design) is a new item. In this case study, subjects recommended to Learner 4 and Learner 5 will be generated.

Table 7-2. Learner-subject rating matrix

	Learner 1	Learner 2	Learner 3	Learner 4	Learner 5
S1: Business Intelligence		4	4	5	
S2: BI Modelling and Analysis	3		2		
S3: BI for Decision Support		3	3		
S4: Database	3	4			
S5: Fundamentals of Data Analytics	4		3		
S6: Data Visualisation and Analytics	5			2	
S7: Data Mining and Visualisation		2		4	
S8: Business Process Design					

The recommendation process is as follows:

1) The recommendation alternatives are selected for Learner 4 and Learner 5 according to Step 1 in the method in Section 7.5. The potential learning activities for Learner 4 are {S2, S3, S4, S5, S8}, and for Learner 5 are {S1, S2, S3, S4, S5, S6, S7, S8}.

2) The matching degrees of the alternative learning activities to the learners are calculated, as shown in Table 7-3.

3) The semantic similarity degrees between learners are calculated, as shown in Table 7-4.

4) The CF similarity degrees between learners are calculated, as shown in Table 7-5.

5) The total similarity degrees between learners are calculated, as shown in Table 7-6.

6) The predicted ratings of the alternative learning activities by the learners are calculated, as shown in Table 7-7.

7) The alternative learning activities are ranked according to their predicted ratings. The learning activities with the highest ratings are recommended to the learners.

Figure 7-14 shows the recommendation result for Learner 4.

Table 7-3. The matching degrees of the learning activities to the learners

	S1	S2	S3	S4	S5	S6	S7	S8
Learner 4		0.66	0.62	0.29	0.33			0.85
Learner 5	0.63	0.44	0.42	0.78	0.43	0.61	0.56	0.31

Table 7-4. The semantic similarity between learners

	Learner 1	Learner 2	Learner 3	Learner 4
Learner 4	0.59	0.67	0.62	1
Learner 5	0.77	0.14	0.06	0.44

Table 7-5. The CF similarity between learners

	Learner 1	Learner 2	Learner 3	Learner 4
Learner 4	0.84	0.94	0.78	
Learner 5	N/A	N/A	N/A	N/A

Table 7-6. The total similarity between learners

	Learner 1	Learner 2	Learner 3	Learner 4
Learner 4	0.72	0.80	0.70	
Learner 5	0.77	0.14	0.06	0.44

Table 7-7. The predicted ratings

	S1	S2	S3	S4	S5	S6	S7	S8
Learner 4		2.91	3.05	2.48	2.59			4.26
Learner 5	3.92	2.56	2.57	3.54	3.04	3.47	3.16	1.53

In the case study, Learner 4 expresses very high requirement on “Business” category and medium requirement on “IT” category, while Learner 5 requires very highly on “IT” category. Their recommendation results reflect their requirements. Additionally, the similar learners, selected according to both the CF similarity and semantic similarity, also take effects in the recommendation process. Even though Learner 5 is a new learner, his/her similar learners can also be identified with the semantic similarity. The new subject *Business Process Design* can also be recommended through the requirement

matching knowledge. It is seen from the case study that the semantic information of learning activities and learners, and the requirement matching knowledge are fully utilised in the recommendation process, the system can recommend learning activities to learners effectively.

7.9 SUMMARY

This chapter has outlined the development of an e-learning recommender system to assist learners to effectively select appropriate learning activities. In the system, the learning activities and learner profiles are modelled as learning activity trees and learner profile trees respectively based on the tree-structured data model proposed in Chapter 3. The tree similarity measure developed in Chapter 3 is used to evaluate the semantic similarity between learning activities or learners. To handle the fuzzy issues in the data in e-learning systems, fuzzy set techniques are used in the data model. A fuzzy category tree is defined to specify the categories that each learning activity roughly belongs to, and the fuzzy category similarity measure is developed to evaluate the semantic similarity between learning activities. A fuzzy required category tree is defined for learners to express their requirements. In addition, the precedence relations between learning activities are also handled through analysing the learning sequences and modelling the prerequisite learning activities. The methodology of the IUTH recommendation approach is adapted and applied in the system. The CF prediction part of the recommendation approach uses the user-based CF technique in this system to effectively recommend different or new learning activities to learners. When finding similar learners, the proposed system draws strength from both the semantic and CF similarities. When calculating the CF similarity, the ratings of the matched learning activities, rather than the exactly common learning activities between two users are used, which alleviates the sparsity problem caused by the sparse user-item rating matrix. A case study shows the effectiveness of the proposed system, in which both new learner and new learning activity can be recommended.

CHAPTER 8
CONCLUSIONS AND FURTHER STUDY

This chapter concludes the whole thesis and provides some further research directions of the topic.

8.1 CONCLUSIONS

This study is motivated by an awareness of practical issues in recommender systems. Recommender systems have been widely developed for various application domains during the past two decades. Even though recommender systems have gained considerable attention and undergone rapid developments, there is still a big challenge in current recommender system research: the items or user profiles in many applications, such as e-business, e-government and e-learning, are so complex that they can only be represented as tree structures. Since the similarity measure is the key technique in recommender systems, the similarity between these tree-structured items or user profiles must be evaluated to make appropriate recommendations. However, previous recommender systems lack the ability to deal with tree-structured data, which cannot guarantee the recommendation performance. This research attempts to address this issue.

The main contributions of this study are as follows:

1) It proposes a tree-structured data model (to achieve Objective 1) and a similarity measure on tree-structured data (to achieve Objective 2) in Chapter 3.

The tree-structured data model is based on the basic mathematical definitions of trees, and assigns tree nodes concept, value and weight. It can be used to model tree-structured items, user preferences, user profiles, etc. in recommender systems. The similarity measure on tree-structured data is developed based on the tree-structured data model. It compares two tree-structured data fully considering their differences of tree structures, node concepts, values and weights. In the similarity measure, a maximum conceptual similarity tree mapping is constructed to identify the conceptually corresponding parts. The conceptual similarity and value similarity between two trees are computed separately, and the final similarity is assessed as the weighted sum of the two. The proposed similarity measure contains two types of measure: symmetric and asymmetric. The symmetric measure treats the node weights of two trees to be compared equally, which can be used to evaluate the semantic similarity between tree-structured items or the semantic similarity between tree-structured users. The asymmetric measure mainly considers the node weights of the first tree, which can be used to evaluate the matching degree of tree-structured items to tree-structured user requirements. The proposed tree-structured data model and similarity measure can also be used in other application areas, in which the data present tree structures, besides recommender systems.

2) It proposes an item tree and user request tree-based hybrid recommendation approach (to achieve Objective 3) in Chapter 4.

To deal with the tree-structured data in recommender systems, an item tree and user request tree-based hybrid (IUTH) recommendation approach is developed. It models tree-structured items and users as item trees and user request trees respectively. The asymmetric similarity measure on tree-structured data is utilized to evaluate the matching degree of the item trees to user request trees, and the symmetric similarity measure on tree-structured data is utilized to evaluate semantic similarity between item trees. In the proposed IUTH recommendation approach, the final predicted rating hybridizes both the requirement matching-based prediction and the traditional item-based CF prediction. In the CF prediction module, item similarity combines the semantic similarity and the item-based CF similarity. The proposed recommendation approach deals well with the tree-structured data which commonly appear in real applications, such as business partner

(supplier or buyer) recommendations. The effectiveness of the approach is shown in a case study. Experimental results on two datasets show that the proposed recommendation approach has good performance and is well-suited in recommending tree-structured items. That is because the semantic information and matching knowledge is comprehensively used in the IUTH recommendation approach.

3) It proposes a fuzzy preference tree model (to achieve Objective 4) and a fuzzy preference tree-based recommendation approach (to achieve Objective 5) in Chapter 5.

To model users' tree-structured preferences with subjectivity and uncertainty, a fuzzy preference tree model is proposed, which integrates both the user's extensionally and intentionally expressed preferences. An incremental construction algorithm of the fuzzy preference tree is developed. A fuzzy preference tree-based recommendation approach is then developed to predict users' preference to unexperienced items. Experimental results on two datasets show that the proposed approach makes accurate recommendations, and demonstrate that the fuzzy preference tree model reflects user preferences effectively. It is concluded that the fuzzy representation can express users' uncertain preferences accurately and the tree-structured preference can comprehensively cover users' preferences.

4) It develops a business partner recommender system – Smart BizSeeker (to achieve Objective 6) in Chapter 6.

The Smart BizSeeker applies the proposed similarity measure on tree-structured data and recommendation approaches based on that, and models business products, buying requests, and preferences as tree-structured data. It handles the tree-structured data in business partner recommendation applications, and can assist business users to effectively select the right business partners (buyers and suppliers) that match their personal business needs and preferences, which can promote the development of e-business services. The development of the system demonstrates the usability of our proposed approaches. The design method of the system can also be referenced in other applications.

5) It develops an e-learning recommender system – ELRS (to achieve Objective 7) in Chapter 7.

The ELRS aims to provide learners personalised recommendations of learning activities. In the system, to comprehensively describe the features of learning activities and learner profiles, a learning activity tree and a learner profile tree are defined respectively based on the tree-structured data model. The similarity measure on tree-structured data is applied to evaluate the semantic similarity between learners. To handle the fuzzy issues in the data, fuzzy set techniques are utilized in the data model. Specifically, a fuzzy category tree is defined to specify the categories that each learning activity roughly belongs to, and a fuzzy required category tree is defined for learners to express their requirements. The methodology of the IUTH recommendation approach is adapted and applied in the system. The CF prediction part uses the user-based CF technique. When finding similar learners, the proposed system draws strength from both the semantic and CF similarities. When calculating the CF similarity, the ratings of the matched learning activities, rather than the exactly common learning activities between two users are used, which alleviates the sparsity problem caused by the sparse user-item rating matrix. In addition, the precedence relations between learning activities are also handled through analysing the learning sequences and modelling the prerequisite learning activities. The developed ELRS can support learners to effectively select appropriate learning activities, which enhances the development of e-learning services. The development of the system also demonstrates the applicability and flexibility of our proposed approaches.

8.2 FURTHER STUDY

There are still some limitations of the current study:

1) Even though the proposed similarity measure on tree-structured data allows that the trees have different structures and matched nodes can be in different levels, it requires that the semantic meanings of the parent-child relations in different tree-structured data must be the same.

2) When modelling users' preferences, this study focuses on the comprehensiveness, but pays less attention to the dynamics of the users' preferences and interests.

3) The parameters in the proposed approach have not been optimized.

This research can be fully advanced in the following aspects:

1) In the future, the various parent-child relations in the tree-structured data will be investigated. The transformation method between tree-structured data with different parent-child relations will be developed.

2) The users' preferences and interests may be changed over time. The more recent interests of a user should play a more important role when making recommendations to the user. The interest drift of users will be investigated when modelling users' preferences.

3) The developed Smart BizSeeker business recommender system and the ELRS e-learning recommender system will be further extended and tested. More real data will be collected to evaluate and improve the proposed recommendation approach. The parameters in the proposed approach will also be optimized. Moreover, we will consider deploying the developed recommender systems in real applications. The Smart BizSeeker may be applied in the website of the Australian local government to support local businesses. The ELRS is considered to be applied in the UTS website as the complement to students. The functionalities of the systems need to be adjusted and improved according to the real requirements, and the web pages need to be further developed.

4) The tree-structure data model has good flexibility and expansibility, so it is suitable to model the features and characteristics of groups of users. We will combine the group recommendation techniques with the tree-structured data model and similarity measure to develop methods to identify user groups and make group recommendations.

REFERENCES

Aamodt, A. & Plaza, E. 1994, 'Case-based reasoning: foundational issues, methodological variations, and system approaches', *AI Communications*, vol. 7, no. 1, pp. 39-59.

Adomavicius, G. & Tuzhilin, A. 2005a, 'Personalization technologies: a process-oriented perspective', *Communications of the ACM*, vol. 48, no. 10, pp. 83-90.

Adomavicius, G. & Tuzhilin, A. 2005b, 'Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734-749.

Adomavicius, G. & Tuzhilin, A. 2011, 'Context-aware recommender systems', in F. Ricci, L. Rokach, B. Shapira & P.B. Kantor (eds), *Recommender Systems Handbook*, Springer US, pp. 217-253.

Akutsu, T. & Halldórsson, M.M. 2000, 'On the approximation of largest common subtrees and largest common point sets', *Theoretical Computer Science*, vol. 233, no. 1, pp. 33-50.

Al-hassan, M., Lu, H. & Lu, J. 2009, 'A framework for delivering personalized e-government services from a citizen-centric approach', *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, ACM, Kuala Lumpur, Malaysia, pp. 436-440.

Al-hassan, M., Lu, H. & Lu, J. 2011, 'Personalized e-government services: tourism recommender system framework', in J. Filipe & J. Cordeiro (eds), *Web*

Information Systems and Technologies, vol. 75, Springer Berlin Heidelberg, pp. 173-187.

Al-Shamri, M.Y.H. & Bharadwaj, K.K. 2008, 'Fuzzy-genetic approach to recommender systems based on a novel hybrid user model', *Expert Systems with Applications*, vol. 35, no. 3, pp. 1386-1399.

Albadvi, A. & Shahbazi, M. 2009, 'A hybrid recommendation technique based on product category attributes', *Expert Systems with Applications*, vol. 36, no. 9, pp. 11480-11488.

Allen, J.F. 1983, 'Maintaining knowledge about temporal intervals', *Communications of the ACM*, vol. 26, no. 11, pp. 832-843.

Amatriain, X., Jaimes, A., Oliver, N. & Pujol, J. 2011, 'Data mining methods for recommender systems', in F. Ricci, L. Rokach, B. Shapira & P.B. Kantor (eds), *Recommender Systems Handbook*, Springer US, pp. 39-71.

Amoroso, D.L. & Reinig, B.A. 2004, 'Personalization management systems: minitrack introduction', *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, vol. Track 7, Big Island, Hawaii.

Armstrong, R., Freitag, D., Joachims, T. & Mitchell, T. 1995, 'Webwatcher: a learning apprentice for the world wide web', *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, pp. 6-12.

Bahrainian, S.A., Bahrainian, S.M., Salarinasab, M. & Dengel, A. 2010, 'Implementation of an intelligent product recommender system in an e-store', *Active Media Technology*, Springer, pp. 174-182.

Balabanovic, M. & Shoham, Y. 1997, 'Fab: content-based, collaborative recommendation', *Communications of the ACM*, vol. 40, no. 3, pp. 66-72.

Basu, C., Hirsh, H. & Cohen, W. 1998, 'Recommendation as classification: using social and content-based information in recommendation', *AAAI/IAAI*, pp. 714-720.

Batet, M., Moreno, A., Sánchez, D., Isern, D. & Valls, A. 2012, 'Turist@: Agent-based personalised recommendation of tourist activities', *Expert Systems with Applications*, vol. 39, no. 8, pp. 7319-7329.

Bellogin, A., Cantador, I., Diez, F., Castells, P. & Chavarriaga, E. 2013, 'An empirical comparison of social, collaborative filtering, and hybrid recommenders', *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 1, pp. 1-29.

Ben-Shimon, D., Tsikinovsky, A., Rokach, L., Meisles, A., Shani, G. & Naamani, L. 2007, 'Recommender system from personal social networks', *Advances in Intelligent Web Mastering*, Springer, pp. 47-55.

Berka, T., Behrendt, W., Gams, E. & Reich, S. 2002, 'Recommending internet-domains using trails and neural networks', in P. Bra, P. Brusilovsky & R. Conejo (eds), *Adaptive Hypermedia and Adaptive Web-Based Systems*, vol. 2347, Springer Berlin Heidelberg, pp. 368-371.

Bhavsar, V.C., Boley, H. & Yang, L. 2004, 'A weighted-tree similarity algorithm for multi-agent systems in e-business environments', *Computational Intelligence*, vol. 20, no. 4, pp. 1-20.

Biletskiy, Y., Baghi, H., Keleberda, I. & Fleming, M. 2009, 'An adjustable personalization of search and delivery of learning objects to learners', *Expert Systems with Applications*, vol. 36, no. 5, pp. 9113-9120.

Bille, P. 2005, 'A survey on tree edit distance and related problems', *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 217-239.

Billsus, D. & Pazzani, M. 2000, 'User modeling for adaptive news access', *User Modeling and User-Adapted Interaction*, vol. 10, no. 2-3, pp. 147-180.

Bivainis, J. 2006, 'Development of business partner selection', *Economics*, vol. 73, no. 1, pp. 7-18.

Bobadilla, J., Ortega, F., Hernando, A. & Alcalá, J. 2011, 'Improving collaborative filtering recommender system results and performance using genetic algorithms', *Knowledge-Based Systems*, vol. 24, no. 8, pp. 1310-1316.

Bobadilla, J., Ortega, F., Hernando, A. & Gutiérrez, A. 2013, 'Recommender systems survey', *Knowledge-Based Systems*, vol. 46, no. 0, pp. 109-132.

Bobadilla, J., Serradilla, F. & Hernando, A. 2009, 'Collaborative filtering adapted to recommender systems of e-learning', *Knowledge-Based Systems*, vol. 22, no. 4, pp. 261-265.

Born, M., Filipowska, A., Kaczmarek, M., Markovic, I., Starzecka, M. & Walczak, A. 2008, 'Business functions ontology and its application in semantic business process modelling', *19th Australasian Conference on Information Systems (ACIS 2008)*, Christchurch, pp. 136-145.

Breese, J.S., Heckerman, D. & Kadie, C. 1998, 'Empirical analysis of predictive algorithms for collaborative filtering', *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, Morgan Kaufmann, San Francisco, CA, pp. 43-52.

Burkard, R.E., Dell'Amico, M. & Martello, S. 2009, *Assignment Problems*, Society for Industrial and Applied Mathematics, Philadelphia.

Burke, R. 1999, 'The wasabi personal shopper: a case-based recommender system', *Proceedings of the 11th National Conference on Innovative Applications of Artificial Intelligence*, John Wiley & Sons, pp. 844-849.

Burke, R. 2000, 'Knowledge-based recommender systems', *Encyclopedia of Library and Information Systems*, vol. 69, no. 32, pp. 175-186.

Burke, R. 2002, 'Hybrid recommender systems: survey and experiments', *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331-370.

Burke, R. 2007, 'Hybrid web recommender systems', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web*, vol. 4321, Springer-Verlag, Berlin Heidelberg, pp. 377-408.

Cantador, I. 2008, 'Exploiting the conceptual space in hybrid recommender systems: a semantic-based approach', Universidad Autonoma de Madrid.

Cao, Y. & Li, Y. 2007, 'An intelligent fuzzy-based recommendation system for consumer electronic products', *Expert Systems with Applications*, vol. 33, no. 1, pp. 230-240.

Carmagnola, F., Vernero, F. & Grillo, P. 2009, 'Sonars: a social networks-based algorithm for social recommender systems', *User Modeling, Adaptation, and Personalization*, Springer, pp. 223-234.

Cayley, A. 1857, 'On the theory of the analytical forms called trees', *Philosophical Magazine Series 4*, vol. 13, no. 85, pp. 172-176.

Chen, C.-M. & Duh, L.-J. 2008, 'Personalized web-based tutoring system based on fuzzy item response theory', *Expert Systems with Applications*, vol. 34, no. 4, pp. 2298-2315.

Chen, C.-M., Duh, L.-J. & Liu, C.-Y. 2004, 'A personalized courseware recommendation system based on fuzzy item response theory', *2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE '04)*, IEEE, pp. 305-308.

Chen, W. 2001, 'New algorithm for ordered tree-to-tree correction problem', *Journal of Algorithms*, vol. 40, no. 2, pp. 135-158.

Chen, Z., Meng, X., Zhu, B. & Fowler, R.H. 2000, 'WebSail: from on-line learning to web search', *Proceedings of the First International Conference on Web Information Systems Engineering, 2000.*, vol. 1, pp. 206-213 vol.201.

Cho, Y.H. & Kim, J.K. 2004, 'Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce', *Expert Systems with Applications*, vol. 26, no. 2, pp. 233-246.

Christakou, C., Vrettos, S. & Stafylopatis, A. 2007, 'A hybrid movie recommender system based on neural networks', *International Journal on Artificial Intelligence Tools*, vol. 16, no. 05, pp. 771-792.

Clarke, T. 2013, 'The advance of the MOOCs (massive open online courses): the impending globalisation of business education?', *Education + Training*, vol. 55, no. 4/5, pp. 403-413.

Cornelis, C., Guo, X., Lu, J. & Zhang, G. 2005, 'A fuzzy relational approach to event recommendation', *Proceedings of the Second Indian International Conference on Artificial Intelligence (IICAI-05)*, Pune, INDIA, pp. 2231-2242.

Cornelis, C., Lu, J., Guo, X. & Zhang, G. 2007, 'One-and-only item recommendation with fuzzy logic techniques', *Information Sciences*, vol. 177, no. 22, pp. 4906-4921.

Crespo, R.G., Martínez, O.S., Lovelle, J.M.C., García-Bustelo, B., Gayo, J.E.L. & Pablos, P.O.d. 2011, 'Recommendation system based on user interaction data applied to intelligent electronic books', *Computers in Human Behavior*, vol. 27, no. 4, pp. 1445-1449.

de Campos, L.M., Fernández-Luna, J.M. & Huete, J.F. 2008, 'A collaborative recommender system based on probabilistic inference from fuzzy observations', *Fuzzy Sets and Systems*, vol. 159, no. 12, pp. 1554-1576.

De Leeuw, J. & Meijer, E. 2008, *Handbook of Multilevel Analysis*, Springer Verlag.

De Meo, P., Quattrone, G., Terracina, G. & Ursino, D. 2007, 'Utilization of intelligent agents for supporting citizens in their access to e-government services', *Web Intelligence and Agent Systems*, vol. 5, no. 3, pp. 273-310.

De Meo, P., Quattrone, G. & Ursino, D. 2008, 'A decision support system for designing new services tailored to citizen profiles in a complex and distributed e-government scenario', *Data & Knowledge Engineering*, vol. 67, no. 1, pp. 161-184.

Deshpande, M. & Karypis, G. 2004, 'Item-based top-N recommendation algorithms', *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 143-177.

Dougiamas, M. & Taylor, P. 2003, 'Moodle: using learning communities to create an open source course management system', *World Conference on Educational Multimedia, Hypermedia and Telecommunications*, vol. 2003, pp. 171-178.

Drachsler, H., Hummel, H. & Koper, R. 2008a, 'Identifying the goal, user model and conditions of recommender systems for formal and informal learning', *Journal of Digital Information*, vol. 10, no. 2, pp. 4-24.

Drachsler, H., Hummel, H.G.K. & Koper, R. 2008b, 'Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model', *International Journal of Learning Technology*, vol. 3, no. 4, pp. 404-423.

Drachsler, H., Pecceu, D., Arts, T., Hutten, E., Rutledge, L., van Rosmalen, P., Hummel, H. & Koper, R. 2010, 'Remashed -- an usability study of a recommender system for mash-ups for learning', *International Journal of Emerging Technologies in Learning*, vol. Supplement, no. S1, pp. 7-11.

Fan, B., Liu, L., Li, M. & Wu, Y. 2008, 'Knowledge recommendation based on social network theory', *2008 IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE 2008)*, IEEE, pp. 1-3.

Fankhauser, S., Riesen, K. & Bunke, H. 2011, 'Speeding up graph edit distance computation through fast bipartite matching', in X. Jiang, M. Ferrer & A. Torsello (eds), *Graph-Based Representations in Pattern Recognition*, vol. 6658, Springer Berlin Heidelberg, pp. 102-111.

Farzan, R. & Brusilovsky, P. 2006, 'Social navigation support in a course recommendation system', in V. Wade, H. Ashman & B. Smyth (eds), *Adaptive Hypermedia and Adaptive Web-Based Systems*, vol. 4018, Springer Berlin Heidelberg, pp. 91-100.

Felfernig, A., Friedrich, G., Jannach, D. & Zanker, M. 2006, 'An integrated environment for the development of knowledge-based recommender applications', *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 11-34.

Felfernig, A., Gula, B., Leitner, G., Maier, M., Melcher, R. & Teppan, E. 2008, 'Persuasion in knowledge-based recommendation', in H. Oinas-Kukkonen, P. Hasle, M. Harjumaa, K. Segerståhl & P. Øhrstrøm (eds), *Persuasive Technology*, vol. 5033, Springer-Verlag, Berlin Heidelberg, pp. 71-82.

Gallupe, R.B. 2007, 'The tyranny of methodologies in information systems research', *SIGMIS Database*, vol. 38, no. 3, pp. 20-28.

Garfinkel, R., Gopal, R., Tripathi, A. & Yin, F. 2006, 'Design of a shopbot and recommender system for bundle purchases', *Decision Support Systems*, vol. 42, no. 3, pp. 1974-1986.

Ghazanfar, M.A. & Prügel-Bennett, A. 2014, 'Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems', *Expert Systems with Applications*, vol. 41, no. 7, pp. 3261-3275.

Golbeck, J. 2006, 'Combining provenance with trust in social networks for semantic web content filtering', *Provenance and Annotation of Data*, Springer, pp. 101-108.

Golbeck, J. & Hendler, J. 2006, 'Filmtrust: movie recommendations using trust in web-based social networks', *Proceedings of the IEEE Consumer Communications and Networking Conference*, vol. 96, Citeseer, pp. 282-286.

Golbeck, J.A. 2005, 'Computing and applying trust in web-based social networks', PhD thesis, University of Maryland.

Goldberg, D., Nichols, D., Oki, B.M. & Terry, D. 1992, 'Using collaborative filtering to weave an information tapestry', *Communications of the ACM*, vol. 35, no. 12, pp. 61-70.

Gonçalves, M.A., Fox, E.A., Watson, L.T. & Kipp, N.A. 2004, 'Streams, structures, spaces, scenarios, societies (5s): a formal model for digital libraries', *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 2, pp. 270-312.

Gorla, J., Lathia, N., Robertson, S. & Wang, J. 2013, 'Probabilistic group recommendation via information matching', *Proceedings of the 22nd International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, Rio de Janeiro, Brazil, pp. 495-504.

Guarino, N. & Giaretta, P. 1995, 'Ontologies and knowledge bases: towards a terminological clarification', in N.J.I. Mars (ed.), *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pp. 25-32.

Guo, X. & Lu, J. 2007, 'Intelligent e-government services with personalized recommendation techniques', *International Journal of Intelligent Systems*, vol. 22, no. 5, pp. 401-417.

He, J. & Chu, W. 2010, 'A social network-based recommender system (SNRS)', in N. Memon, J.J. Xu, D.L. Hicks & H. Chen (eds), *Data Mining for Social Network Data*, vol. 12, Springer US, pp. 47-74.

Herlocker, J., Konstan, J.A. & Riedl, J. 2002, 'An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms', *Information Retrieval*, vol. 5, no. 4, pp. 287-310.

Herlocker, J.L., Konstan, J.A., Borchers, A. & Riedl, J. 1999, 'An algorithmic framework for performing collaborative filtering', *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, Berkeley, California, United States, pp. 230-237.

Herrera, F. & Martinez, L. 2000, 'A 2-tuple fuzzy linguistic representation model for computing with words', *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 746-752.

Herrera, F. & Martinez, L. 2001, 'A model based on linguistic 2-tuples for dealing with multigranular hierarchical linguistic contexts in multi-expert decision-making', *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 2, pp. 227-234.

Hsu, S., Wen, M.-H., Lin, H.-C., Lee, C.-C. & Lee, C.-H. 2007, 'AIMED - a personalized TV recommendation system', in P. Cesar, K. Chorianopoulos & J. Jensen (eds), *Interactive TV: a Shared Experience*, vol. 4471, Springer Berlin Heidelberg, pp. 166-174.

Huang, Z., Chung, W. & Chen, H. 2004, 'A graph model for e-commerce recommender systems', *Journal of the American Society for Information Science and Technology*, vol. 55, no. 3, pp. 259-274.

Huang, Z., Zeng, D. & Chen, H. 2007, 'A comparison of collaborative-filtering recommendation algorithms for e-commerce', *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68-78.

Hung, L.-p. 2005, 'A personalized recommendation system based on product taxonomy for one-to-one marketing online', *Expert Systems with Applications*, vol. 29, no. 2, pp. 383-392.

Hwang, C.-S. & Chen, Y.-P. 2007, 'Using trust in collaborative filtering recommendation', *New Trends in Applied Artificial Intelligence*, Springer, pp. 1052-1060.

Jameson, A., Baldes, S. & Kleinbauer, T. 2004, 'Two methods for enhancing mutual awareness in a group recommender system', *Proceedings of the Working Conference on Advanced Visual Interfaces*, ACM, Gallipoli, Italy, pp. 447-449.

Jameson, A. & Smyth, B. 2007, 'Recommendation to groups', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web, Methods and Strategies of Web Personalization*, vol. 4321, Springer Berlin Heidelberg, pp. 596-627.

Jia, R., Jin, M. & Liu, C. 2010, 'A new clustering method for collaborative filtering', *2010 International Conference on Networking and Information Technology (ICNIT)*, pp. 488-492.

Jonker, R. & Volgenant, A. 1987, 'A shortest augmenting path algorithm for dense and sparse linear assignment problems', *Computing*, vol. 38, no. 4, pp. 325-340.

Jungnickel, D. 2008, *Graphs, Networks, and Algorithms*, 3rd edn, Springer-Verlag, Berlin Heidelberg.

Kailing, K., Kriegel, H.-P., Schönauer, S. & Seidl, T. 2004, 'Efficient similarity search for hierarchical data in large databases', in E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm & E. Ferrari (eds), *Advances in Database Technology - EDBT 2004*, vol. 2992, Springer Berlin Heidelberg, pp. 676-693.

Kim, H.-N., Ji, A.-T., Ha, I. & Jo, G.-S. 2010, 'Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation', *Electronic Commerce Research and Applications*, vol. 9, no. 1, pp. 73-83.

Kim, K.-j. & Ahn, H. 2005, 'Using a clustering genetic algorithm to support customer segmentation for personalized recommender systems', in T. Kim (ed.), *Artificial Intelligence and Simulation*, vol. 3397, Springer Berlin Heidelberg, pp. 409-415.

Kim, K.-j. & Ahn, H. 2008, 'A recommender system using GA K-means clustering in an online shopping market', *Expert Systems with Applications*, vol. 34, no. 2, pp. 1200-1209.

Klein, P. 1998, 'Computing the edit-distance between unrooted ordered trees', in G. Bilardi, G. Italiano, A. Pietracaprina & G. Pucci (eds), *Algorithms — ESA' 98*, vol. 1461, Springer Berlin / Heidelberg, pp. 1-1.

Kuhn, H.W. 1955, 'The Hungarian method for the assignment problem', *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83-97.

Lawrence, R.D., Almasi, G.S., Kotlyar, V., Viveros, M.S. & Duri, S.S. 2001, 'Personalization of supermarket product recommendations', *Data Mining and Knowledge Discovery*, vol. 5, no. 1-2, pp. 11-32.

Leacock, C. & Chodorow, M. 1998, 'Combining local context and wordNet similarity for word sense identification', in C. Fellbaum (ed.), *Wordnet: An Electronic Lexical Database*, MIT Press, Cambridge, pp. 265 – 283.

Lee, T., Chun, J., Shim, J. & Lee, S.-g. 2006, 'An ontology-based product recommender system for B2B marketplaces', *International Journal of Electronic Commerce*, vol. 11, no. 2, pp. 125-155.

Leung, C., Chan, S. & Chung, F. 2006, 'A collaborative filtering framework based on fuzzy association rules and multiple-level similarity', *Knowledge and Information Systems*, vol. 10, no. 3, pp. 357-381.

Leung, W.-k.C. 2009, 'Enriching user and item profiles for collaborative filtering: from concept hierarchies to user-generated reviews', The Hong Kong Polytechnic University, Hong Kong.

Lin, Z., Wang, H., McClean, S. & Liu, C. 2008, 'All common embedded subtrees for measuring tree similarity', *International Symposium on Computational Intelligence and Design*, vol. 1, pp. 29-32.

Linden, G., Smith, B. & York, J. 2003, 'Amazon.com recommendations: item-to-item collaborative filtering', *Internet Computing, IEEE*, vol. 7, no. 1, pp. 76-80.

Lopes, G.R., Moro, M.M., Wives, L.K. & De Oliveira, J.P.M. 2010, 'Collaboration recommendation on academic social networks', *Advances in Conceptual Modeling–Applications and Challenges*, Springer, pp. 190-199.

Lu, C., Su, Z.-Y. & Tang, C. 2001, 'A new measure of edit distance between labeled trees', in J. Wang (ed.), *Computing and Combinatorics*, vol. 2108, Springer-Verlag, Berlin Heidelberg, pp. 338-348.

Lu, J. 2004, 'Personalized e-learning material recommender system', *Proceedings of the 2nd International Conference on Information Technology for Application (ICITA 2004)*, pp. 374-379.

Lu, J., Guo, X., Huynh, A.T. & Benkovich, L. 2004, 'SRS: a subject recommender system to enhance e-learning personalisation', *Proceedings of the Third International Conference on Information (Info'2004)*, eds L. Li & K.K. Yen, International Information Institute, Hosei University, Tokyo, Japan, pp. 253-256.

Lu, J., Shambour, Q., Xu, Y., Lin, Q. & Zhang, G. 2010, 'BizSeeker: a hybrid semantic recommendation system for personalized government-to-business e-services', *Internet Research*, vol. 20, no. 3, pp. 342-365.

Lu, J., Shambour, Q., Xu, Y., Lin, Q. & Zhang, G. 2013, 'A web-based personalized business partner recommendation system using fuzzy semantic techniques', *Computational Intelligence*, vol. 29, no. 1, pp. 37-69.

Lu, J., Shambour, Q. & Zhang, G. 2009, 'Recommendation technique-based government-to-business personalized e-services', *The 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009)*, pp. 1-6.

Ma, H., Yang, H., Lyu, M.R. & King, I. 2008, 'Sorec: social recommendation using probabilistic matrix factorization', *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ACM, pp. 931-940.

Maâtallah, M. & Seridi, H. 2012, 'Enhanced collaborative filtering to recommender systems of technology enhanced learning', *ICWIT 2012*, pp. 129-138.

Markellou, P., Mousourouli, I., Sirmakessis, S. & Tsakalidis, A. 2005, 'Personalized e-commerce recommendations', *2005 IEEE International Conference on e-Business Engineering*, Beijing, China, pp. 245-252.

Masthoff, J. 2011, 'Group recommender systems: combining individual models', in F. Ricci, L. Rokach, B. Shapira & P.B. Kantor (eds), *Recommender Systems Handbook*, Springer US, pp. 677-702.

McCarthy, K., Reilly, J., McGinty, L. & Smyth, B. 2004, 'Thinking positively - explanatory feedback for conversational recommender systems', *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04) Explanation Workshop*, pp. 115-124.

McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B. & Nixon, P. 2006, 'CATS: a synchronous approach to collaborative group recommendation', *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, AAAI Press, Melbourne Beach, Florida, pp. 86-91.

Melnik, S., Garcia-Molina, H. & Rahm, E. 2002, 'Similarity flooding: a versatile graph matching algorithm and its application to schema matching', *Proceedings of 18th International Conference on Data Engineering, 2002*, pp. 117-128

Melville, P., Mooney, R.J. & Nagarajan, R. 2002, 'Content-boosted collaborative filtering for improved recommendations', *Eighteenth National Conference on Artificial intelligence*, American Association for Artificial Intelligence, Edmonton, Alberta, Canada, pp. 187-192.

Middleton, S., Roure, D. & Shadbolt, N. 2009, 'Ontology-based recommender systems', in S. Staab & R. Studer (eds), *Handbook on Ontologies*, Springer Berlin Heidelberg, pp. 779-796.

Mobasher, B., Jin, X. & Zhou, Y. 2004, 'Semantically enhanced collaborative filtering on the web', in B. Berendt, A. Hotho, D. Mladenič, M. Someren, M. Spiliopoulou & G. Stumme (eds), *Web Mining: From Web to Semantic Web*, vol. 3209, Springer Berlin Heidelberg, pp. 57-76.

Mooney, R.J. & Roy, L. 2000, 'Content-based book recommending using learning for text categorization', *Proceedings of the Fifth ACM Conference on Digital Libraries*, ACM, pp. 195-204.

Munkres, J. 1957, 'Algorithms for the assignment and transportation problems', *Journal of the Society for Industrial & Applied Mathematics*, vol. 5, no. 1, pp. 32-38.

Nanopoulos, A., Rafailidis, D., Symeonidis, P. & Manolopoulos, Y. 2010, 'Musicbox: personalized music recommendation based on cubic analysis of social tags', *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 407-412.

Niu, L. 2009, 'The cognition-driven decision process for business intelligence: a model and techniques', University of Technology, Sydney.

Núñez-Valdéz, E.R., Cueva Lovelle, J.M., Sanjuán Martínez, O., García-Díaz, V., Ordoñez de Pablos, P. & Montenegro Marín, C.E. 2012, 'Implicit feedback techniques on recommender systems applied to electronic books', *Computers in Human Behavior*, vol. 28, no. 4, pp. 1186-1193.

O'Donovan, J. 2009, 'Capturing trust in social web applications', in J. Golbeck (ed.), *Computing with Social Trust*, Springer London, pp. 213-257.

Oprea, A., Hornung, T., Ziegler, C.-N., Eggs, H. & Lausen, G. 2013, 'A hybrid B2B app recommender system', in F. Daniel, P. Dolog & Q. Li (eds), *Web Engineering*, vol. 7977, Springer Berlin Heidelberg, pp. 490-493.

Ouangraoua, A. & Ferraro, P. 2009, 'A constrained edit distance algorithm between semi-ordered trees', *Theoretical Computer Science*, vol. 410, no. 8-10, pp. 837-846.

Palau, J., Montaner, M., López, B. & De La Rosa, J.L. 2004, 'Collaboration analysis in recommender systems using social networks', *Cooperative Information Agents VIII*, Springer, pp. 137-151.

Pazzani, M. 1999, 'A framework for collaborative, content-based and demographic filtering', *Artificial Intelligence Review*, vol. 13, no. 5-6, pp. 393-408.

Pazzani, M. & Billsus, D. 2007, 'Content-based recommendation systems', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web*, vol. 4321, Springer-Verlag, Berlin Heidelberg, pp. 325-341.

Popescu, G. 2013, 'Group recommender systems as a voting problem', in A.A. Ozok & P. Zaphiris (eds), *Online Communities and Social Computing*, vol. 8029, Springer Berlin Heidelberg, pp. 412-421.

Porcel, C. & Herrera-Viedma, E. 2010, 'Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries', *Knowledge-Based Systems*, vol. 23, no. 1, pp. 32-39.

Porcel, C., López-Herrera, A.G. & Herrera-Viedma, E. 2009, 'A recommender system for research resources based on fuzzy linguistic modeling', *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 5173-5183.

Porcel, C., Moreno, J.M. & Herrera-Viedma, E. 2009, 'A multi-disciplinar recommender system to advice research resources in university digital libraries', *Expert Systems with Applications*, vol. 36, no. 10, pp. 12520-12528.

Pronk, V., Verhaegh, W., Proidl, A. & Tiemann, M. 2007, 'Incorporating user control into recommender systems based on naive bayesian classification', *Proceedings of the 2007 ACM Conference on Recommender Systems*, ACM, Minneapolis, MN, USA, pp. 73-80.

Quijano-Sánchez, L., Bridge, D., Díaz-Agudo, B. & Recio-García, J. 2012, 'A case-based solution to the cold-start problem in group recommenders', in B. Agudo & I.

Watson (eds), *Case-Based Reasoning Research and Development*, vol. 7466, Springer Berlin Heidelberg, pp. 342-356.

Quijano-Sanchez, L., Recio-Garcia, J.A., Diaz-Agudo, B. & Jimenez-Diaz, G. 2013, 'Social factors in group recommender systems', *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 1, pp. 1-30.

Ramaswamy, L., Deepak, P., Polavarapu, R., Gunasekera, K., Garg, D., Visweswariah, K. & Kalyanaraman, S. 2009, 'Caesar: a context-aware, social recommender system for low-end mobile devices', *Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'09)*, IEEE, pp. 338-347.

Renda, M.E. & Straccia, U. 2005, 'A personalized collaborative digital library environment: a model and an application', *Information Processing & Management*, vol. 41, no. 1, pp. 5-21.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. 1994, 'GroupLens: an open architecture for collaborative filtering of netnews', *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, ACM, Chapel Hill, North Carolina, United States, pp. 175-186.

Resnick, P. & Varian, H.R. 1997, 'Recommender systems', *Communications of the ACM*, vol. 40, no. 3, pp. 56-58.

Resnik, P. 1995, 'Using information content to evaluate semantic similarity in a taxonomy', *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 1, Morgan Kaufmann Publishers Inc., Montreal, Quebec, Canada, pp. 448-453.

Ricci, F. & Senter, L. 1998, 'Structured cases, trees and efficient retrieval', *Advances in Case-Based Reasoning*, vol. 1488, pp. 88-99.

Richter, T. 1997, 'A new algorithm for the ordered tree inclusion problem', in A. Apostolico & J. Hein (eds), *Combinatorial Pattern Matching*, vol. 1264, Springer Berlin / Heidelberg, pp. 150-166.

Riesen, K. & Bunke, H. 2009, 'Approximate graph edit distance computation by means of bipartite graph matching', *Image and Vision Computing*, vol. 27, no. 7, pp. 950-959.

Rodríguez, R.M., Espinilla, M., Sánchez, P.J. & Martínez-López, L. 2010, 'Using linguistic incomplete preference relations to cold start recommendations', *Internet Research*, vol. 20, no. 3, pp. 296-315.

Ruiz-Montiel, M. & Aldana-Montes, J. 2009, 'Semantically enhanced recommender systems', in R. Meersman, P. Herrero & T. Dillon (eds), *On the Move to Meaningful Internet Systems: OTM 2009 Workshops*, vol. 5872, Springer-Verlag, Berlin Heidelberg, pp. 604-609.

Sakawa, M. 1993, *Fuzzy Sets and Interactive Multiobjective Optimization*, Plenum New York.

Salehi, M. & Kamalabadi, I.N. 2013, 'Hybrid recommendation approach for learning material based on sequential pattern of the accessed material and the learner's preference tree', *Knowledge-Based Systems*, vol. 48, no. 0, pp. 57-69.

Salehi, M. & Kmalabadi, I.N. 2012, 'A hybrid attribute–based recommender system for e–learning material recommendation', *IERI Procedia*, vol. 2, no. 0, pp. 565-570.

Sanz, I., Mesiti, M., Guerrini, G. & Berlanga, R. 2008, 'Fragment-based approximate retrieval in highly heterogeneous XML collections', *Data & Knowledge Engineering*, vol. 64, no. 1, pp. 266-293.

Sanz, I., Pérez, M. & Berlanga, R. 2008, 'Designing similarity measures for XML', in Q. Li, S. Spaccapietra, E. Yu & A. Olivé (eds), *Conceptual Modeling - ER 2008*, vol. 5231, Springer Berlin Heidelberg, pp. 514-515.

Sarwar, B., Karypis, G., Konstan, J. & Reidl, J. 2001, 'Item-based collaborative filtering recommendation algorithms', *Proceedings of the 10th International Conference on World Wide Web*, ACM, pp. 285-295.

Schafer, J., Frankowski, D., Herlocker, J. & Sen, S. 2007, 'Collaborative filtering recommender systems', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web*, vol. 4321, Springer-Verlag, Berlin Heidelberg, pp. 291-324.

Schafer, J.B., Konstan, J. & Riedl, J. 2001, 'E-commerce recommendation applications', in R. Kohavi & F. Provost (eds), *Applications of Data Mining to Electronic Commerce*, Springer US, pp. 115-153.

Schickel-Zuber, V. & Faltings, B. 2007, 'OSS: a semantic similarity function based on hierarchical ontologies', *the 2007 International Joint Conference on Artificial Intelligence* vol. 7, Hyderabad, India, p. 6.

Serrano-Guerrero, J., Herrera-Viedma, E., Olivas, J.A., Cerezo, A. & Romero, F.P. 2011, 'A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0', *Information Sciences*, vol. 181, no. 9, pp. 1503-1516.

Sevarac, Z., Devedzic, V. & Jovanovic, J. 2012, 'Adaptive neuro-fuzzy pedagogical recommender', *Expert Systems with Applications*, vol. 39, no. 10, pp. 9797-9806.

Shambour, Q. & Lu, J. 2010, 'A framework of hybrid recommendation system for government-to-business personalized e-services', *Seventh International Conference on Information Technology: New Generations (ITNG)*, pp. 592-597.

Shambour, Q. & Lu, J. 2011a, 'Government-to-business personalized e-services using semantic-enhanced recommender system', in K. Andersen, E. Francesconi, Å. Grönlund & T. Engers (eds), *Electronic Government and the Information Systems Perspective*, vol. 6866, Springer Berlin Heidelberg, pp. 197-211.

Shambour, Q. & Lu, J. 2011b, 'A hybrid multi-criteria semantic-enhanced collaborative filtering approach for personalized recommendations', *2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 1, pp. 71-78.

Shambour, Q. & Lu, J. 2011c, 'A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services', *International Journal of Intelligent Systems*, vol. 26, no. 9, pp. 814-843.

Shambour, Q. & Lu, J. 2012, 'A trust-semantic fusion-based recommendation approach for e-business applications', *Decision Support Systems*, vol. 54, no. 1, pp. 768-780.

Shambour, Q.Y. 2012, 'Hybrid recommender systems for personalized government-to-business e-services', University of Technology, Sydney.

Shardanand, U. & Maes, P. 1995, 'Social information filtering: algorithms for automating "word of mouth"', *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press/Addison-Wesley Publishing Co., Denver, Colorado, United States, pp. 210-217.

Shiratsuchi, K., Yoshii, S. & Furukawa, M. 2006, 'Finding unknown interests utilizing the wisdom of crowds in a social bookmark service', *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, IEEE Computer Society, pp. 421-424.

Shishehchi, S., Banihashem, S.Y., Zin, N.A.M. & Noah, S.A.M. 2011, 'Review of personalized recommendation techniques for learners in e-learning systems', *2011 International Conference on Semantic Technology and Information Retrieval (STAIR)* IEEE, pp. 277-281.

Shvaiko, P. & Euzenat, J. 2013, 'Ontology matching: state of the art and future challenges', *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 158-176.

Smyth, B. 2007, 'Case-based recommendation', in P. Brusilovsky, A. Kobsa & W. Nejdl (eds), *The Adaptive Web*, vol. 4321, Springer Berlin Heidelberg, pp. 342-376.

Smyth, B. & Cotter, P. 2000, 'A personalised TV listings service for the digital TV age', *Knowledge-Based Systems*, vol. 13, no. 2–3, pp. 53-59.

Sobhanam, H. & Mariappan, A. 2013, 'A hybrid approach to solve cold start problem in recommender systems using association rules and clustering technique', *International Journal of Computer Applications*, vol. 74, no. 4, pp. 17-23.

Solskinnsbakk, G., Gulla, J.A., Haderlein, V., Myrseth, P. & Cerrato, O. 2012, 'Quality of hierarchies in ontologies and folksonomies', *Data & Knowledge Engineering*, vol. 74, no. 0, pp. 13-25.

Tai, K.-C. 1979, 'The tree-to-tree correction problem', *Journal of the Association for Computing Machinery (JACM)*, vol. 26, no. 3, pp. 422-433.

Tan, S., Bu, J., Chen, C. & He, X. 2011, 'Using rich social media information for music recommendation via hypergraph model', in S.C.H. Hoi, J. Luo, S. Boll, D. Xu, R. Jin & I. King (eds), *Social Media Modeling and Computing*, Springer London, pp. 213-237.

Terán, L., Ladner, A., Fivaz, J. & Gerber, S. 2012, 'Using a fuzzy-based cluster algorithm for recommending candidates in e-elections', *Fuzzy Methods for Customer Relationship Management and Marketing: Applications and Classifications*, vol. 6, pp. 115-138.

Terán, L. & Meier, A. 2010, 'A fuzzy recommender system for eElections', in K. Andersen, E. Francesconi, Å. Grönlund & T. van Engers (eds), *Electronic Government and the Information Systems Perspective*, vol. 6267, Springer Berlin Heidelberg, pp. 62-76.

Terán, L. & Meier, A. 2011, 'Smart participation – a fuzzy-based platform for stimulating citizens' participation', *International Journal for Infonomics (IJI)*, vol. 4, no. 3/4, pp. 501-512.

Torsello, A., Hidovic, D. & Pelillo, M. 2004, 'Four metrics for efficiently comparing attributed trees', *17th International Conference on Pattern Recognition (ICPR'04)*, vol. 2, pp. 467-470.

Tyler, S.K. & Zhang, Y. 2008, 'Open domain recommendation: social networks and collaborative filtering', *Advanced Data Mining and Applications*, Springer, pp. 330-341.

Vaishnavi, V.K. & Kuechler Jr, W. 2007, *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*, CRC Press.

Vygotskiĭ, L.L.S. 1978, *Mind in Society: The Development of Higher Psychological Processes*, Harvard University Press.

Wang, J.-C. & Chiu, C.-C. 2008, 'Recommending trusted online auction sellers using social network analysis', *Expert Systems with Applications*, vol. 34, no. 3, pp. 1666-1679.

Watters, A. 2013, 'Got MOOC? massive open online courses are poised to change the face of education', *School Library Journal*, vol. 59, no. 4, pp. 28-31.

Wei, K., Huang, J. & Fu, S. 2007, 'A survey of e-commerce recommender systems', *IEEE International Conference on Service Systems and Service Management*, pp. 1-5.

Woerndl, W. & Groh, G. 2007, 'Utilizing physical and social context to improve recommender systems', *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*, IEEE Computer Society, pp. 123-128.

Xin, W., Jamaliding, Q. & Okamoto, T. 2009, 'Discovering social network to improve recommender system for group learning support', *2009 International Conference on Computational Intelligence and Software Engineering (CiSE)*, pp. 1-4.

Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y. & Chen, Z. 2005, 'Scalable collaborative filtering using cluster-based smoothing', *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, Salvador, Brazil, pp. 114-121.

Xue, Y., Wang, C., Ghenniwa, H. & Shen, W. 2009, 'A tree similarity measuring method and its application to ontology comparison', *Journal of Universal Computer Science*, vol. 15, no. 9, pp. 1766-1781.

Yager, R.R. 2003, 'Fuzzy logic methods in recommender systems', *Fuzzy Sets and Systems*, vol. 136, no. 2, pp. 133-149.

Yan, Q. 2008, 'Modeling and simulation of instant messageing on internet', *2008 International Conference on Computer Science and Information Technology (ICCSIT '08)*, pp. 841-844.

Yang, L., Sarker, B., Bhavsar, V. & Boley, H. 2005, 'A weighted-tree simplicity algorithm for similarity matching of partial product descriptions', *Proceedings of The International Society for Computers and Their Applications (ISCA) 14th International Conference on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005)*, Toronto, Ontario, Canada, pp. 55-60.

Yang, R., Kalnis, P. & Tung, A.K.H. 2005, 'Similarity evaluation on tree-structured data', *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, ACM, Baltimore, Maryland, pp. 754-765.

Yang, W.-S., Cheng, H.-C. & Dia, J.-B. 2008, 'A location-aware recommender system for mobile shopping environments', *Expert Systems with Applications*, vol. 34, no. 1, pp. 437-445.

Zadeh, L.A. 1965, 'Fuzzy sets', *Information and Control*, vol. 8, no. 3, pp. 338-353.

Zadeh, L.A. 1975, 'The concept of a linguistic variable and its application to approximate reasoning—I', *Information Sciences*, vol. 8, no. 3, pp. 199-249.

Zaiane, O.R. 2002, 'Building a recommender agent for e-learning systems', *Proceedings of 2002 International Conference on Computers in Education*, vol. 51, pp. 55-59

Zaıane, O.R. & Luo, J. 2001, 'Web usage mining for a better web-based learning environment', *Proceedings of Conference on Advanced Technology for Education*, Citeseer, pp. 60-64.

Zenebe, A. & Norcio, A.F. 2009, 'Representation, similarity measures and aggregation methods using fuzzy sets for content-based recommender systems', *Fuzzy Sets and Systems*, vol. 160, no. 1, pp. 76-94.

Zenebe, A., Zhou, L. & Norcio, A.F. 2010, 'User preferences discovery using fuzzy models', *Fuzzy Sets and Systems*, vol. 161, no. 23, pp. 3044-3063.

Zhang, G. 1998, *Fuzzy Number-Valued Measure Theory*, Tsinghua University Press, Beijing.

Zhang, G. & Lu, J. 2003, 'An integrated group decision-making method dealing with fuzzy preferences for alternatives and individual judgments for selection criteria', *Group Decision and Negotiation*, vol. 12, no. 6, pp. 501-515.

Zhang, G. & Lu, J. 2004, 'Using general fuzzy number to handle uncertainty and imprecision in group decision-making', *Intelligent Sensory Evaluation: Methodologies and Applications*, pp. 51-70.

Zhang, G. & Lu, J. 2009, 'A linguistic intelligent user guide for method selection in multi-objective decision support systems', *Information Sciences*, vol. 179, no. 14, pp. 2299-2308.

Zhang, K. 1993, 'A new editing based distance between unordered labeled trees', in A. Apostolico, M. Crochemore, Z. Galil & U. Manber (eds), *Combinatorial Pattern Matching*, vol. 684, Springer-Verlag, Berlin Heidelberg, pp. 254-265.

Zhang, K. 1996, 'A constrained edit distance between unordered labeled trees', *Algorithmica*, vol. 15, no. 3, pp. 205-222.

Zhang, K. & Shasha, D. 1989, 'Simple fast algorithms for the editing distance between trees and related problems', *SIAM Journal of Computing*, vol. 18, no. 6, pp. 1245-1262.

Zhang, K., Statman, R. & Shasha, D. 1992, 'On the editing distance between unordered labeled trees', *Information Processing Letters*, vol. 42, no. 3, pp. 133-139.

Zhang, L., Zhu, M. & Huang, W. 2009, 'A framework for an ontology-based e-commerce product information retrieval system', *Journal of Computers*, vol. 4, no. 6, pp. 436-443.

Zhang, X. & Wang, H. 2005, 'Study on recommender systems for business-to-business electronic commerce', *Communications of the IIMA*, vol. 5, no. 4, pp. 53-62.

Zhang, Z., Lin, H., Liu, K., Wu, D., Zhang, G. & Lu, J. 2013, 'A hybrid fuzzy-based personalized recommender system for telecom products/services', *Information Sciences*, vol. 235, no. 0, pp. 117-129.

Zhao, Y., Li, Z., Wang, X. & Halang, W.A. 2012, 'Decision support in e-business based on assessing similarities between ontologies', *Knowledge-Based Systems*, vol. 32, no. 0, pp. 47-55.

Ziegler, C.-N. & Lausen, G. 2004, 'Analyzing correlation between trust and user similarity in online communities', *Trust Management*, Springer, pp. 251-265.

Abbreviations

ANN	Artificial Neural Network
Austrade	Australian Trade Commission government agency
B2B	Business-to-Business
B2C	Business-to-Consumer
BPM	Basic Preference Module
CB	Content-Based
CBR	Case-Based Reasoning
CF	Collaborative Filtering
CI	Computational Intelligence
CPC	Constrained Pearson Correlation
CS	Cold-Start
E-CRM	Electronic Customer Relationship Management
ELRS	E-Learning Recommender System
FIRT	Fuzzy Item Response Theory
FLM	Fuzzy Linguistic Modelling
FSPR	Fuzzy Semantic Product Relevance
FTCP-RS	Fuzzy-Based Telecom Product Recommender System
FTM	Fuzzy Set Theoretic Method
G2B	Government-to-Business
G2C	Government-to-Citizen
GA	Genetic Algorithms
GRS	Group Recommender Systems
HFSR	Hybrid Fuzzy Semantic Recommendation
HSR	Hybrid Semantic Recommendation

IBCF	Item-Based CF
IBPL	Intelligent Business Partner Locator
IPS	International Partner Search
IUTH	Item Tree and User Request Tree-Based Hybrid
KB	Knowledge-Based
MAE	Mean Absolute Error
MCS	Maximum Common Sub-tree
PCRS	Personalized Courseware Recommender System
Pe-Gov	Personalized e-Government
PLRS	Personalized e-Learning Material Recommender System
PPM	Primal Preference Module
SNA	Social Networks Analysis
SRS	Subject e-Learning Recommender System
STEF	Smart Trade Exhibition Finder
TeCF	Trust-enhanced CF
UBCF	User-Based CF
UDL	University Digital Libraries
UI	User Interface
WPS	Wasabi Personal Shopper