# Design, Implementation and Evaluation of Advanced Recommendation Models in Mobile Tourist Information System

A thesis submitted in fulfilment
of the requirements for the degree of
**Master of Science**
in Computer Science
University of Waikato
by

**Saijai Junmanee**

Supervisor
Dr.Annika Hinze

THE UNIVERSITY OF
WAIKATO
*Te Whare Wānanga o Waikato*

2006

# Abstract

Providing personalized recommendations in a mobile tourist information system reveals the existing limitations of the recommendation algorithms that are currently employed. One of the most crucial limitations is the lack of user information needed to build a user model, especially at the beginning of the system usage. In this thesis, we adopt and extend the basic concepts of the three recommendation paradigms collaborative filtering, content-based, and knowledge-based recommendation. We propose our recommendation algorithms using a user's personal information (e.g., preferences, travel histories) to replace the missing information. The designed recommendation component and recommendation algorithms are embedded as a recommendation service in the Tourist Information Provider (TIP). The prototype has been evaluated on the effectiveness and performance of the proposed recommendation algorithms. The results of the evaluation are discussed, with reference to remedies for the existing limitations. Our conclusion is that our new algorithms effectively solve the lack of information problem. Furthermore, the results of the experiments on the performance of the algorithms illustrate some appealing aspects for further study. With regard to our improvements of the user interface, we have also evaluated (using a first-cut analysis) the user-system interaction. The analysis reveals that the new interface conveys clearly the reasoning used to select recommendations provided to users. This assists the users to feel a sense of control which will help in their decision making as to which sights to visit.

# Acknowledgement

I would like to sincerely express my feelings to all my supporters during this project. First of all, the year 2006 is very special for all Thai people as it is the 60th Anniversary of His Majesty the King Bhumiphol Adulyadej, Rama IV,'s Accession to the Throne. It is my honour to finish my master degree in this pleasing year. I would like to dedicate all the good credits I have in this project as an honourable present to His Majesty. *Long Live His Majesty the King Bhumiphol.*

I would like to thank my supervisor Dr. Annika Hinze who brings along the TIP system and puts on a lot of effort to extend it to become a knowledgeable research project. All the support, ideas and discussions she has given to me while I was working on this project made it a pleasing working environment.

I would like to express my gratitude to my dearest family, especially my mother and Jitsatha Thatavakorn who are always very supportive. With their love and encouragement, I can concentrate on and think positively no matter how hard and stressful a circumstance is.

My study in New Zealand was financially supported by (and would not have been possible without) the cooperation scheme between the New Zealand and the Thai Government through the NZAID study awards. I would like to thank Sonya Saunders, the scholarship administrator, for helping me out in every request I have made. Because of her help, my stay and study in New Zealand was safe and sound.

I also would like to thank the governor of the Electricity Generating Authority of Thailand (EGAT) who granted a three-year study leave and all my colleagues in the Production Efficiency Division who have taken care of my tasks during my stay in New Zealand.

I also would like to thank Bryan Genet for proof reading all parts of this thesis.

Many thanks to all the academic staff of the Computer Science Department for the knowledge they have given to me during my study and also the administrative staff, especially Pauline Wilson, for all their help and support. Thank you everyone in the Information Systems and Databases Group and the TIP development team, especially, Qui Quan who always stayed up late in the lab with me.

Last by not least, I would like to thank every New Zealander whom I have met during my stay. Your nice and warm hospitality brought me one of the greatest experiences in my life.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The development of advanced communication and computing systems helps us to easily get access to almost any electronic source of information in just one click. Nowadays, accessing electronic information sources even goes far beyond our desks. Advanced mobile information systems together with location based services allow us to get almost any kind of electronic information from anywhere at anytime. Using a mobile system, we do not have to sit in front of a computer in order to gather information from on-line information suppliers. On the contrary, we can receive the required information by interacting with our hand-held devices, e.g., a mobile phone, from our current location. Therefore, it is not a surprise to see someone sitting next to us in a hair salon booking movie tickets, checking their flight schedule or accessing the weather forecast channel via their tiny mobile phone.

As mobile devices decrease in size, weight, and price and increase in power, storage, connectivity and positioning capabilities, delivering information to a user based on their location or context has become important in many information system applications. The Scottish *Citylink* [1], a provider of Express Coach services in Scotland, has employed a service called *Txt ticketing*. The service provides a new way of delivering tickets to customers who buy tickets on-line or via a call center. In addition to the existing methods of delivery, the user also has the option of having their ticket reference sent to their mobile phone as a text message. The customers then show the reference code to the driver when boarding the coach. Another example of a mobile application is a free personalized notification service named *Alerts* offered by Yahoo.com [2]. A user subscribes to topics they are interested in, e.g., stock market, breaking news, weather, etc. The user then receives a notification whenever new content regarding their information inquiry becomes available. An alert message can be delivered via e-mail as well as pager, or mobile phone. These two applications have indicated a high potential for providing information to mobile users who usually reposition themselves during their information inquiry session. One prominent application area, where a user can fully take advantage of getting connected to the information sources and performing location-based information inquiries while they are moving around an area, is the tourism industry.

Using a mobile tourist information system, the tourist is no longer wan-

dering around a city with a tourist guide book. They also do not need to stop by a tourist information center for their information inquiry. They can get information based on their current location or context such as weather, ways of travel, etc., from their mobile devices. Similar to web sites, the mobile tourist information system provides information about the places in which the user is currently interested. Yet, the system also allows their users to relocate themselves while they are interacting to the system. Demanding information regarding their current context, such as weather and time, is also applicable. In addition, it gives recommendations about other places the user may consider visiting based on their preferences, their access history, their visit history or their current context [15, 19].

This chapter first introduces the general idea of recommendation services in mobile tourist information systems. Then we introduce the recommendation component employed in the current version of the Tourist Information Provider (TIP). We use this system as a starting point for our project and to explain our application scenarios for more advanced recommendation services. Subsequently, the objective of this study is described.

## 1.1 Recommendations in Tourist Information Systems

The usage of advanced technology such as the internet has dramatically reduced the barriers to publishing and distributing information. Consequently, efficiently and effectively searching for valuable information through all the available information is crucial. In order to narrow down a number of choices of available products which may meet a user's needs, several e-commerce web sites have provided recommendation service to their users. The services suggest products and provide the users with information to facilitate their decision-making processes. These e-commerce web sites may employ different approaches to meet their users' needs. Amazon.com, for instance, immediately recommends best-seller books and new release books for certain categories, without asking for any user input. In contrast, www.activebuyers-guide.com asks the users to choose the answers from a list of questions about the products that best describe their needs, e.g., photo quality, zoom in capability, size and price for a digital camera. The returned results will be ranked by which products best meet the user's answers [29].

The number of tourism portals that provide information filtered by users' specific requests and preferences has been increasing [28]. Thus, searching for relevant information in the tourism area, a user also faces the danger of information overload. The idea of providing recommendation services in tourist information systems comes from the same principle as recommendation services employed in e-commerce web sites. A recommendation service may start by asking a tourist to define some trip characteristics and personal interests, such as the party traveling, the available budget and the transportation means. Then they search through the available travel information to find the most appropriate recommendations [25]. Giving the user a list of famous sights

around the area is another possibility for recommendations.

Nevertheless, providing recommendations in the tourism environment is more complicated than preparing a list of products a user may want to buy in general e-commerce web sites. Zipf [38] claims that a successful tourist application requires good integration of personalization and context-awareness. He argued that a system offering the most relevant information in a given situation , determined by place, time, user interests and so forth, will be more successful than a system offering only default information.

In general, tourists go traveling under various circumstances, e.g., traveling for business or pleasure, traveling alone or in group and so on. Normally, their requirements are different according to their travel contexts, for example, less budget restriction in a business trip or no adventure activities when traveling with small kids, etc. Therefore, a user may have different requirements depending on their travel contexts that they define to the system in each travel session. Another factor is that tourists frequently change their location during their travel. A consequence of this mobility is that the contextual conditions around them always change across time. Furthermore, information in which these tourists are interested is location-dependent by nature. So the tourist's current position then acts as another parameter for the recommendation system. Besides these user requirements, the systems also need to consider other contextual information about the sights, for instance, weather conditions, opening and closing time of the places, available transportation to the place and so forth [4].

In conclusion, giving recommendations in mobile tourist information systems requires a good combination of personalization and context-awareness. To choose relevant information for a user, the system needs to be able to learn and build a user's information requirement models based on the user's current position and context, (e.g., on private vacation with family) as well as the context of the sights that best match this user's needs.

## 1.2 Recommendation Component in TIP 2.0

The Tourist Information Provider TIP is an advanced mobile tourist information system (see [17, 18, 21, 26]). The system is a combination of an event notification service (ENS) and a location-based service (LBS). The main focus of the system is to deliver users information about sights based on their location, interest, travel route and sight-related information. The users receive information via mobile devices such as mobile phones or Personal Digital Assistants (PDA). Sights in TIP are gathered into several semantic groups, such as cathedrals, sculptures, buildings and so on. The first prototype of TIP (TIP 1.0) recommends other sights in the group if the user has already visited at least two other sights [21]. For example, the user will get information about other cathedrals if they have visited two of other cathedrals in the past. This simple recommendation is given based on the assumption that the user likes the sights in the group, if they visited at least two of them.

In TIP 2.0, the recommendation service has been extended by the author to provide further recommendation methods [15, 19]:

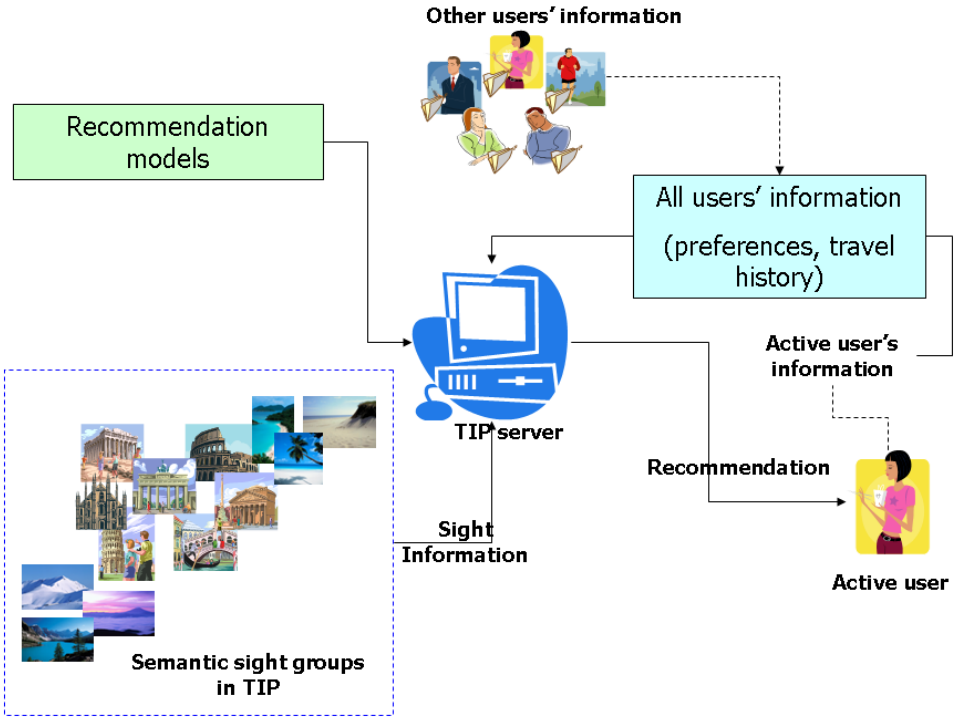



Figure 1.1: Overview of recommendation component in TIP 2.0

1. The users can ask for sights *close to* or *near by* their current location.

2. The users can request the system to recommend sights similar to sights that they liked in the past.

3. The user can request the system to recommend sights enjoyed by other users, who have similar taste to this user.

Figure 1.1 gives an overview of the recommendation component in TIP 2.0, which has been developed by the author in a previous project. The recommendation approaches in TIP 2.0 are more complex than the one implemented in TIP 1.0. Information about the users, for example, their likes, dislikes, feedback scores and travel history, have been employed as an essential input for providing recommendations to the users. The results of the study [15, 19] have confirmed a high potential for employing recommendation services in mobile tourist information system. Analyzing the results of this study, we identify limitations of the implemented recommendation models as follows:

1. **New user problem**

   TIP 2.0 generates a users' preferences based on their interests in semantic groups of sights, for instance, churches, cultural-buildings, offered by the system as well as their feedback scores given to sights in their past visits. When a new user enters the system however, the system knows nothing

about their preferences or impressions of the sights. The lack of initial information necessary for building an appropriate user model prevents the system from providing recommendations. This problem initiates a cold start problem, as explained in the following paragraph.

2. **Sparse feedback scores or cold start problem**

   Identifying a group of users who have similar taste to a given user depends on a number of feedback scores given by these users. Thus, if the number of users is small relative to the number of sights in the system, especially at the beginning of the system usage, then the coverage of feedback scores becomes very sparse. This problem leads to an inability to find a group of similar users and a failure to give recommendations. Cold-starts are a significant problem in giving recommendations. If an initially given recommendation does not make sense to the users, the users' confidence in the system may drop; the users may stop using the system.

3. **Users with specific preferences**

   This also known as the *Gray Sheep* problem [37]. It refers to a user whose preferences are unusual compared to other users in the system. Typically, opinions given to the items by this user do not consistently agree to any groups of the users in the system. These individuals will rarely receive accurate recommendations.

4. **Over-specialization**

   The user is restricted to see items similar to those already rated since the system can only recommend items scoring highly against their preferences.

5. **Transparency and user control**

   Understanding the relationship between the user's input to the system (ratings made by user) and system's output (recommendations) allows the user to initiate an efficient interaction with the system. Transparency allows users to meaningfully revise their input in order to improve recommendations, rather than being blinded by the given result [35]. Feeling that they are in control, users may visit a recommended sight whereas they might be unwilling to commit to a vacation spot without understanding the reasoning behind such a recommendation. In TIP 2.0, the users are in control that they have provided their interests as well as their feedback scores to the system [19]. The users can also change their interests whenever they want. However, no study has been conducted so far that evaluates the system's transparency and user control.

6. **User satisfaction**

   The user's satisfaction with the recommendations given by the system reflects the quality of the implemented user models, the recommendation algorithms and the quality of the available data. However, a difficulty in evaluating if a recommendation approach satisfies the users more than another lies in the evaluation criteria [11]. One simple way to determine

if the user is pleased with the recommendations is to record the user interaction to the system. If the user follows a given recommendation we may assume they are satisfied with the recommendation. Another way is to ask the user to directly rate the recommendations according to their approval. User satisfaction has not been measured in our previous study.

7. **Scalability**

   Providing recommendations requires computations that are very expensive and grow non-linearly with the number of users and items in the database. Therefore, to successfully employ recommendations on the web, sophisticated data structures and advanced, scalable architectures are required to provide recommendations with acceptable delay time responding to the users' requests [27]. Similar to the webs, giving recommendations in a mobile tourist information system tends to encounter scalability problem when the system is increasingly utilized by the users. The preliminary results of implementing the three recommendation models in TIP 2.0 does not allow for reasoning about scalability of the implemented recommendation approaches. Thus, it is vital to identify the recommendation models which show good scalability.

In summary, TIP 2.0 has shown a high potential for providing effective recommendation models in a mobile tourist information application. Nevertheless, several existing limitations have become apparent and need to be addressed.

## 1.3 Application Scenarios

In the previous paragraph, we identified some limitations of the current recommendation component in TIP which may overshadow the advantage of the service. Accordingly, further examinations of the implementation need to be done. The following two application scenarios present the functionalities of new recommendation models that will help solve the existing drawbacks. These two scenarios are used as reference scenarios throughout this thesis.

### 1.3.1 Scenario for New User in TIP

Joey arrives at Nelson, a city located on the north-west coast of the South Island of New Zealand. He stops at the tourist information center to ask how to get to the YHA where he has booked a room. The reception at the information center gives him directions and additionally suggests that he may register with the TIP system so that he can access additional information via his mobile phone from anywhere, anytime he wants. Joey has never used the mobile tourist information systems before but he decides to try it. He first registers with the system by setting up his login id and password. Then he

defines his interests by selecting from a list of tourist attraction given by the system. Joey selects 'beaches' and 'outdoor activities' from the list. From his selection, the system then identifies other users who have similar interests to Joey, based on the information about his interests he has given. The next morning, Joey logs into the system asking for recommendations on places to visit. The following list of recommendation methods is provided:

- *Get recommended sights that match Joey's interests.*
- *Get recommended sights that are near to his current location.*
- *Get recommended sights which other users liked who have similar interests as Joey.*
- *Get a list of 'must see' sights which contain famous tourist spots in Nelson.*

As this is his first time on the system, Joey does not yet trust the recommendations he gets. He decides to play around for a while to see if the given recommendations match his tastes. Joey selects each recommendation method one by one to see and compare the result. He also tries changing his interests to see how the recommended items change. As a result, Joey feels more confident, so he decides to follow sights recommended by other liked-minded users and he chooses to go sea kayaking at the Abel Tasman national park.

### 1.3.2 Scenario for Experienced User in TIP

Daniel arrived at Christchurch, New Zealand just this afternoon after spending a week in Nelson. Figure 1.2 shows the tourist attractions around Christchurch city center area. He was introduced to the TIP system when he visited the Te Papa museum in Wellington two weeks ago. He has defined himself to the system as interested in 'arts', 'museums', 'parks' as well as 'outdoor activities'. After using the system for one week, he trusts that contributing feedback scores to the system helps improve his preferences stored in the system and, thus, improves the quality of the recommendations he will receive. He has given his feedback scores of every place he has visited.

At the same time, Anne arrives at Wellington, the capital of New Zealand. She plans to stay overnight before heading to Christchurch. This is not her first trip to New Zealand. She was in Christchurch last year for two days and was highly impressed by the city so she has decided to come back again. In her last visit, Anne used the TIP system. She was impressed by Hagley Park and the Botanic Gardens so she gave high scores to both places as her feedback to the system.

Next morning, Daniel starts his journey early at the Cathedral Square, indicated by a big circle on the map in Figure 1.2. He would like to visit other places around the area and asks the TIP system to recommend places. He also would like to see other users' opinions of these places. The system gives details of Hagley Park, which match his preference for 'parks' and has high feedback scores from Anne and other similar users. After his visit, he is also overwhelmed by Hagley Park so he gives his feedback to the system providing

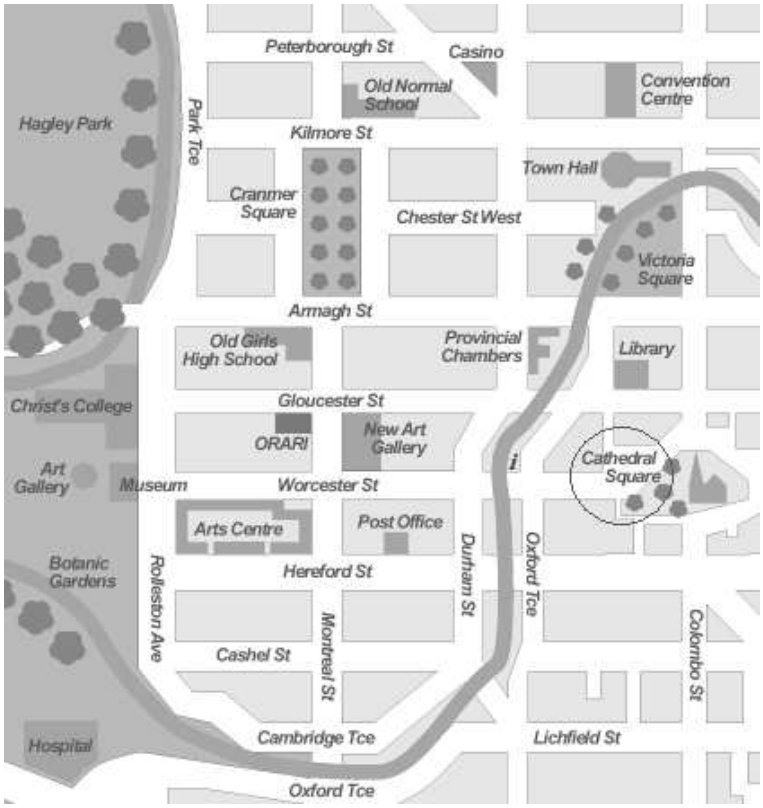



Figure 1.2: Christchurch city center map

a high score. Based on his preference, the information given by Anne two days ago and information from other users who have been here before and have similar preferences to Daniel, the system suggests to him to visit the Botanic Gardens.

After visiting the Beehive and Parliament House, Anne decides to connect to the TIP system again. She has defined that she is interested in 'beaches', 'park' and 'outdoor activities'. However, she knows that in this city she can hardly find any places matching her interests. So she decides to ask for *recommended sights liked by other users who have similar travel history.* Anne did not define her interest in museums but Daniel and other users, who were impressed by Hagley Park and the Botanic Gardens, liked the Te Papa museum. Therefore, TIP suggests Anne visit the museum before she departs to the South Island.

In these two scenarios, limitations in TIP 2.0 are overcome by the way in which the new recommendation component is presented. Missing information in a new user's feedback score has been complimented by their interests which they have defined regarding the sight groups. Similarity of travel history is used to form a group of users who have similar interests, in order to generate recommendations. Other recommendation approaches commonly provided in e-commerce web sites such as high scores sights or famous tourist spots in a particular area are also given.

## 1.4 Goal of this Study

The idea of giving recommendations to the users has been used in other areas, such as e-commerce web sites. However, as we have argued in the previous project [19] providing recommendations in a tourist information system is still not implemented. The simple recommendation method employed in TIP 1.0 [21] forms a first attempt to provide recommendations to a user based on their travel route. Three more complex approaches are implemented in TIP 2.0 in order to confirm the possibility of providing recommendation services in a mobile tourist information system. As a result, we have confirmed the high potential of providing recommendation services in TIP. However, the implemented recommendation approaches have led to some issues which need to be closely considered, as explained in Section 1.2. Consequently, this study extends our previous study in order to confirm the advantages of providing recommendation services in the tourist information system as well as to address some drawbacks we have learned from the previous study. Here, we focus on the following issues:

1. Extend the existing recommendation approaches and further the implementations by using a combination of user information (e.g., interests, feedback scores, travel history) and sight information (e.g., location and group) to remedy a shortage of information used in the recommendation process.

2. Evaluate the implemented recommendation approaches by identifying advantages and drawbacks among these recommendation models, both qualitatively and quantitatively. The recommendation models will be evaluated against the limitations which we analyzed in Section 1.2.

The goal of this study is to extend, implement, analyze and evaluate recommendation models we have generated in the previous study. We intend to find appropriate recommendation approaches which can compliment the existing advantages and balance the current drawbacks of providing recommendation in a mobile tourist information system. We believe that the result of this study will shield the users from the information overload as well as provide the users with satisfactory personalized recommendations in a mobile tourist information environment.

## 1.5 Structure of this Thesis

In Chapter 2, the background of the study is described. We start by explaining typical recommendation paradigms. Then parameters for generating recommendations are analyzed. Related work regarding both recommendation systems and tourist information systems are given.

In Chapter 3, we first list further recommendation models that we plan to implement in this study. Then we explain each approach in more details.

Chapter 4 gives details of the design of the algorithms used in the recommendation models which we introduce in Chapter 3. We first define terms and

definitions which will be used in the algorithms. We then transform the outline of the recommendation models into the implementation procedures and the calculation formulas.

Chapter 5 is concerned with the implementation of the recommendation component. Our focus is on the employment of the designed recommendation algorithms introduced in Chapter 4. We begin with the overview of TIP system implementation follow by implementation structure and implementation details of the recommendation component.

To validate the effectiveness of our implementation, we illustrate evaluation issues in Chapter 6. We design our evaluation framework based on the implications we gain from an investigation on the system evaluations conducted by the existing recommendation system research groups. In our evaluation plan, we propose a qualitative evaluation to confirm effectiveness of our designed recommendation algorithms. We also conduct quantitative evaluations to verify performance of these algorithms.

Chapter 7 illustrates qualitative evaluations. We first describe the setting of evaluations follow by the evaluation results. Then we give analysis of the results and summary.

Chapter 8 is concerned with quantitative evaluations. We begin this chapter with the experimental setting where we describe data set, methodologies, and experimental platform. Then we explain the results and analysis of the results.

Finally, there is a conclusion which summarizes this thesis and gives an outlook for future work in Chapter 9.

# Chapter 2

# Background

The fundamental goal of recommendation systems is to help users choose products that meet their needs in an effective way. However, building real world recommendation systems is extremely difficult. Developing efficient recommendation systems requires careful elicitation of user requirements, task analysis, development and tuning of the recommendation algorithms, plus design and testing of the graphical user interface [31]. In this chapter, we illustrate the general concepts of the three recommendation paradigms that have been used in many recommendation services, especially in e-commerce web systems. We then introduce details of the TIP system and classify the information regarding users and sights that are available in the system. The available information is defined as the parameters for generating the recommendation component which will be implemented as a service in TIP. We conclude with a brief overview of related work in both tourist information systems and recommendation systems.

## 2.1 Recommendation Paradigm

Recommendation or recommender systems are intelligent methods the users may apply when searching through huge volumes of information [30]. These systems help the user to make their decisions by providing them with recommendations, predictions, opinions, or user configured lists of items. The users then evaluate the given information according to their personal preferences. In general, there are two major factors in recommendation systems which need to be successfully put together in order to provide significant recommendations. One is the *items* and the other is the *users*. During a matching process, these systems compare the items' features, (e.g., music genre, book category) against information about the users, (e.g., their interest and/or their opinion about the items). The user's opinion is usually in the form of ratings or feedback scores. Some systems store the user information as their personal profiles in order to lessen the user's effort in providing their preferences. The existing recommendation systems have been developed based on the following three recommendation paradigms.

### 2.1.1 Content-based Recommendation

The content-based recommendation approach utilizes information about items and a particular user's requirements in order to provide them with recommendations. The system may explicitly ask the user to define his interests in one profile, e.g., a user of an on-line book store defines his interest in computer science, database system and programming languages. The other way of collecting user interests can be done implicitly by extracting user information from their past activities, e.g., their purchase history or a period of time they spent to consider a particular item before handing their purchase order etc. The system stores this information in the form of *user profiles* and refers to it during the information matching process in order to generate recommendations. The approach suggests items to the user based on the result of comparing the items' content to the users' preferences. So items that are similar to the items the user bought in their past purchase are recommended to the user. This has the advantage of being able to recommend items which match the users' preferences without waiting until the items are rated by other users. Moreover, users with a unique interest can get their information even though what they like is different from others. However, it is likely that the user is restricted to recommended items similar to what they liked before [8, 22, 24]. This approach has been implemented in recommendation systems which recommend, e.g., music [8], books [24] to the users.

### 2.1.2 Collaborative Filtering

The basic idea of the collaborative filtering paradigm is to provide item recommendations or predictions based on the opinions of other like-minded users. Rather than working out the similarity between items and user preferences, the collaborative filtering approach focuses on opinions the users have given about the items in their past activities. Opinions can be explicitly given by the user as a rating score, generally within a certain numerical scale, or can be implicitly derived from purchase records, by analyzing timing logs, by mining web hyper-links and so on [33]. As the system collects more ratings from more users, the possibility that someone in the system will be a good match for a given user increases. The computation of similarities between ratings provided by the given user and other users in the system forms a group of users who have close opinions to the given user or *neighbors* of the given user. Items that neighbors like are then recommended to the given user, as they will probably also like them. The advantage of this approach is that the given user is not restricted to recommendations about items in the same categories as they preferred in the past. However, it must be initialized with a large amount of data to generate effective recommendations because a system with a small base of ratings is unlikely to be very useful [6, 21]. The system can be useful for the users when a sufficient number of ratings on an item have been collected. This approach has been implemented and extended in [22]. Another prevailing example of the implementation is the Amazon.com web site. Amazon recommends the user buy some items other users have bought together with the items the user has decided to buy. Furthermore, the user also gets

opinions given by other users from their ratings and product reviews.

### 2.1.3 Knowledge-based Recommendation

The knowledge-based recommendation approach uses knowledge about users and products to pursue a knowledge-based approach to generate a recommendation, reasoning about what products meet the users requirements [7]. These recommender systems offer a dialog that effectively walks the user down a discrimination tree of product features. The *Adaptive Place Advisor* proposed by Goker et al. [10] is a conversational recommendation system designed to help users select a restaurant that meets their preferences. The system utilizes traces of the interaction with the user to adapt its similarity calculation, thereby personalizing the product retrieval and the conversation with the user. The user interacts with the system in order to narrow down the choices in an interactive fashion. This interaction takes the form of a sequence of questions, most designed to eliminate some items from consideration. According to their functionalities, the knowledge-based recommendation systems neither need explicit user feedback scores nor their preferences. However this advantage is hindered by a requirement for a knowledge engineering algorithm which mostly derives from machine learning techniques. The other drawback is the fact that only static suggestions can be gained from the system [6, 9]. Another system which has been implemented in this approach is the FindMe system [7].

In summary, each of the three recommendation paradigms mentioned above have advantages which would be practical to implement in some area while their drawbacks would impede their implementation in others. To provide effective recommendations in TIP, we need to balance these advantages and drawbacks.

## 2.2 Parameters for Generating Recommendation in TIP

Up to now, we have seen that all recommendation systems deal with two main parameters, the *items* and the *users*. Generally, the items are varied according to each provider, for example books, news, movies etc. The core idea of the three recommendation paradigms explained in the previous section is to narrow down choices of items given to the user and introduce the items to the user which might be of interest as well as discard other irrelevant items. To generate effective recommendations in the TIP system, we have considered taking all three recommendation paradigms into our experiments. Therefore, we are more specific about how we recognize these parameters in TIP. Subsequently, we have classified the items and the users into five subjects. The following section explains these five subjects in more detail.

**S1 User profile**

A user's profile specifies information of interest about the user. The system learns the user's preferences from the given information and provides recommendations based on the acquired knowledge about the user.

**S2 Context of a user**

A user's context may specify current location, time, weather, means of travel of a particular user etc. Current context determines for example, the suitability of a sight, distance from the users' current location and opening and /or closing time of the sight.

**S3 Context of a sight**

The sight context contains information about groups or types of sights for recommendation, which have certain features in common e.g., churches. The context of sights also covers their location, operating hours and weather conditions. Recommendations might be given on the assumption that users who have visited several sights in a group might be interested in seeing more sights of this group. A sight might not be recommended if it will close within half an hour.

**S4 User travel history**

The user's travel history which includes places, times and locations the user has visited. This information is a track of the users movements. Therefore the system will not recommend places the user has already visited. The system may learn user preferences from what users did in the past and predict what they would like to visit or do in the future.

**S5 User Feedback**

User feedback is ratings or scores given to the sights in their past visits. This requirement is identified in two forms.

**a)** *Feedback of this user*: The user receives recommendations based on the similarity of sights to other sights this user gave positive feedback about.

**b)** *Feedback of similar users*: Sights which other similar users liked may be recommended to the user.

As a result, the user gains wider information based not only on their preferences but also their similar preferences with other users.

Combining these five subjects and the three recommendation paradigms leads to the recommendation models. We believe that these proposed approaches are the best of compromise between the advantages and drawbacks

of the three paradigms. We therefore propose to confirm the idea by implementing and studying these recommendation models. Detail of the proposed recommendation models are explained in the Chapter 3.

## 2.3 Related Work

This section summarizes the examination in a previous study of the six tourist information systems including TIP 1.0 and five recommender systems. For more details refer to [19]. In the previous study [19], we analyzed tourist information systems for their information delivery and their recommendation function whereas the recommender systems (implemented in several areas such as book stores, music, web movies, restaurants and tourism) have been evaluated and compared to recommendation provided in TIP 1.0. In addition, both systems are analyzed and compared to the five subjects for recommendation generation (S1–S5) defined in the previous section. The tourist information systems concentrate mainly on the users' context, for instance their location and their preferences for information delivery. The accuracy of calculating the current location (S2) of the user and constraints of the size of the display are the main focus. Even though two of the six systems, TIP 1.0 [21] and GUIDE [9], have considered almost all of the five subjects for their information delivery, no advanced recommendation service is yet implemented. For the study of recommender systems, with the exception of the TIP system, the other five recommender systems in the study do not yet provide recommendations in a mobile environment. All of them are implemented as Web applications. Therefore, user context (S2) and sight context (S3) have not been utilized to deliver recommendations to the users. These systems also defined their user profile (S1) differently. CBCF [22] and LIBRA [24] consider the users' past rating as their profile while MRS [8] takes the user's access history into consideration. In other words, user history (S4) has a strong influence on the recommendations given in the three systems as well. In FindMe [7] the user's profile is built from their interactions with the system. Three of the systems, Fab , CBCF and LIBRA, require explicit feedback scores from the users while the other two, MRS and FindMe use information from the users' past access to the systems as their user feedback scores. Although the five subjects have been addressed in these recommendation systems, as yet none of the systems takes all of them to generate recommendation. Furthermore, these systems have encountered shortcomings in their implemented recommendation paradigms. Even though several attempts have been made to fix the problems, it seems that they could not solve one problem without creating another.

## 2.4 Summary

In this chapter, we first introduced three recommendation paradigms: content-based recommendation, collaborative filtering and knowledge-based recommendation. Then we identified two major parameters for the recommendation systems the *users* and the *items* into the following five subjects, *user*

*profile*, *context of a user*, *context of a sight*, *user travel history*, and *user feedback*, which we believe are crucial for recommendation service in TIP. From the last study (described in Section 2.3), we drew the conclusion that the existing research projects concerning recommender systems are still searching for some approaches to decrease the drawbacks as well as to promote the advantages they have found in their implementations. Meanwhile, the current mobile tourist information systems are still struggling with the accuracy of the user's location and the delivery of the information to small screen devices. Accordingly, none of the above recommendation approaches have yet been fully utilized in the environment of a mobile tourist information system.

# Chapter 3

# Advanced Recommendation Models for the TIP System

Recommendation models that are applicable to the TIP system have been defined based on the idea of providing personalized suggestions about sights to visit to a user. The recommendation models are generated by considering possibilities of combinations of the parameters for recommendation generating, recommendation paradigms and the existing recommendations component in TIP 1.0. Six of the proposed models have been implemented as one of the services provided in TIP 2.0. In this chapter, we describe general concepts of the proposed recommendation models in Section 3.1. Details of these approaches are explained in Section 3.2.

## 3.1 Proposed Recommendation Models

As our previous study [15] has shown that none of the other tourist information systems have fully implemented the recommendation functions in their systems. Hence, the current study is conducted in the TIP 2.0 system by generating the recommendation models, employing six basic recommendation models in the system and studying possibilities of putting the recommendation service into action. Results of this study show the positive effects of having recommendations as a service in TIP. To find out more consequences we decided to carry out a further study in TIP to achieve more appropriate recommendation models to be implemented. In this section, we describe the proposed recommendation models in five major approaches as shown in Table 3.1. We illustrate each of these approaches by giving title of the major approach, title of the recommendation model and a description.

Table 3.1: Proposed Recommendation Approaches

| Approaches | Models | Description |
|---|---|---|
| **A.Pure Approaches** | *A1.Content-based Recommendation* | Gives recommendation based on a particular user's profile (S1) and their feedback (S5a). Sights similar to what the user liked in the past are recommended. |
| | *A2.Collaborative Filtering Recommendation* | Recommends sights liked by other users who are similar to a particular user. This information is based on their previous feedback (S5a and S5b). Sights highly rated by these similar users are recommended. |
| | *A3.Knowledge-based Recommendation* | Recommends places based on sight context; it recommends sights that are semantically-related to sights this user has visited in the past (S2 and S4). For instance, a user gets recommendations about further beaches after they visited two beaches. |
| | *A4.Must-see Sights* | Recommends preset places that are the points of interest in a particular area, e.g., Sky Tower in Auckland. These points of interests can be defined based on the feedback of a large set of users (S5a and S5b). |
| | *A5.Nearby Sights* | Takes user context, sight context and user history into account (S2 and S3 and S4). The user context is user's current location, time and means of their travel. The user who travels by car will get more recommendations on farther point of interest or up coming activities than the user who travels by bike or on foot. Then the system suggests the user to go to the place they can conveniently visit and have never been to before. |
| | *A6.User Profile* | Gives recommendations on sights that match this user's profile (S1). |

Continued on next page

**Table 3.1 – continued from previous page**

| Approaches | Models | Description |
|---|---|---|
| **B.Compound Approaches** | *B1.Nearby Sights and User Profile* | Extends approach A5 by filtering the results of A5 according to a particular user's profile before giving a final recommendation. The user is required to explicitly define their preferences when they first register with the system. |
| | *B2.Revise Profile* | To recommend up-to-date things to this user, their profile (S1) may be revised according to their feedback given to the system (S5a). |
| | *B3.Extend Profile* | Gives recommendations on sights that match this user's extended profile. The user's profile is extended using information about other users. After establishing a group of similar users, information in their profiles is added to this user's profile. |
| **C.Extended Content-based Approaches** | *C1.Implicit Feedback* | Based on the principle of content-based recommendation but the user need not explicitly give their feedback to the system. Their feedback is created from the information in their user profile (S1) and the information on what they have done in the past which is recorded in their user history (S4). |
| | *C2.Content-boosted Recommendation* | A combination of the content-based recommendation and the collaborative filtering. If feedback given by this user is not yet enough, the data set for collaborative filtering is extended by simulating missing user feedback based on the feedback of other similar users. This approach is proposed in [22]. |

Continued on next page

**Table 3.1 – continued from previous page**

| Approaches | Models | Description |
|---|---|---|
| | *C3.Context-aware Feedback* | Uses content-based recommendation where the user gives their feedback according to circumstances of their context, e.g., the user prefer going to restaurant X when it is raining or the user likes going to cafe Y on a sunny day because it is near the beach. |
| | *C4.Implicit Context-aware Feedback* | Uses content-based recommendation based on this user's feedback (S5a) that are recorded according to sight context (S3) and user history (S4). The user need not explicitly give their feedback to the system but it is created from the information in their user history (S4) and the sight context (S3). |
| | *C5.User Information and Feedback* | Takes user profile (S1), user context (S2), sight context (S3), user history (S4) and their feedback (S5a) to verify recommendation to a particular user. However user context may or may not be considered. |
| **D.Extended Knowledge-based Approach** | *D1.Supplementary Sight Context* | Updates sight context according to the feedback of the user (S5a and S5b). Recommendations are given based on the information stored about the sights, e.g., the semantic groups they belong to. Feedback from user given about the sights may create new groups. |
| **E.Extended Collaborative Filtering Approaches** | *E1.User Profile* | Assumes that the users like items that match their user profile (S1). Therefore if no feedback (both S5a and S5b) is available from the number of users, the feedback is simulated by creating positive synthetic feedback data based on the user's profile. This synthetic feedback is then used as input for collaborative filtering. |
| | *E2.User History* | Similar to E1. Synthetic feedback is created based on the information in users' histories (S2). |

Continued on next page

**Table 3.1 – continued from previous page**

| Approaches | Models | Description |
|---|---|---|
| | *E3.User Profile and User History* | A combination of E1 and E2. Synthetic feedback is created for a group of similar users based on information from their user profiles (S1) as well as their user histories (S4). |

Both the pure and extended recommendation models described in Table 3.1 above are applied as the recommendation service in TIP. The next section explains these recommendation models in more detail.

## 3.2 Details of Recommendation Models

The five major recommendation models explained in Table 3.1 illustrate the concept of combining five parameters for recommendation generating and the three recommendation paradigms. These attempts aim to balance existing advantages and drawbacks that occurred in other application areas. To get a clear picture of the concept, we explain these proposed recommendation models in more detail. In this section, we reorder these models by placing the pure approach of the three recommendation paradigms together with their extended approaches. The following sections describe these recommendations in more detail.

### 3.2.1 Pure and Extended Collaborative Filtering Approaches

The collaborative approach recommends items that other users have liked. The approach pays attention to the similarity of the users rather than similarity of items. For each user a set of nearest neighbour users is found whose past ratings have the strongest correlation. Scores for unseen items are predicted based on a combination of the scores known from the nearest neighbors [5].

Figure 3.1 shows how the pure collaborative recommendation model functions in TIP. The nearest neighbours of user A are set up by calculating correlations between user A and other users. The higher the correlation the closer they are. A group of nearest neighbours of user A consists of other users who have the strongest correlation figures. Once the group of neighbours is found, predicted feedback scores for the sights that user A has not visited are calculated based on feedback scores given by these neighbours. Sights which hold high predicted feedback scores are recommended to user A in a descending order.

Since recommendations given by the pure collaborative model are made based solely on the similarity to other users, problems arise when a new sight appears in the database. This sight cannot be recommended to a user until

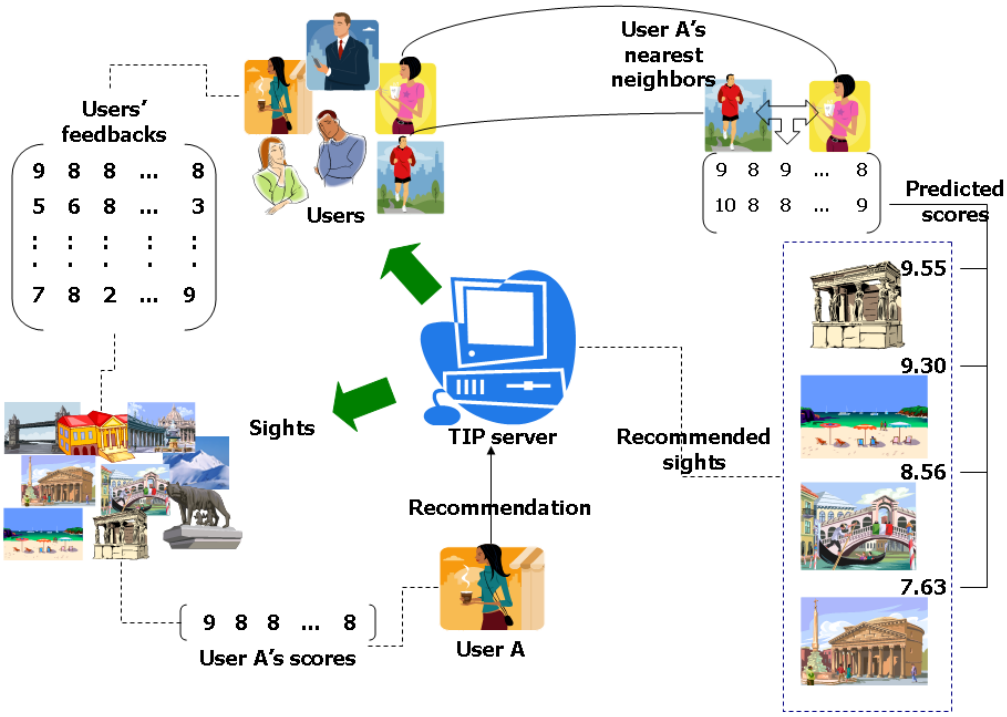



Figure 3.1: Collaborative recommendation models in TIP

enough feedback scores obtained from other users. Accordingly, when the growth of new sights is too rapid for feedback scores to be given, the feedback score coverage will be very sparse. This would weaken the system's ability to provide more recommended sights. Another problem occurs for the user who has unusual preferences compared to the rest of the population. There will not be any other users who are particularly similar. This would lead to poor or even no recommendations for such users.

To overcome these problems, user feedback scores used to find the user's nearest neighbours are not only taken from the ones that have been given explicitly but also from other sources of the user's information such as their profiles and their travel history. We assume the user is likely to give high feedback scores to sights that match their user profile. We also think that sights that the user has visited several times are likely to be their favorites. In this case, we give high feedback scores to sights the user has visited at least twice in their travel history. The three extended collaborative filtering models enlarge sparse user feedback scores from information in the user profiles, their travel history and a combination of their profile and travel history. Figure 3.2 describes a method of filling sparse user feedback scores by using user information in the user's profile or user's travel history or both of these information sources. A missing score is assigned to be a maximum score, in this case 10, if it matches the user profile or if the user has visited them at least twice. Otherwise a feedback score of 5 is given which means the user feels indifferent

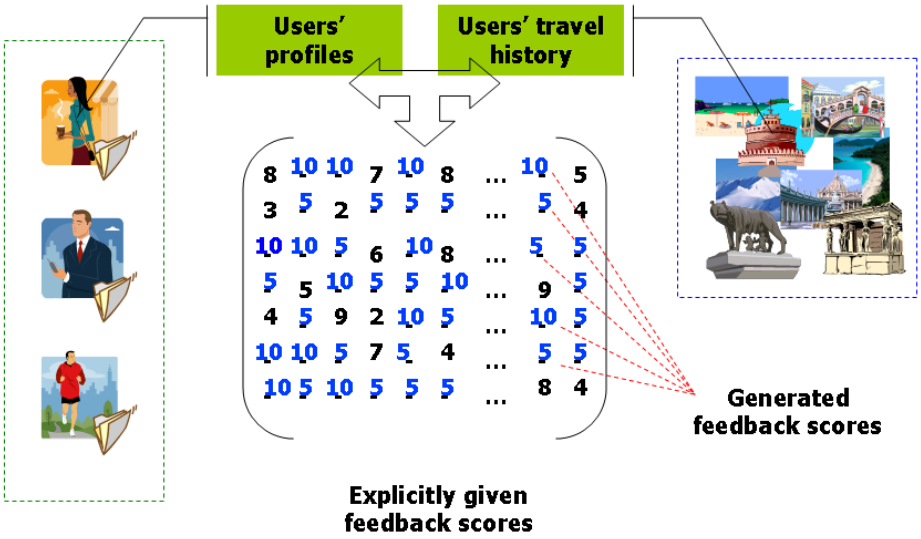



Figure 3.2: Extended sparse feedback score by information in users' profiles and/or their travel history

to the sight. These scores are used to find the correlation coefficient and form a group of nearest neighbours.

### 3.2.2 Pure and Extended Content-based Approaches

The pure content-based recommendation approach relies heavily on similarity between the items' content and the users' preferences. The idea is to identify particular items that are likely to match each user's taste or liking. In TIP, the content-based recommendation models are required to recommend sights that their content features match with those users' preferences. The two key factors of these recommendation models, item contents and user preferences, are pre-classified. TIP requires the users to define their interest the first time they register with the system. Meanwhile, sights in TIP are put together into semantic groups based on their communal content such as architecture, mountains and beaches.

The pure content-based recommendation model is generated based on the assumption that if the user liked any sight in the semantic groups they have defined they are interested in, they are likely to prefer more visits to the other sights in these groups. Not only the content of the sighs but also the user's preferences are taken into account in this model. The system checks if the user has visited any sights in the semantic sight groups they are interested in and

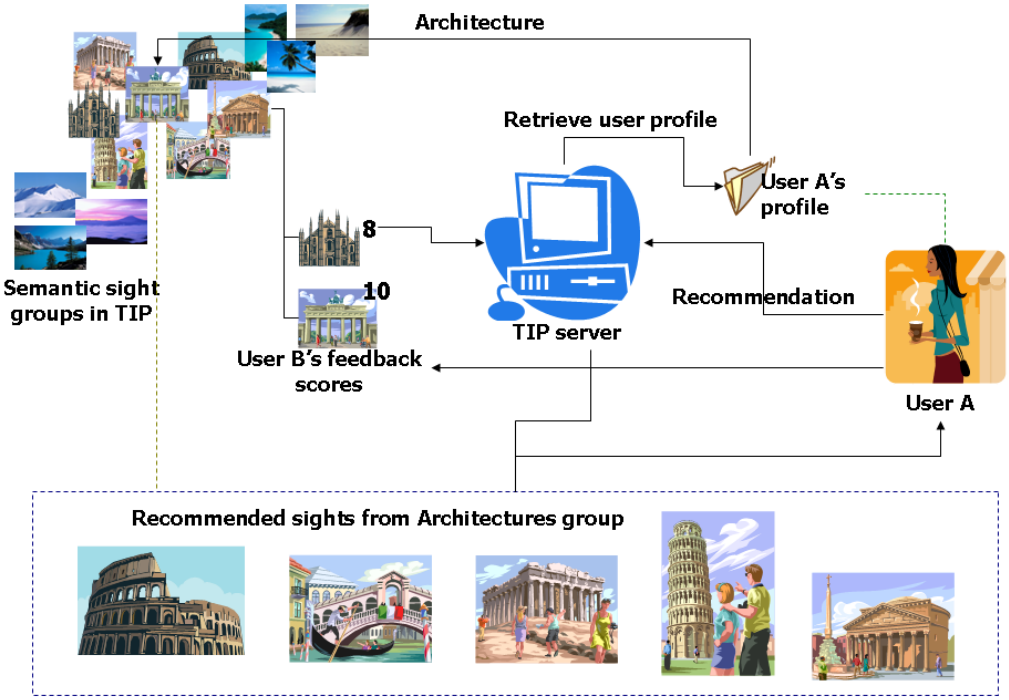



Figure 3.3: Content-based recommendation models in TIP

has given high feedback scores to those sights. Should any of these semantic group has the average feedback scores higher than a specific value which in this case is $\geq 7$, other sights in the groups are recommended. Figure 3.3 describes how a filter algorithm in the pure content-based recommendation model works.

Since the pure content-based model can only recommend sights scoring highly in a user's profile, the user is restricted to recommendations similar those already rated. In a mean time, another problem emerges when a new user starts to user the system. Since the user's feedback score is a crucial factor for this recommendation model, it is not possible for this user to get any recommendations at the beginning. The extended content-based models are trying to enlarge information about the users' preferences to increase the possibilities of finding matched items from the system. Despite the user feedback scores explicitly given by the users, we then transparently extract more user feedback by generating implicit scores from information in user profile and user travel history. The hybrid content-based, collaborative approaches [C1, C2, and C5] are created based on the idea presented in [5]. We retain user profiles and directly compare these profiles to establish similar users for collaborative recommendations. Users receive recommended sights both when they score highly against their own profile and when they are given high feedback scores by other users with similar profiles.

Figure 3.4 describes the extended content-based recommendation models functionalities. We determine similar users based on two sources of user infor-

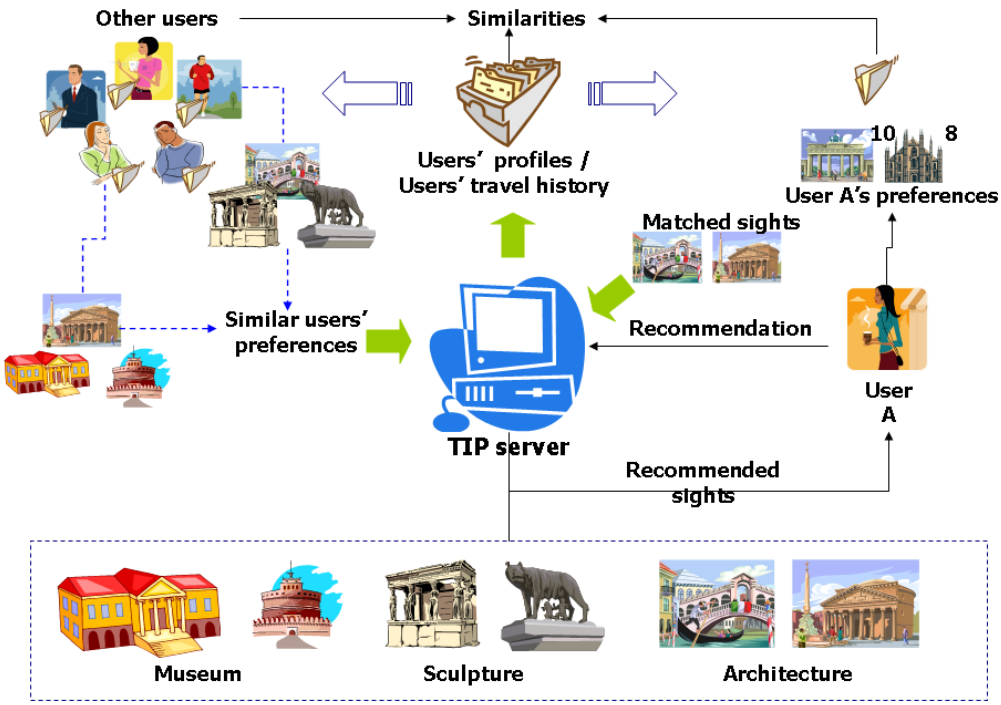



Figure 3.4: Extended Content-based recommendation models in TIP

mation. One is their profile and the other is their travel history. As a result, the user need not explicitly provide much feedback on the sights they have visited, but it is still not restricted to receiving recommendations only on sights similar to those highly rated in the past. The extended approaches avoid the limitation of sights suggested to the user as well as reduce user effort in giving feedback scores.

### 3.2.3 Pure and Extended Knowledge-based Approaches

The knowledge-based recommendation models pay attention to neither the sights' content nor the users' preferences. These models use the user's behaviour as a representation, commonly using machine-learning techniques to discover useful patterns in the behaviour [23]. TIP 1.0 employs a simple knowledge-based recommendation. TIP makes an assumption about the user behavior that if they have visited at least two sights in the same semantic group, they are likely to want to visit other sights in the group. The system then recommends other sights in the group to the user. Figure 3.5 explains the functionality of the knowledge based recommendation model implemented in TIP 1.0. Other sights in the architecture group are given to the users after the system finds the user has already visited two of the sights in the group.

The idea of extending the knowledge-based recommendation models attempts to interact with the user according to their preferences. The model

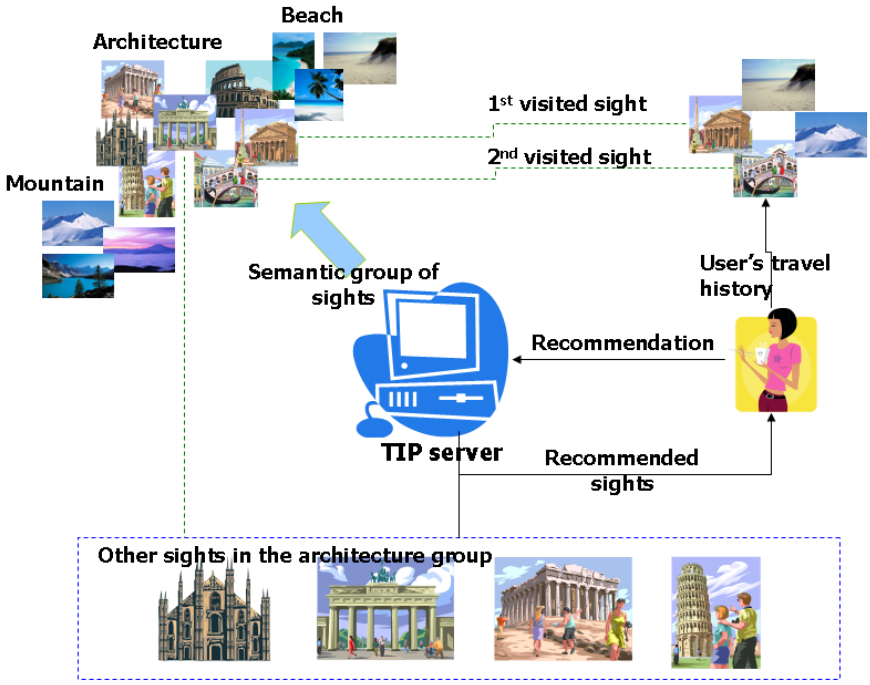



Figure 3.5: The pure knowledge-based approach in TIP 1.0

would exploit supplementary sight context with the existing semantic groups. The new semantic groups may be created based on the feedback score given by the user.

### 3.2.4 Compound Approaches

The compound recommendation model combines user information such as the user's current location and profile, to create filtering criteria. Nearby sights which matched a user's preferences are recommended to the user. We take information from other users as well as feedback scores into account in order to extend and/or revise the user's profile. Accordingly, the user would get up-to-date recommended sights in accordance with what they liked or disliked in their past visits.

In Figure 3.6 a list of nearby sights to visit is given to the user based on their preference and their current location. Although a sculpture sight is near by, the system discards it from the recommended list since the user is not interested in this semantic sight group. This figure also reveals an extension of the user's profiles by extracting information from other user's profiles to form a group of similar users. Similar users' preferences are exploited as the user's extended profile. User's feedback score is the other source of information used to revise the user profile. A semantic group that the user has given low feedback scores to will be deleted from the user profile. Meanwhile if some

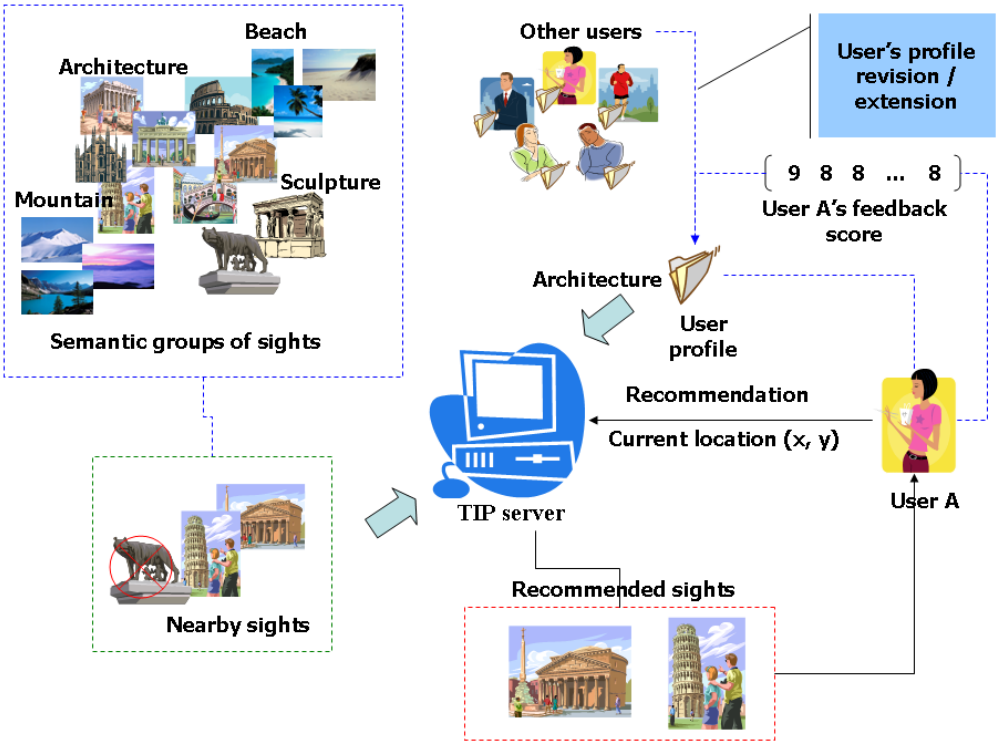



Figure 3.6: Compound recommendation models

sights in a semantic group have gained high feedback scores from the users, this semantic group will be added into the user profile.

### 3.2.5 Other Pure Approaches

We have explained the four major approaches of the recommendation models in the previous four subsections. The three pure recommendation models, the pure content-based recommendation, the pure collaborative filtering and the pure knowledge-based recommendation have already been addressed. All of the above approaches have taken at least two of the system parameters into account for giving recommendations. In this subsection we describe the other three pure approaches which take one user parameter into consideration to provide recommendation to the user.

Figure 3.7 shows the functions of these three pure approaches. The system determines the user's current location before giving the user a list of nearby sights to visit. Using the current location as a constraint for the user's information inquiry also gives a list of the points of interest in a particular area (the so called the must-see sights) to the user. For example, if the user is now in Auckland, New Zealand, the system will recommend the user to visit the Sky Tower. The last model takes the user profile into account. The user is recommended to see sights that match the preferences defined in their profile.

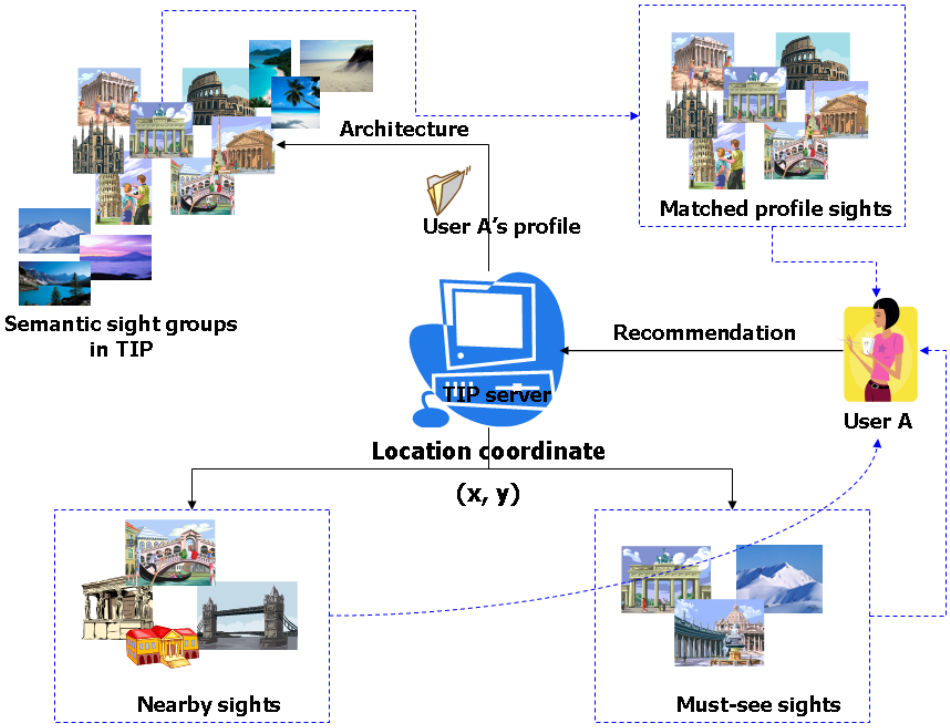



Figure 3.7: Other three pure recommendation models

## 3.3 Summary

In this chapter, we have classified into five major groups the applicable models for the recommendation service in TIP. These five groups include the pure approaches, the compound approaches and the extended approaches of each of the three recommendation paradigms. The idea behind the extended approaches is to use available user information such as their preferences verified in their user profile, their travel history and a combination of these two sources of information to replace the missing information required for generating the recommendations. We have verified the general concepts of these recommendation models as well as their descriptions in detail. In this study, we select to employ the pure and extended algorithms using the three recommendation paradigms. Design, implementation and evaluation of the selected models are conducted as a proof of concept and are explained in the next chapters.

# Chapter 4

# Design of Recommendation Algorithms

In the previous two chapters, we have learned the general principle of the three recommendation paradigms used in the existing recommendation systems. We analyzed five parameters required for generating recommendation models in TIP. The combination of the three recommendation paradigms and the five parameters have generated the outlines of the recommendation models to be implemented. In this chapter, we explain the design of the recommendation algorithms generated for these recommendation models. We start by defining terms used in the filter algorithms employed in these recommendation models in Section 4.1. We then introduce equations for and functionalities of these filter algorithms in Section 4.2.

## 4.1 Terms and Definitions

The following paragraphs explain terms and definitions used for the filter algorithms we have employed in this study.

**Definition 4.1.** ***User:*** *A user $u_x$ is a person who registers with the TIP system and interacts with the recommendation component in the system.*

Let $m$ be the number of users who have registered with the TIP system. The set of all users is $U = \{u_1, u_2, u_3, \ldots, u_x, \ldots, u_m\}$. We define an *active user* $u_a \in U$ as a particular user who is currently asking for recommendations.

**Definition 4.2.** ***Sight:*** *A sight $s_x$ is an item of interest a user might be interested in.*

Let $n$ be the number of sights stored in the TIP system. The set of all sights is $S = \{s_1, s_2, s_3 \ldots, s_x, \ldots, s_n\}$. A sight contains information or facts which can be classified into various topics, e.g., information about a particular building may consist of the architecture of the building, the person who designed or built the building, the year the building was built, etc. (see Definition 4.5).

Meanwhile, information about a specific beach may contain location, famous activities, e.g., surfing, diving or sea kayaking etc. A sight can be assigned feedback scores by the users who want to express their opinions (see Definition 4.8). The sight can be suggested to users who ask for a recommendation for a place or places to visit (see Definition 4.12).

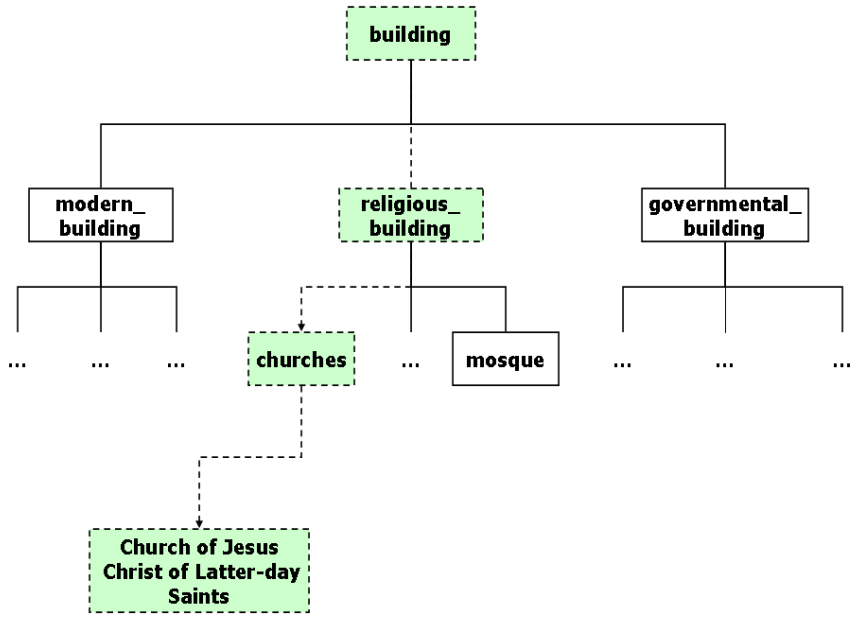



Figure 4.1: Hierarchical structure of the sight *The Church of Jesus Christ of Latter-Day Saints* verified in the sight group *Churches* and other two upper sight groups

**Definition 4.3.** ***Sight group:*** *A sight group $g_x$ is a set of sights which have some arbitrary features in common.*

A given sight can belong to more than one sight group if it is considered to have arbitrary features which are in common with more than one sight group. According to this concept, a given sight may be assigned to be under several defined sight groups or in a hierarchical sight group structure. For instance, the Temple of The Church of Jesus Christ of Latter-Day Saints located in Hamilton, New Zealand is considered to belong to the sight groups, *building* $\rightarrow$ *religious_building* $\rightarrow$ *churches* as indicated by dashed lines and boxes in Figure 4.1.

Let $k$ be the number of sight groups defined in the TIP system. The set of all sight groups is $G = \{g_1, g_2, g_3, \ldots, g_x, \ldots, g_k\}$. Each sight group $g_x$, where $x \in [1, k]$ consists of a set of sights.

**Definition 4.4.** ***Sight group profile:*** *A sight group profile $p$ is a set of sight groups a user $u_x$ has chosen as their sight groups of interests.*

Sight group profile is one type of a user preference information. Each user is required to choose one or more sight groups they are interested in to define their preferences to the system. The selected sight groups are referred to in a sight group profile. User interest in a particular sight group, $\alpha$, is

$$\alpha : U \times G \rightarrow \{0,1\} \text{ where } \alpha(u,g) = \begin{cases} 0 & : \text{ no interest of user } u \\ & \quad \text{in sight group } g \\ 1 & : \text{ interest of user } u \\ & \quad \text{in sight group } g \end{cases}$$

A sight group profile $p$ is defined as $p : U \rightarrow \{0,1\}^k$ and a sight group profile of a particular user $u_x$ is $p_{u_x} = \{\alpha(u_x, g_1), \alpha(u_x, g_2), \ldots, \alpha(u_x, g_k)\}$ where $g_i$ refers to a sight group defined in the TIP system.

**Definition 4.5.** ***Topic:*** *A topic $t_x$ is a type of information or facts about a sight stored in the system.*

Typically, TIP delivers general information about a particular sight together with its picture to the user. More detailed information about the sight depends on the sight group it belongs to. For example, there is information about architecture for churches whereas there is information about size or dimension (width x height) and artist for sculptures. The information is structured into levels which means that more interested users can get more detailed information and the less interested users only receive the high level information.

**Definition 4.6.** ***Topic profile:*** *A topic profile is a set of topics that a user is interested in.*

Each user is required not only to define their preferences to sight groups available in the system but also is expected to define their interests in topics such as *architecture, history, general etc.* for those sight groups they are interested in. This is to specify which detailed information about a sight will be shown to them. The selected topics are referred to in a topic profile.

**Definition 4.7.** ***Travel history:*** *A travel history $H$ is defined as a temporarily ordered list of sights visited by a user $u$ in the past.*

Whenever the user $u$ visits the sight $s_j$, this information is stored in their travel history. User travel history contains not only a sight the user has visited but also date and time which is referred to as a time stamp $t_j \in T$ (T being all timestamps) of their visit. A travel history of user $u$ is $H(u) = \{(s_j, t_j) | s_j \in S \wedge u \in U \wedge t_j \in T \wedge u \, visited \, s_j\}$.

**Definition 4.8.** ***Feedback:*** *A feedback $f$ is a numeric value which represents a user's opinion about a sight.*

A feedback is generally given in the form of a rating score and follow a specific numerical scale from 1 to 10 (e.g., 1-bad, 5-indifferent, 10-excellent): $f : U \times S \rightarrow [1, 10] \subseteq \mathbb{N}$ with

$$f(u,s) = \begin{cases} 1\ldots4 & : \text{ user } u \text{ has low impression of sight } s \\ 5 & : \text{ user } u \text{ has indifferent impression of sight } s \\ 6\ldots10 & : \text{ user } u \text{ has a good to excellent impression of sight } s \\ no-value & : \text{ user } u \text{ has not yet given an impression of sight } s \end{cases}$$

If a user changes their given opinion to a certain sight later on, the new opinion is kept and the existing one is discarded. We distinguish two types of feedback. One is called *given feedback* which can be either explicitly provided by a particular user to a given sight or inferred from available information of this user, e.g., their preferences stored in their sight group profiles and/or their travel histories. Hereafter, we refer to explicitly given feedback as $f_g$ and implicit given feedback as $f'_g$. The number of ($'$) in the implicit given feedback indicates the type of user information used, e.g., $f''_g$ uses information from the user sight groups profile, $f'''_g$ uses information from the user travel history and $f''''_g$ uses information from both information sources.

The other type of feedback is called *predicted feedback* ($f_p$). Predicted feedback is a numeric value calculated by the system and it is regarded as an opinion about a given sight the active user may give. Generally, the predicted feedback is calculated for a sight about which the active user has not yet given their opinion. The system generates predicted feedback by first choosing a subset of appropriate users based on a similarity of their feedback to the active user. Then a weighted aggregate of these users' feedback is used to generate the predicted feedback for the active user.

**Definition 4.9.** ***Prediction:*** *A prediction $\lambda$ is a numeric value which represents the degree of interest a user may have about a sight.*

Generally a prediction, $\lambda : U \times S \rightarrow [1,10] \subseteq \mathbb{N}$, is calculated for an individual sight or sights belonging to a particular sight group about which the active user has not yet defined their interest. The system generates a prediction by first selecting a subset of appropriate users based on a similarity of their interests in sight groups or sights to the active user. Then a weighted aggregate of these users' feedback is used to generate the prediction for the active user. Prediction on the degree of interest in a sight group $g$ of the user $u$, $\lambda(u,g)$, is applied to all of the sights belonging to this sight group.

**Definition 4.10.** ***Similarity Factor:*** *A similarity factor $\zeta$ identifies similarity of preferences between the active user $u_a$ and another user.*

A similarity factor is a numeric value which is calculated by the applied calculation algorithm and ranges from -1 to 1, $\zeta : U \times U \rightarrow [-1,1] \subseteq \mathbb{R}$. The similarity factor between the active user $u_a$ and user $u$, $\zeta(u_a,u)$, can be calculated based on historical information such as their feedback scores given to sights in their past visits, their travel histories or their interests in sight groups that have been defined in their sight group profiles.

**Definition 4.11.** ***Neighbour:*** *A neighbour $N$ of an active user $u_a$ is another user who has historically shown preferences similar to the active user.*

A criterion for which another user is a neighbor of the active user is considered from an interpretation of a calculated similarity factor between them. Generally, a specific numeric value is set to be a criterion for a valid similarity factor. Therefore, the neighbourhood $N$ of the active user $u_a$ is $N(u_a) = \{u|u \in U \wedge u \neq u_a \wedge \zeta(u_a, u) > criterion\}$ where $\zeta(u_a, u)$ describes the similarity factor between the active user $u_a$ and user $u$.

**Definition 4.12.** ***Recommendation:*** *A recommendation $R$ is a set of sights that the system predicts an active user $u_a$ would like to visit.*

Recommendations are given to the active user when the user asks for sights to visit. Recommendations given to the active user are varied depending on the recommendation model selected by the active user.

## 4.2 Design of Recommendation Algorithms

So far, we have defined terms and definitions to be used for the recommendation algorithms. In this section, we describe the design of the filter algorithms employed in the selected recommendation models in more detail.

### 4.2.1 Algorithms Using Collaborative Filtering Paradigm

In the following paragraphs, we describe the recommendation algorithms that are generated based on the knowledge of collaborative filtering recommendations. The idea behind these algorithms is that it may be of benefit to a user searching for information to consult the behavior of other users who share the same or related interests [27]. The recommendation models A2, E1, E2 and E3 described in Chapter 3 are developed based on this assumption.

In this project, a typical scenario for the collaborative filtering algorithms consists of a set of $m$ users $U = \{u_1, u_2, u_3, \ldots, u_a, \ldots, u_m\}$ and a set of $n$ sights $S = \{s_1, s_2, s_3 \ldots, s_x, \ldots, s_n\}$. Each user $u$ has a set of feedback scores, $f_g(u, s)$, which they have explicitly given to sights on their past visits. When an *active user*, $u_a$, asks for recommendations our task is to find a sight or sights that he or she may be interested in by using a collaborative filtering recommendation approach. To fulfil our task, we choose the most prevalent algorithms used in collaborative filtering which are called *neighbourhood-based methods* [12]. Neighbourhood-based methods can be separated into three steps.

1. Collect the feedback scores provided to sights by the users : the *feedback scores collection.*

2. Weight all these users with respect to similarity with the active user and select a subset of these users as a set of predictors for the active user : the *neighbourhood formation.*

3. Normalize feedback scores and compute a prediction from a weighted combination of selected neighbour's feedback scores for sights for which the active user has not yet given feedback scores : the *recommendation generation.*

The following sections explain these three steps in more detail.

#### 4.2.1.1 Feedback Scores Collection

In our collaborative filtering recommendation scenario, a problem space can be formulated as a matrix of users versus sights. In this case, we formulate a $m \times n$ *user-sight* matrix $F$.

$$F = \left\{ \begin{array}{ccccc} f_g(u_1, s_1) & f_g(u_1, s_2) & f_g(u_1, s_3) & \dots & f_g(u_1, s_n) \\ f_g(u_2, s_1) & f_g(u_2, s_2) & f_g(u_2, s_3) & \dots & f_g(u_2, s_n) \\ f_g(u_3, s_1) & f_g(u_3, s_2) & f_g(u_3, s_3) & \dots & f_g(u_3, s_n) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ f_g(u_a, s_1) & f_g(u_a, s_2) & f_g(u_a, s_3) & \dots & f_g(u_a, s_n) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ f_g(u_m, s_1) & f_g(u_m, s_2) & f_g(u_m, s_3) & \dots & f_g(u_m, s_n) \end{array} \right\}$$

Each entry $F(i, j)$ in matrix $F$ represents a feedback score $f_g(u_i, s_i)$ of user $u_i$ on sight $s_i$. Each individual feedback score is either a numerical value indicating the user's preference or a *no-feedback* value indicating the user has not yet evaluated that sight.

Generally, it is not possible for each user to give their feedback scores to all sights in the matrix even for a very enthusiastic one. As a result, this matrix is usually sparse because of numerous *no-feedback* values. This makes it harder for the filtering algorithms to generate satisfactory results. To reduce sparsity of the initial user-sight matrix, we simply insert implicit feedback scores to replace the *no-feedback* values. We propose these implicit feedback scores be generated using the following techniques:

- **Default Feedback**

  This simple technique is applied in the pure collaborative filtering approach (A2). In this case, we reduce the sparsity of the user-sight matrix $F$ by placing default feedback scores for those *no-feedback* values. In order to prevent both positive and negative influences, we use a neutral score, which in this case is the score = 5.

$$f'_g(u, s) = \left\{ \begin{array}{lcl} f_g(u, s) & : & \text{if user } u \text{ rated sight } s \\ 5 & : & otherwise \end{array} \right.$$

- **Using information in the user profile**

  This technique is applied to the extended collaborative filtering by the user profile approach (E1). We reduce the sparsity of matrix $F$ by using information which a user has defined in their user profile. We assume that the user is likely to give high feedback scores to sights they have defined as their likes. As a result, a score of 10 is allocated to a sight

which belongs to a sight group that is stored as a preference in the user's sight group profile, or a score of 5 (neutral score) otherwise.

$$f''_g(u,s) = \begin{cases} f_g(u,s) & : \text{ if user } u \text{ rated sight } s \\ 10 & : \text{ if sight } s \text{ belongs to sight groups user } u \\ & \quad \text{has defined in their sight group profile} \\ 5 & : \textit{otherwise} \end{cases}$$

- **Using information from the user travel history**

  This technique is applied to the extended collaborative filtering by the user travel history approach (E2). We reduce the sparsity of matrix $F$ by using information in the user travel history. A user is assumed to like a particular sight if he or she has visited this sight at least two times. As a result, each *no-feedback* value is allocated by a score of 10, or a score of 5 (neutral score) otherwise.

$$f'''_g(u,s) = \begin{cases} f_g(u,s) & : \text{ if user } u \text{ rated sight } s \\ 10 & : \text{ if user } u \text{ has visited sight } s \text{ at least twice} \\ 5 & : \textit{otherwise} \end{cases}$$

- **Using a combination of information from user travel history and user profile**

  This technique is applied to the extended collaborative filtering by the user profile and user travel history approach (E3). A combination of the two sources of user information is used in order to fill in *no-feedback* values. A neutral score is placed into a *no-feedback* value if both the user profile and the user travel history information cannot be applied.

$$f''''_g(u,s) = \begin{cases} f_g(u,s) & : \text{ if user } u \text{ rated sight } s \\ 10 & : \text{ if sight } s \text{ belongs to sight groups user } u \\ & \quad \text{has defined in their sight group profile} \\ & \quad \textit{or} \\ & : \text{ if user } u \text{ has visited sight } s \text{ at least twice} \\ 5 & : \textit{otherwise} \end{cases}$$

Feedback scores, which are collected in this step, will be used to find the similarity between the active user and other users. The next section explains the similarity calculation and a neighbourhood formation for the active user.

#### 4.2.1.2 Neighbourhood Formation

The neighbourhood formation step focuses on two processes. First, we weight all the users with respect to their similarity with the active user in the *similarity calculation* process. Then, we select a subset of these users as a set of neighbours for the active user in the *neighbourhood selection* process. The following paragraphs describe these two processes in more detail.

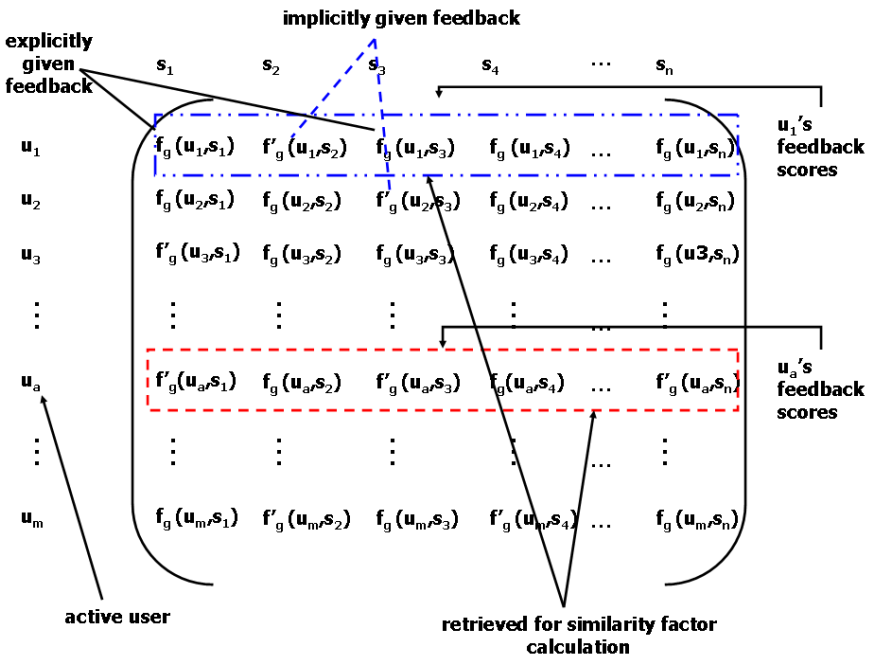



Figure 4.2: Feedback scores retrieval for similarity factor calculation

- **Similarity calculation**

  In the previous step, feedback scores from the users given to sights are collected in matrix $F$. Each row of the matrix represents the given feedback of each user. To identify which user $u_i$ has similar preferences to the active user $u_a$ with regard to their feedback, first the row representing user $u_i$'s feedback scores is retrieved. Then a *similarity factor* $\zeta(u_a, u_i)$ between these two users is calculated. Figure 4.2 illustrates a feedback score retrieval of user $u_1$ for a similarity factor calculation. The retrieved feedback consists of both implicit and explicit feedback. Papagelis et al. [27] reported in their study that several similarity calculation algorithms, for instance, cosine vector similarity, Pearson correlation, Spearman correlation etc., are available and have been widely used. However, the Pearson correlation performed the best for collaborative filtering algorithms. Therefore, we have decided to employ the Pearson correlation for the user-user similarity calculation, based on this result.

  The *Pearson correlation coefficient* is derived from a linear regression model that relies on a set of assumptions regarding the data [12]. The assumptions are:

  – the relationship must be linear;

  – the errors must be independent and have a probability with mean of zero;

  – the variance of every setting of the independent variable must be constant.

We revise the typical equation used to calculate a Pearson correlation in accordance with the definitions of the parameters we defined in the previous section. Hereafter, every equation we proposed for our calculation is revised using the same approach. We refer to the result of Equation 4.1 as a *similarity factor* between user $u_a$ and user $u_b$, $\zeta(u_a, u_b)$.

$$\zeta(u_a, u_b) = \frac{\sum_{i=1}^{n}(f_g(a,i) - \bar{\theta}_{u_a}) * (f_g(b,i) - \bar{\theta}_{u_b})}{\sqrt{\sum_{i=1}^{n}(f_g(a,i) - \bar{\theta}_{u_a})^2} * \sqrt{\sum_{i=1}^{n}(f_g(b,i) - \bar{\theta}_{u_b})^2}} \tag{4.1}$$

where

$f_g(a,i)$ denotes the feedback assigned to sight $s_i$ by user $u_a$,

$\bar{\theta}_{u_a}$ is the mean feedback given by user $u_a$,

$f_g(b,i)$ denotes the feedback assigned to sight $s_i$ by user $u_b$,

$\bar{\theta}_{u_b}$ is the mean feedback given by user $u_b$,

$n$ is the total number of sights.

All users are classified with respect to their similarity with the active user. The next measure is to select a subset of these users as the set of neighbours of the active user.

- **Neighbourhood selection**

  To select neighbours of the active user, we identify the similarity factors based on the interpretation of the Pearson correlation coefficient. Simon [34] described the meaning of the Pearson correlation coefficient and its interpretation as a measure of the strength of a linear relationship between two variables which, in our scenario, are the feedback scores of the two users. The Pearson correlation coefficient ranges from $-1$ to $+1$. Positive correlation indicates that both variables increase or decrease together, whereas negative correlation indicates that as one variable increases, the other decreases, and vice versa. He suggested the following interpretation of the correlation.

  - -1.0 to -0.7 strong negative association
  - -0.7 to -0.3 weak negative association
  - -0.3 to +0.3 little or no association
  - +0.3 to +0.7 weak positive association
  - +0.7 to +1.0 strong positive association

  Figure 4.3 shows a neighbourhood selection process. A subset of all other users is selected to construct a set of neighbours of the user $u_a$ based on

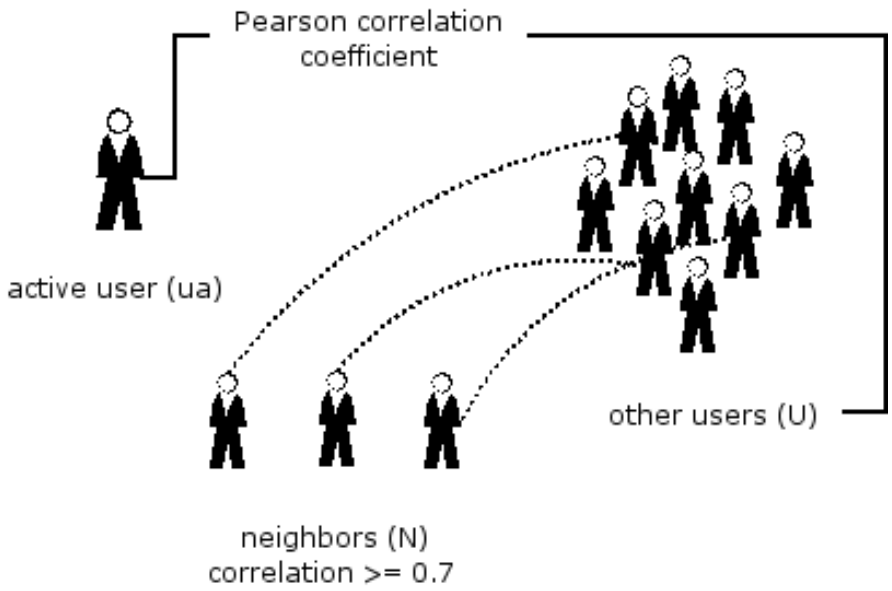



Figure 4.3: Neighbour of the active user

their similarity factor (a Pearson correlation coefficient). Typically, correlations above *0.70* mean the two variables have a strong positive association. Based on this interpretation, we set the criteria for a similarity factor to be greater than or equal to *0.7* for the selection of a neighbour of the active user. Therefore, the users who have Pearson correlation values from +0.7 to +1.0 are regarded as the neighbours of the active user $u_a$: $N(u_a) = \{u|u \in U \wedge u \neq u_a \wedge 0.7 \geq \zeta(u_a, u) \leq 1.0\}$

#### 4.2.1.3 Recommendation Generation

In the previous step, we chose a set of like-minded users or *neighbours* of the active user based on the similarity of their feedback scores. In this final step, we generate recommendations for the active user. To generate recommendations using the collaborative filtering paradigm we first compute a *predicted feedback* for sights which the active user has not yet given their feedback scores. The predicted feedback, $f_p$, is within the same scale (e.g., from 1 to 10) as the *given feedback*, $f_g$, addressed by the active user. We then consider sights that have predicted feedback values ranging from 7 to 10 as recommendations to the active user.

Figure 4.4 illustrates a subset of the *user-sight* matrix described in Figure 4.2. We collect feedback scores given to sights by the neighbours of the active user in a *neighbour-sight* matrix. Each *no-feedback* value for the active user $u_a$ will be replaced by a *predicted feedback* score. This is conducted by performing a weighted average of deviations from the neighbour's mean as

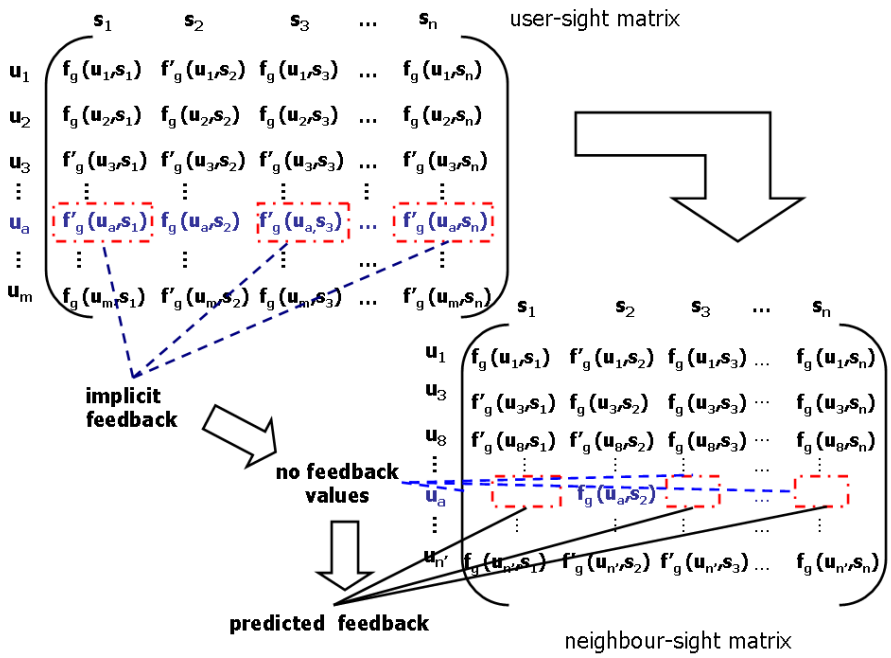



Figure 4.4: A *neighbour-sight* matrix shows empty feedback values of the active user.

shown in Equation 4.2.

$$f_p(u_a, s) = \bar{\theta}_{u_a} + \frac{\sum_{i=1}^{m'} (f_g(i, s) - \bar{\theta}_{u_i}) * \zeta(u_a, u_i)}{\sum_{i=1}^{m'} \zeta(u_a, u_i)} \tag{4.2}$$

where

$f_p(u_a, s)$ denotes the *predicted feedback* for the active user $u_a$ for sight $s$,

$\bar{\theta}_{u_a}$ is the mean feedback given by user $u_a$,

$\bar{\theta}_{u_i}$ is the mean feedback given by user $u_i$,

$\zeta(u_a, u_i)$ is the similarity factor between active user $u_a$ and neighbour $u_i$,

$m'$ is the number of neighbours.

The recommendations consist of sights which have high predicted feedback ($\geq 7.0$) and . These sights are arranged in descending order. The sight with the highest predicted score is on the top of the list followed by the second highest score and so forth. Finally, the recommendations $R_{u_a} = \{s_j | s_j \in S \wedge f_p(u_a, s_j) \wedge f_p \geq 7.0 \wedge j \in [1, n]\}$ are given to the active user.

In summary, the four recommendation algorithms using the collaborative filtering paradigm [A2, E1, E2, and E3] illustrate our attempt to address the

sparsity of the feedback scores matrix. We first generate *implicit feedback* values ($f'_g, \ldots, f''''_g$) by using default values or the available user information (e.g., sight group profile, travel history) and fill in the missing values. Then the similarity factor between the active user and other users is calculated. Other users who have strong positive association to the active user (similarity factors, $\zeta$, $\geq 0.7$) are selected as the active user's neighbours. A weighted combination of these neighbour's given feedback scores for sights is used to calculate the *predicted feedback* for sights the active user has not yet expressed their feedback scores to. Finally, sights with high predicted feedback scores ($\geq 7$) are recommended to the active user.

### 4.2.2 Algorithms Using Content-based Paradigm

The following paragraphs describe the recommendation algorithms that are generated based on the knowledge of the content-based recommendation algorithm. In content-based recommendation systems, the user first expresses some preferences for a set of products. Then the system retrieves from a catalogue the items that share common features with the products the user has judged as their interests [32]. The recommendation models A1, C1, C2, C3, C4, and C5 described in Chapter 3 are developed based on this assumption. We consider the *user preferences* not only from the information about sight groups the user defines that they are interested in, but also their feedback scores they have given to the sights in these groups. We assume that the higher the average value of the feedback score of each sight group, the more the user is interested in the sights in this sight group.

In this project, a typical scenario for the content-based algorithms consists of a set of $m$ users $U = \{u_1, u_2, u_3, \ldots, u_x, \ldots, u_m\}$ and a set of $k$ sight groups $G = \{g_1, g_2, g_3, \ldots, g_x, \ldots, g_k\}$. Each sight group $g_x$ contains a number of sights $s_i$, where $i = 1, 2, 3, \ldots, x, \ldots, n$ for which they are classified. Each user $u_i$ is required to define a set of sight groups they are interested in when they first register with the system. These sight groups are referred to as the sight group profile $p(u_i) = \{\alpha(u_i, g_1), \alpha(u_i, g_2), \ldots, \alpha(u_i, g_k)\}$. The user $u_i$ also has a set of feedback scores, $f_g(i, s)$, which they have explicitly given to sights in their past visits.

To generate the pure content-based recommendations [A1] for the active user $u_a$, first the sights the active user has visited are collected based on the sight group they are in. Then the average value of the feedback scores given to sights in each of these sight group, $\bar{\gamma}(u_a, g)$, is calculated using Equation 4.3.

$$\bar{\gamma}(u_a, g) = \frac{1}{n} \sum_{j=1}^{n} f_g(u_a, s_j) \tag{4.3}$$

where

$s_j$ is a sight belonging to the sight group $g$, $s_j \in g$,

$n$ is the total number of sights in the sight group $g$ that the active user $u_a$ has visited and given their feedback scores to.

The recommendations consist of other sights in sight groups that have high

average values (in this case $\geq 7$), $R_{u_a} = \{s_j | s_j \in g \wedge \bar{\gamma}(u_a, g) \geq 7 \wedge j \in [1, n]\}$, are then given to the active user.

As pointed out in Chapter 3, Section 3.2.2, the pure content-based recommendation algorithm does not provide any *surprise* recommendations to the user because the system can only recommend sights highly rated in the user profile. However, the algorithm causes several problems. The first problem is *over-specialization* [5]. In other words, the recommendation algorithm is too restricted so that the user will miss a chance to visit other sights in the group especially if they start their visits with bad impressions. The second problem is that since the users are highly encouraged to give their opinions of the sights they have visited, a new user will not get any recommendations until they give some feedback scores to the sights they themselves have visited. To address this shortcoming, we extend the simple contented-based recommendation algorithm by applying knowledge of the *neighbourhood-based methods* from the collaborative filtering paradigm. The idea behind these extended algorithms is to provide recommendations based on the preferences of other users who have similar *user information* to the active user. We do not concentrate on the similarities between the feedback scores given to sights by the users, rather we focus on the similarities between other types of user information available (e.g., user profile and user travel history). We adapt the three steps of the neighbourhood-based collaborative filtering algorithm into the following measures:

1. Collect information about the user (e.g., user profile, and user travel history) : the *user information collection.*

2. Weight all these users with respect to similarity with the active user and select a subset of these users as a set of predictors for the active user : the *neighbourhood formation.*

3. Create recommendations based on the neighbours' preferences : the *recommendation generation.*

The following sections explain more detail of these three measures.

#### 4.2.2.1 User Information Collection

We do not collect the feedback scores given to the sights by the users as we did in the collaborative filtering algorithms. Rather, we form a *user-user information* matrix $V$ and assign a score to each of the entries. Considering the available user information (e.g., sight groups profile and travel histories), the matrix $V$ can be either a $m \times k$ *user-sight group* matrix or a $m \times n$ *user-sight* matrix. Each entry $V(i, j)$ in matrix $V$ represents a score which is given according to validation of the information $j$ of the user $i$. We assign a score to each of the entries using any of the following techniques:

- **User-sight group profile matrix**

  A *user-sight group profile* matrix is a $m \times k$ *user-sight group* matrix $V$. Each entry $V(i, j)$ in matrix $V$ represents an interest $v'(u_i, g_j)$ of user $u_i$ in sight group $g_j$.

$$V = \left\{ \begin{array}{ccccc} v'(u_1, g_1) & v'(u_1, g_2) & v'(u_1, g_3) & \dots & v'(u_1, g_k) \\ v'(u_2, g_1) & v'(u_2, g_2) & v'(u_2, g_3) & \dots & v'(u_2, g_k) \\ v'(u_3, g_1) & v'(u_3, g_2) & v'(u_3, g_3) & \dots & v'(u_3, g_k) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v'(u_a, g_1) & v'(u_a, g_2) & v'(u_a, g_3) & \dots & v'(u_a, g_k) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v'(u_m, g_1) & v'(u_m, g_2) & v'(u_m, g_3) & \dots & v'(u_m, g_k) \end{array} \right\}$$

We assign a maximum score of 10 to the sight group the user has defined as their interest, otherwise a neutral score of 5 is assigned.

$$v'(u, g) = \left\{ \begin{array}{lcl} 10 & : & \text{if the user } u \text{ defined their interest in} \\ & & \text{the sight group } g \\ 5 & : & \textit{otherwise} \end{array} \right.$$

- **User-user travel history matrix**

  A *user-user travel history* matrix is a $m \times n$ *user-sight* matrix $V$. Each entry $V(i, j)$ in matrix $V$ represents the interest $v''(u_i, s_j)$ of the user $u_i$ in the sight $s_j$ based on a number of their visits.

$$V = \left\{ \begin{array}{ccccc} v''(u_1, s_1) & v''(u_1, s_2) & v''(u_1, s_3) & \dots & v''(u_1, s_n) \\ v''(u_2, s_1) & v''(u_2, s_2) & v''(u_2, s_3) & \dots & v''(u_2, s_n) \\ v''(u_3, s_1) & v''(u_3, s_2) & v''(u_3, s_3) & \dots & v''(u_3, s_n) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v''(u_a, s_1) & v''(u_a, s_2) & v''(u_a, s_3) & \dots & v''(u_a, s_n) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ v''(u_m, s_1) & v''(u_m, s_2) & v''(u_m, s_3) & \dots & v''(u_m, s_n) \end{array} \right\}$$

  A maximum score of 10 is assigned to the sight the user has visited at least twice in the past and a neutral score of 5 otherwise.

$$v''(u, s) = \left\{ \begin{array}{lcl} 10 & : & \text{if the user } i \text{ has visited the sight } j \\ & & \text{at least twice} \\ 5 & : & \textit{otherwise} \end{array} \right.$$

- **User-sight group profile and travel history matrix**

  A *user-user travel history and sight group profile* matrix is also a $m \times n$ *user-sight* matrix $V$. Each entry $V(i, j)$ in matrix $V$ represents the interest $v'''(u_i, s_j)$ of the user $u_i$ in the sight $s_j$ based on two conditions. The sight is in the sight groups the user is interested in or the user has visited the sights at least twice in the past. A maximum score of 10 is assigned to a sight if either of the two conditions can be applied and a neutral score of 5 otherwise.

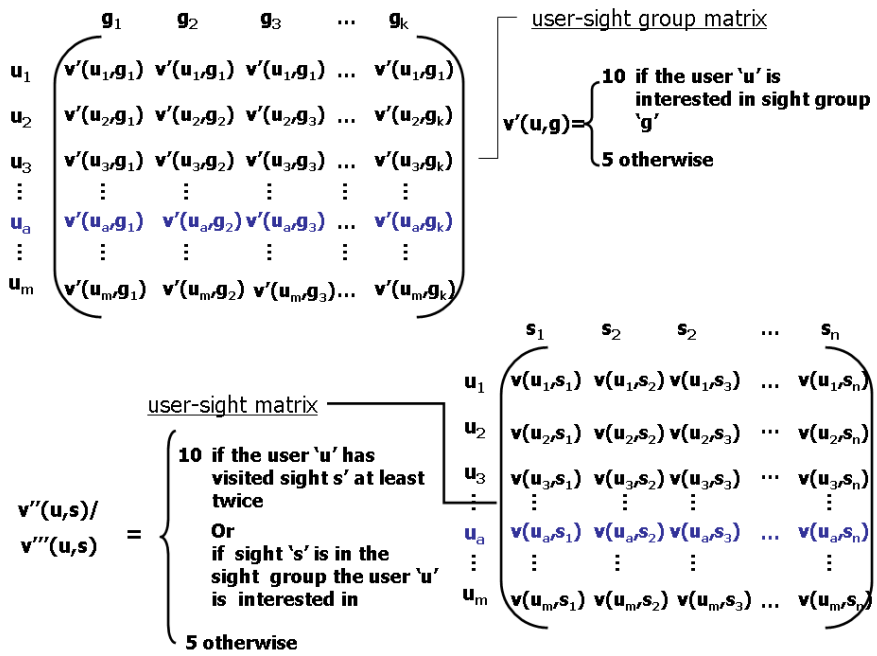



Figure 4.5: A *user-user* information matrix generated in the user information collection measure.

$$v'''(u,s) = \begin{cases} 10 & : \text{ if sight } s \text{ belongs to sight groups user } u \\ & \quad \text{has defined in their sight group profile} \\ & \quad \textit{or} \\ 10 & : \text{ if the user } i \text{ has visited the sight } j \\ & \quad \text{at least twice} \\ 5 & : \textit{otherwise} \end{cases}$$

Figure 4.5 explains the three techniques employed in the user information collection. The upper matrix illustrates a *user-sight group* matrix whereas the lower matrix illustrates a *user-sight* matrix using information about the users' travel histories or a combination of the two sources of user information. These techniques are implemented in the extended content-based recommendation approaches using implicit feedback [C1] and using user information and feedback [C5].

#### 4.2.2.2 Neighbourhood Formation

We apply the same principle as the neighbourhood formation method used in the collaborative filtering algorithms. We classify our neighbourhood formation into two measures depending on the user information gathered in the user-user information collection step. The following paragraphs explain these two measures in more detail.

- **Similarity on sight group of interests**

  Here, we are not interested in the similarity of feedback scores given to sights by the users. Rather, we concentrate on the similarity of their

interests in the sight groups. We modify our original equation used to calculate the similarity factor (see Equation 4.1). We propose to use the score assigned to the sight groups according to the users' interests defined in their their sight group profiles. In this case we define the similarity factor between the active user $u_a$ and user $u$ as the *Similarity factor on the Sight Group of Interest*, $\zeta_g(u_a, u)$, as shown in Equation 4.4.

$$\zeta_g(u_a, u_b) = \frac{\sum_{i=1}^{k}(v'(a,i) - \bar{\theta}_{u_a}) * (v'(b,i) - \bar{\theta}_{u_b})}{\sqrt{\sum_{i=1}^{k}(v'(a,i) - \bar{\theta}_{u_a})^2} * \sqrt{\sum_{i=1}^{k}(v'(b,i) - \bar{\theta}_{u_b})^2}} \tag{4.4}$$

where

$v'(a,i)$ denotes the score assigned to sight group $g_i$ for user $u_a$,

$\bar{\theta}_{u_a}$ is the mean score assigned to user $u_a$,

$v'(b,i)$ denotes the assigned score to sight group $g_i$ for user $u_b$,

$\bar{\theta}_{u_b}$ is the mean score assigned to user $u_b$,

$k$ is the total number of sight groups,

All users are classified with respect to the similarity of their interests in the sight groups with the active user. Then we select a group of neighbours for the active user $N(u_a) = \{u | u \in U \wedge u \neq u_a \wedge 0.7 \geq \zeta_g(u_a, u) \leq 1.0\}$

- **Similarity on sights of interest**

  This measure concentrates on similarity of the user interest in the sights between two users. We define the similarity factor of the active user $u_a$ and user $u$ as, the *Similarity Factor on the Sight of Interest*, $\zeta_s(u_a, u)$. We use $v(a,i)$ as a placeholder for the the score assigned to sight $s_i$ by user $u_a$ based on the type of information used. The entry $v(u,i)$ will be referred to as $v''(a,i)$ when we concentrate on the user travel history whereas $v'''(a,i)$ means both the user interest in the sight groups and their travel history are of interest. The calculation is conducted using Equation 4.5 (modification of Equation 4.1). Here, we propose to use the score we assign to the sights according the information in their travel histories and/or their user sight group profiles.

$$\zeta_s(u_a, u_b) = \frac{\sum_{i=1}^{n}(v(a,i) - \bar{\theta}_{u_a}) * (v(b,i) - \bar{\theta}_{u_b})}{\sqrt{\sum_{i=1}^{k}(v(a,i) - \bar{\theta}_{u_a})^2} * \sqrt{\sum_{i=1}^{n}(v(b,i) - \bar{\theta}_{u_b})^2}} \tag{4.5}$$

where

$\bar{\theta}_{u_a}$ is the mean score assigned to user $u_a$,

$v(b, i)$ denotes the assigned score to sight group $g_i$ for user $u_b$,

$\bar{\theta}_{u_b}$ is the mean score assigned to user $u_b$,

$n$ is the total number of sights.

All users are classified with respect to the similarity of their interests in the sights with the active user. Then we select a group of neighbours for the active user$N(u_a) = \{u|u \in U \wedge u \neq u_a \wedge 0.7 \geq \zeta_s(u_a, u) \leq 1.0\}$

#### 4.2.2.3 Recommendation Generation

The recommendations generated by the extended content-based algorithms are slightly different according to the user information used to form the *user-user information* matrix and to find the set of neighbours for the active user. We divide the recommendation generation into two categories. One is the recommendations generated from other users who have the similar sight groups of interest. The other is the recommendations generated from other users who have similar sights of interest.

- **Recommendation from similarity on sight groups of interest**

  In this case, neighbours of the active user have been chosen from the similarity between their interests in the sight groups. We compute a prediction $\lambda(u_a, g)$ from a weighted combination of selected neighbour's scores for sight groups $g$ the active user $u_a$ has not yet visited or given feedback scores to any sights in the group. We modify Equation 4.2 as shown in Equation 4.6. A weighted average of deviations from the neighbour's mean score is assigned to *no-score* sight groups.

$$\lambda(u_a, g) = \eta + \frac{\sum_{i=1}^{m'} (v'(i, g) - \bar{\theta}_{g_{u_i}}) * \zeta(u_a, u_i)}{\sum_{i=1}^{m'} \zeta_g(u_a, u_i)} \tag{4.6}$$

where

$\lambda(u_a, g)$ represents the *prediction* for the active user $u_a$ for sight group $g$,

$\eta$ is the *neutral score of 5* which represents a mean feedback score of the active user $u_a$,

$\bar{\theta}_{g_{u_i}}$ is the mean feedback given to sights in sight group $g$ by user $u_i$,

$\zeta_g(u_a, u_i)$ is the similarity factor of the sight group profile between the active user and neighbour $u_i$,

$m'$ is total the number of neighbours.

We apply the same criteria used to classify the recommendations in the collaborative filtering. We consider sight groups that have prediction values ranging from 7 to 10 as recommendations to the active user. Therefore, the recommendations $R_{u_a}$ contain sights in the sight groups that have high predictions $R_{u_a} = \{s_i | s_i \in g \wedge \lambda(u_a, g) \geq 7 \wedge i \in [1, n]\}$.

- **Recommendation from similarity on sights of interest**

  In this case, the neighbours of the active user are chosen from the similarity on their sights of interest. We compute a prediction from a weighted combination of selected neighbour's feedback scores for sights liked by the neighbours and the active user has not yet visited. We adjust equation 4.2 by replacing the mean given feedback by the active user, $\bar{\theta}_{u_a}$, with the neutral score ,$\eta$, as shown in Equation 4.7.

$$\lambda(u_a, s) = \eta + \frac{\sum_{i=1}^{m'} (f_g(i, s) - \bar{\theta}_{u_i}) * \zeta(u_a, u_i)}{\sum_{i=1}^{m'} \zeta_s(u_a, u_i)} \tag{4.7}$$

  Where

  $\lambda(u_a, s)$ represents the *prediction* for the active user $u_a$ for sight $s$,

  $\eta$ is the *neutral score of 5* which represents a mean feedback score of the active user $u_a$,

  $\bar{\theta}_{g_{u_i}}$ is the mean feedback given to sights by user $u_i$,

  $\zeta_s(u_a, u_i)$ is the similarity factor of the interests in sight between the active user $u_a$ and neighbour $u_i$,

  $m'$ is total the number of neighbours.

  The recommendations $R_{u_a}$ contain sights that have high predictions $R_{u_a} = \{s_i | s_i \in S \wedge \lambda(u_a, s) \geq 7 \wedge i \in [1, n]\}$.

In summary, we design the extended content-based recommendation algorithms [C1, C2, and C5] to address the *over-specialization* and the *new user* problems found in the pure approach. We employ knowledge of the collaborative filtering in order to form a group of users who have similar interests in the sight groups and/or travel history as the active user. We then recommend to the active user a set of sights liked by these users.

### 4.2.3 Algorithms Using Knowledge-based Paradigm

The knowledge-based recommendation algorithms do not explicitly require a knowledge of the user's preferences or their feedback scores given to any items.

Rather, the algorithms implicitly extract the user's behaviour from their past activities. Firstly, the knowledge contained in a repository of past user choices is exploited to initialize the recommendation process with a set of implicit preferences. Secondly, the system involves the user in a dialogue where the user is encouraged to provide critiques and feedback to the system recommendations. These critiques are then incorporated by the system into a new query that tries to better model the user's preferences [25]. The recommendation models A3 and D1 described in Chapter 3 are developed based on this assumption.

In this project, a typical scenario for the knowledge-based algorithms consists of a set of $m$ users $U = \{u_1, u_2, u_3, \ldots, u_x, \ldots, u_m\}$ and a set of $n$ sights $S = \{s_1, s_2, s_3 \ldots, s_x, \ldots, s_n\}$. The system deliberately collects a list of sight, $s$, each user, $u$, has visited in the user's travel history $H(u) = \{(s_j, t_j) | s_j \in S \wedge u \in U \wedge t_j \in T \wedge u\, visited\, s_j\}$ where $t_j$ is a time stamp of the visit.

To generate the pure knowledge-based recommendations [A3] for the active user, $u_a$, we assume that the user who has seen several sights in the group is interested in seeing more. Based on this assumption, firstly we examine the knowledge of past user choices contained in the active user's travel history $H(u_a)$. Secondly, we count the number of sights in each sight group $g$ the active user $u_a$ has visited, $count(u_a, g)$. Finally, we recommend a set of sights , $R_{u_a} = \{s_j | s_j \in g \wedge count(u_a, g) \geq 2 \wedge j \in [1, n]\}$, to the active user $u_a$ if the number of visits of the sight group $g$ meet the criterion ($\geq 2$). This follows the idea proposed in TIP 1.0.

The extended knowledge-based recommendation [D1] does not narrow down recommendations using a dialogue where the user is encouraged to provide critiques and feedback as conducted in [25, 31]. Rather, we apply the principle of the content-based algorithm. Information/facts stored about the sights or *topic*, $t$, are taken into account as a supplementary sight context. A sight group may be divided further into several sub sight groups based on the topics they have in common. For example, a sight group sculpture may be divided into several sub groups based on the information about the artist, size, etc. The user then will get recommendations about the sculptures which are created by the same artists as other sculptures they visited in the past.

In summary, the knowledge-based algorithms implicitly extract the user's behaviour from their travel history. The pure knowledge-based algorithm recommends sights in the sight group to the user if the number of visits of this sight group meet the criteria. The recommendations may be confined by using the information stored about the sights.

## 4.3 Summary

In this chapter, we described our design of the recommendation algorithms which are created for the following recommendation models:

- using the collaborative filtering A2, E1, E2, and E3;
- using the content-based recommendation A1, C1, and C5;
- the knowledge-based recommendation A3, D1.

For each of the above recommendation models, we transformed the outline described in Chapter 3 into the calculation formulas. We set up criteria for the selection of the neighbours and the predicted feedback for sights to be recommended to the user. The criteria are employed for both the collaborative filtering and the extended content-based algorithms. The detail of the implementation of these recommendation algorithms is explained in the next chapter.

# Chapter 5

# Implementation

This chapter is concerned with the implementation of the algorithms described in the previous chapter. We begin with an overview of the TIP system implementation. Then, we describe the architecture of the recommendation component in Section 5.2. Finally, the details of implementation are given in Section 5.3.

## 5.1 Overview of TIP System Implementation

The original prototype of TIP (TIP 1.0) is implemented in a client-server approach using a central database, supporting desktop clients as well as mobile clients on a hand-held device with appropriate interfaces [21]. The heart of the system is the filter engine cooperating with the location engine. The filter engine selects the appropriate information from the database depending on user and sight context whereas the location engine detects and collaborates the current location of the user with the system. In this prototype, the location engine is triggered whenever the user moves from their current location. A simple recommendation mechanism is employed in this prototype: the system recommends other sights in the same semantic group if the user has already visited at least two sights in this group.

The use of personalized recommendations that take into account all of the personal information provided by a user is introduced in TIP 2.0. The study reported in [19] adopted and modified techniques from recommender systems to extend the simple recommendations about the sights provided in the first prototype. As a result, more user information (e.g., preferences, feedback) is stored in the database and a recommendation component was embedded as another type of the filter engine. The current version of TIP (TIP 2.5) is being extended in order to support more services, e.g., navigation by maps where the current location and the location of sights is dynamically indicated on the map. All services are represented by a component on the server side. On the client side, representation is provided by a thick or thin client [14].

Figured 5.1 expresses the functionality and architecture model of the current version of TIP (TIP 2.5). The PostgreSQL object-relational database management system and the Java Servlet Technology are used to develop the

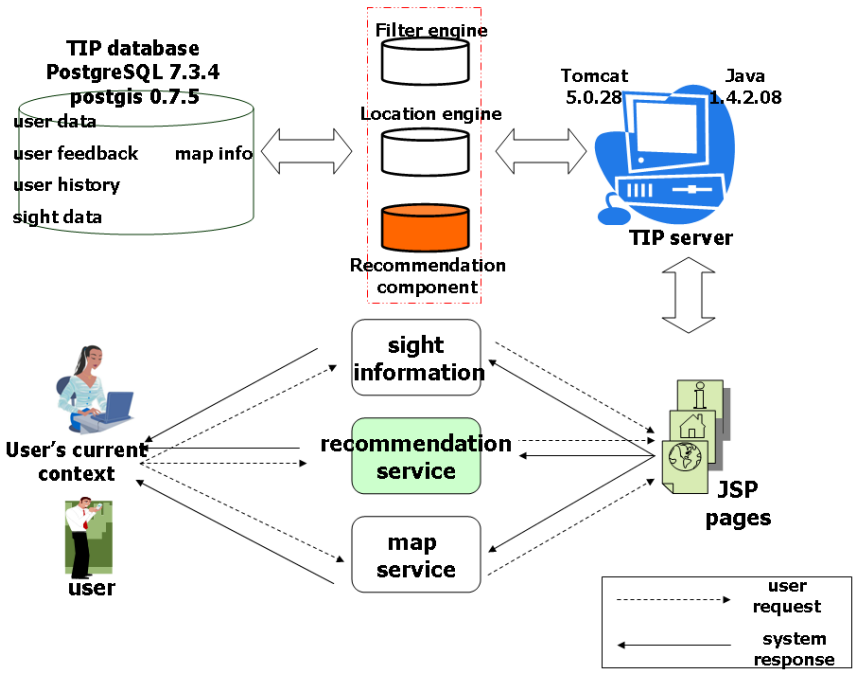



Figure 5.1: Recommendation service, map service and system architecture of TIP 2.5

TIP system. On the server side, Jakarta Tomcat 5.0.28 is deployed in order to work as a special servlet engine for the Java servlet technology. The Java version 1.4.2.08 Standard Extension is used for the implementation. The database is the PostgreSQL 7.3.4 with postgis 0.7.5 extension for the spatial or the location coordination part of the database. The client side can be a desktop computer or a pocket PC. The user sends their requests and receive corresponding responses via the JSP user interfaces which are altered according to the types of the devices being used.

## 5.2 Architecture of the Recommendation Component

Figure 5.2 shows the architecture model of the recommendation component for TIP 2.5. On a client side, the recommendation component has a *user interface* which provides three interfaces to a user: profile registration, feedback expression and recommendation operation. The users' profiles and their feedback together with sight information are stored in the data storage. The recommendation generating process on a server side is triggered when the *recommendation manager* receives a request for recommendations from the user. The recommendation manager then verifies which of the recommendation models is called for. The *request message* is simultaneously passed to both the *data collector* and the *filter engine*. While the *data collector* retrieves the required user and sight information from the data storage and prepares the information before passing it on, the *filter engine* selects the filter algorithm to be

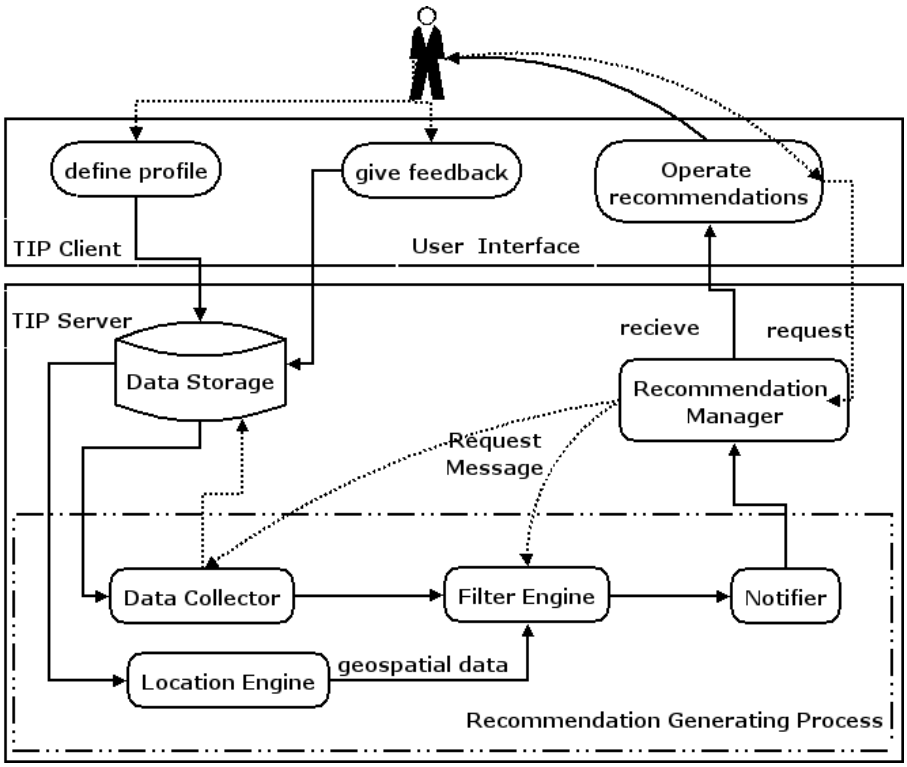



Figure 5.2: Architecture model of the Recommendation Component

employed. The current location of the user detected by the location engine is sent to the the filter engine, if required. After that, the generated recommendations are delivered to the notifier, which creates a notification and sends it back to the recommendation manager. Finally, the user obtains a response from the system indicating whether or not the recommendation option they have chosen is successful and the generated recommendations are delivered.

This architecture model is very general and thus leaves many details to be defined for an actual implementation. The next section describes each of the elements utilized in the recommendation component.

### 5.2.1 User Interface

Since the recommendation component requires a cooperation between the users and the system, the *User Interface* is a communication channel where interactions between these two parties happen. As shown in Figure 5.2, communications between users and TIP regarding their recommendation requirements occur in three circumstances. First, users need a way to manage their profiles. They are able to edit their profile on a dedicated page when they first register with TIP. They also need to be able to add a new interest or withdraw the existing interests from their profiles on this page later on. Second, users are required to express their opinion about a sight they visited in order to acquire more complicated recommendations from the system. They select a numeric value which represents their degree of satisfaction to a sight on a page dedicated to manage their feedback. The users' opinion about a particular sight

may change from time to time; thus only the latest feedback given to the sight is stored in the database. The third circumstance is when users request for and/or receive recommendations. In this case, a user interface is a simple page where an interchange between the user's request and the system's response occurs. In this prototype, a list of recommendation options, which represents our designed algorithms, is presented to the users. A selected option is passed on as a request message to the recommendation manager. After the entire recommendation generating process is completed, the result is displayed on the recommendation page. The result is either a list of the recommended sights and their detail information if the process is succeeded or an error message indicating which problem has occurred during the process.

### 5.2.2 Recommendation Manager

The *Recommendation Manager* works as a middle person who connects the user on the client side and the recommendation generating process on the server side. Figure 5.3 describes two functionalities of the recommendation manager: the user's request handling and the system's response management. This section is concerned mainly with the user's request handling since the system's response management will be explained in detail later on in the architecture model of the notifier. After receiving the *request message* from the recommendation interface, the recommendation manager forwards the message to the *Request Message Parser*. Hereafter, it is referred to as *RPM*. The RPM identifies the user's id and passes it on to the data collector. Meanwhile, the recommendation option is detected and forwarded to the filer engine. If failure occurs during the request message parsing, the error message is sent to the recommendation manager.

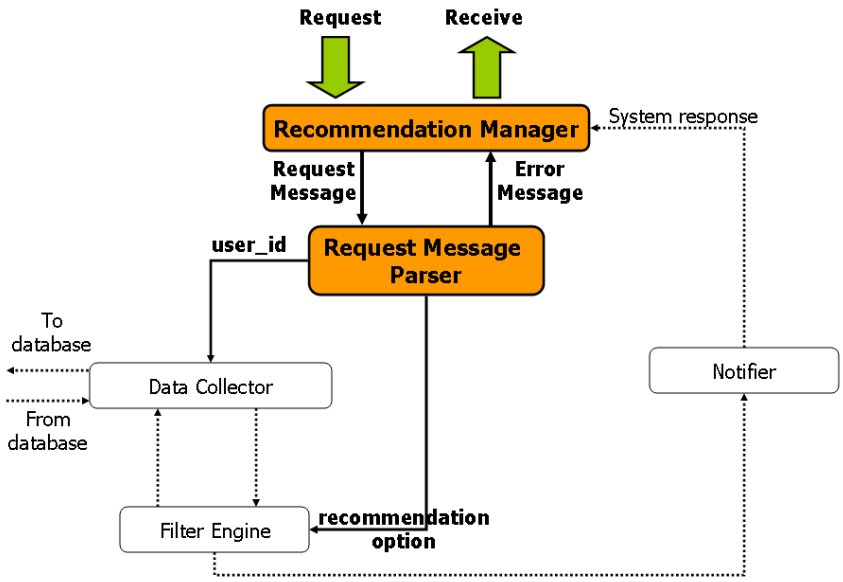



Figure 5.3: Architecture model of the *Recommendation Manager*

### 5.2.3 Data Collector

The *Data Collector* works as a gateway to the information source during the recommendation generating process, since it is directly connected to the data storage. Figure 5.4 shows the embedded mechanism called *data preprocessor* which coordinates the data collector with the filter engine. As each of the recommendation algorithms requires different types of user's and sight information, the filter engine passes on a message regarding which type of information is required. This message together with the user id passed on by the RPM is used to generate an SQL statement which is then forwarded to the database. The recommendation option indicates if the retrieved information is required to be processed before it is forwarded to the filter engine (see Section 4.2.1.1 and Section 4.2.2.1 for more detail).

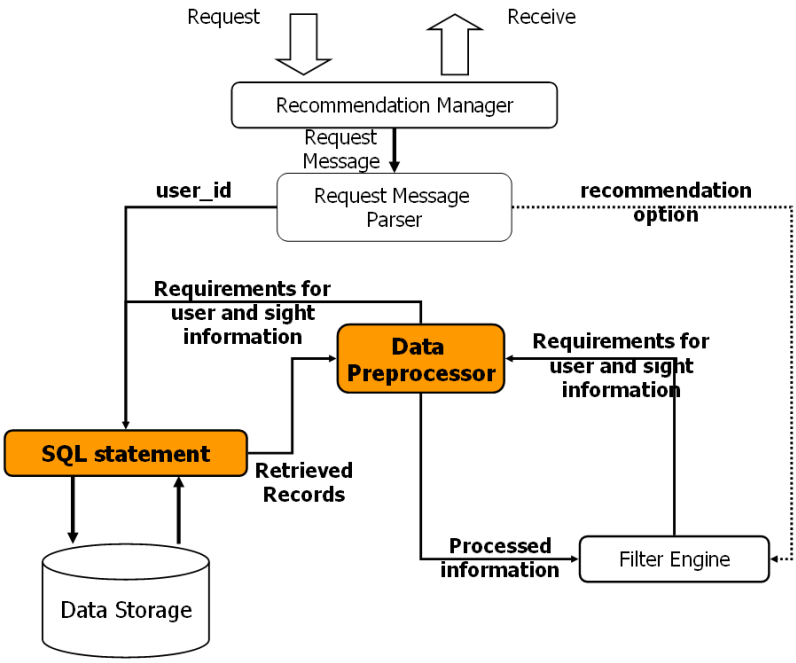



Figure 5.4: Architecture model of the *Data Collector*

### 5.2.4 Data Storage

This section is concerned with the information source or the *Data Storage* in the TIP system. Since TIP is implemented using a central database approach, we depict the data storage used as the entity relationship diagram shown in Figure 5.5. The main entities we consider are partitioned into two major parts the *information delivery* and the *user*. The information delivery consists of the sights and information about the sights whereas the user contains information about the users. The subsequent paragraphs explain these two parts in more detail.

As we have introduced earlier, sights in TIP are classified into *sight groups* or so called *clusters of sights*. The sight group is the collection of sights that have some characteristics in common. There are groups of sights that have

the same architecture (e.g., churches, beaches, arts etc.) which are called *by default* sight groups because they are obvious. Meanwhile, creating a sight group based on other features the sights have in common – e.g., location (city), time (17th century) etc.– is also substantial. Therefore, there can be a group of sights that are located in the same area (e.g., the public arts on Victoria Street, Hamilton) or a group of sights that were built in a certain period of time (e.g., the building built before 1900) and so forth. We refer to these sight groups as *semantic clusters*.

We addressed the idea of the hierarchical classification structure in order to classify a particular sight into its sight groups in Chapter 4. The structure indicates a flexible way of analyzing the content of a sight so that it can be put into default sight groups as well as generated semantic clusters. For instance, to classify the sight *the statue of Rocky Horror's Riff Raff*, we can do as follow:

- a hierarchical structure of default sight groups [*arts* → *public arts* → *sculpture* → *the statue of Rocky Horror's Riff Raff*];
- a semantic cluster of sculptures in a certain area, which in this case is Victoria Street, Hamilton, New Zealand [*New Zealand* → *North Island* → *Hamilton* → *Victoria Street* → *the statue of Rocky Horror's Riff Raff*];

and so forth. The entities *subSight* and *subSightgroup* shown in Figure 5.5 are used to store the hierarchical information of the sights and sight groups. For each sight groups, information about the sights in the group is categorized into several topics related to some available facts. The sight information is also stored in a hierarchy structure indicated by the entities *topic* and *subTopic* as shown in Figure 5.5.

The user side encompasses user data and user profile. The users define profiles regarding certain sight groups they are interested in and topics of the sight information they would like to know. These user interests in the sight groups are stored as the users' sight group profiles whereas the topic of interests are stored in the users' topic profiles. The user data contains their login id and password used to access the system, their feedback scores given to the sights in their past visits and their travel histories. The users' travel histories are retrieved when their GPS location is sent to the TIP system as the users are at a sight and interacting with the system. Since we concentrate on the data model used in the recommendation component in TIP, Figure 5.5 is part of the data model designed for TIP 2.5. The full ER Diagram is shown in Appendix A.

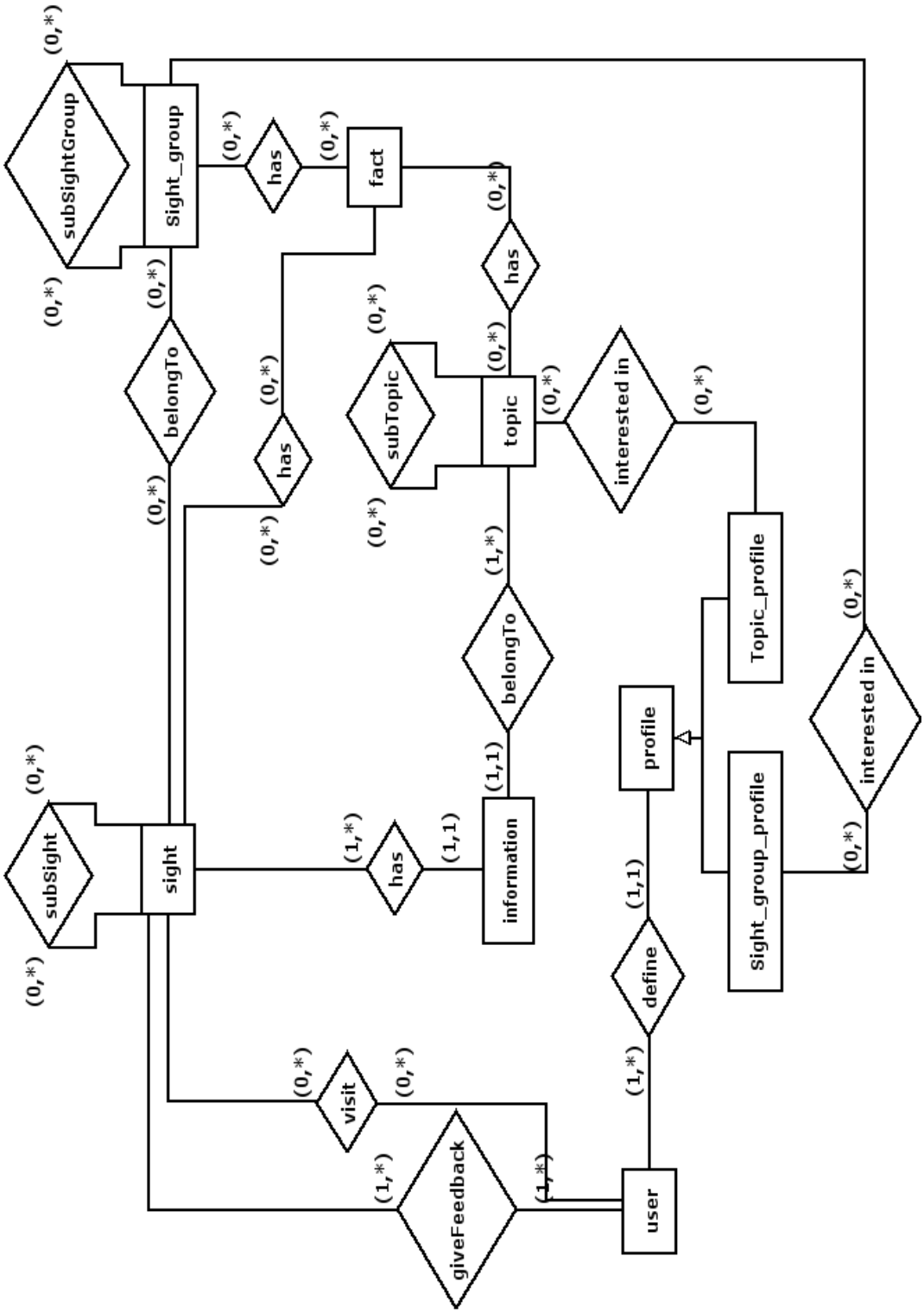



Figure 5.5: Data model used for the Recommendation Component in TIP

### 5.2.5 Location Engine

The *Location Engine* detects the current location of the user. This may be triggered by time (e.g., send every minute) or by location (e.g., send if the user moved more than 500m) or by user interaction with the system interface (e.g., a button being pressed to retrieve the new location). In the desktop-bound prototype, the user's current location is directly typed in using either name or coordinates of a sight. A collaboration between the filter engine and the location engine occurs when the user's current location is one of the constraints of the recommendation algorithms (e.g., a user is asking for sights of interest around the area). The *Geo-spatial Mapping Unit* in the location engine first maps the name of a sight to its coordinates. Then the area of interest is calculated. In this case, we define our area of interest as 500 metres from the centre of the current location. Then the result is sent to the filter engine. Figure 5.6 shows the interactions of the location engine.

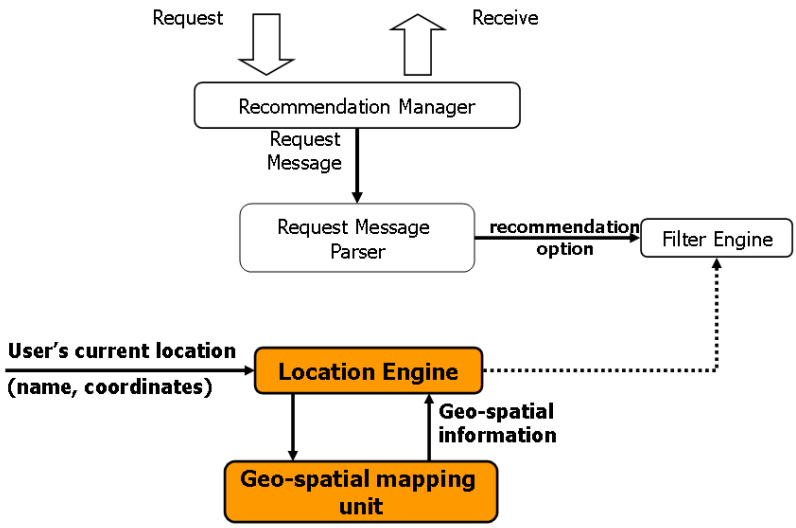



Figure 5.6: Functionalities of the *Location Engine*

### 5.2.6 Filter Engine

Figure 5.7 describes the architecture model of the *Filer Engine*. The filter engine is the heart of the recommendation component. It comprises all of the filter algorithms. When the filter engine receives a message from the recommendation manager, it first matches the recommendation option to its filter algorithm. Once the filter algorithm is assigned, the information requirement message is sent to the data collector. The basic idea of every filter algorithm is to collect sight information which matches the user's request and isolate the access information retrieved from the data collector (see Chapter 4 for details of each of the filter algorithms). Accordingly, all constraints conveyed

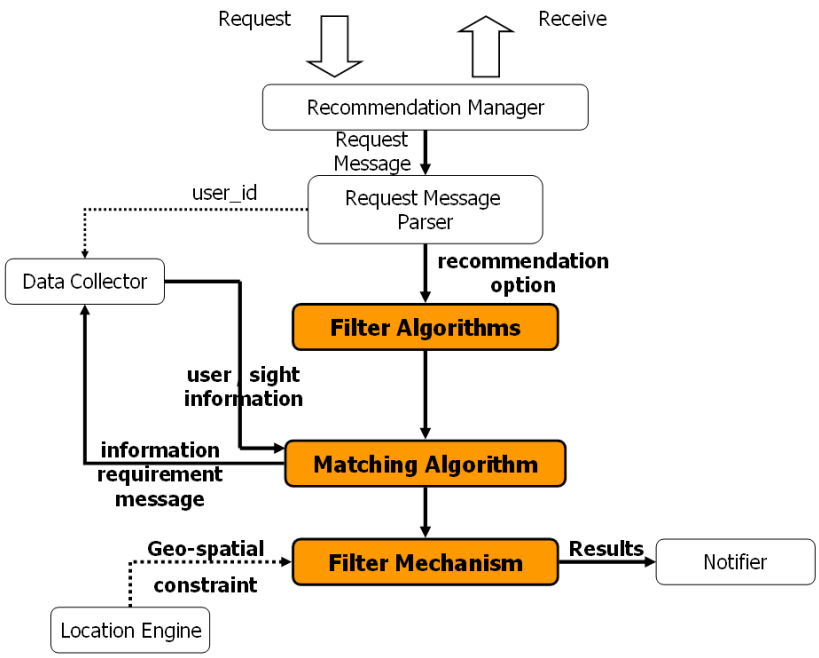



Figure 5.7: Architecture model of the *Filter Engine*

by the user's request message, including the geo-spatial constraint if required, are combined to generate the recommendations. After the entire process is completed, the result is passed to the notifier.

### 5.2.7 Notifier

The architecture of the notifier is straightforward. It works as a messenger who delivers the result of the filter engine to the recommendation manager. Either a success message together with a set of recommended sights or a failure message is sent to the recommendation manager. A failure occurs if there is no sight to be recommended from the recommendation options or there is an internal malfunction during the recommendation process. After receiving the message, the system's response management in the recommendation manager passes on this message to the user interface and the result is displayed to the user.

To summarize, we base the recommendation component in the TIP system mainly on the filter algorithms we illustrated in the previous chapter. Our design comprises six elements. A user communicates with the system via the dedicated user interfaces on the client side while the cooperation between the other five elements on the server side generate the recommendations for the user.

## 5.3 Details of Implementation

In this section, our main focus is on the details of the implementation. We begin this section with an overview of the structure of the implementation. Then we explain the details of our implementation using the UML class diagram.

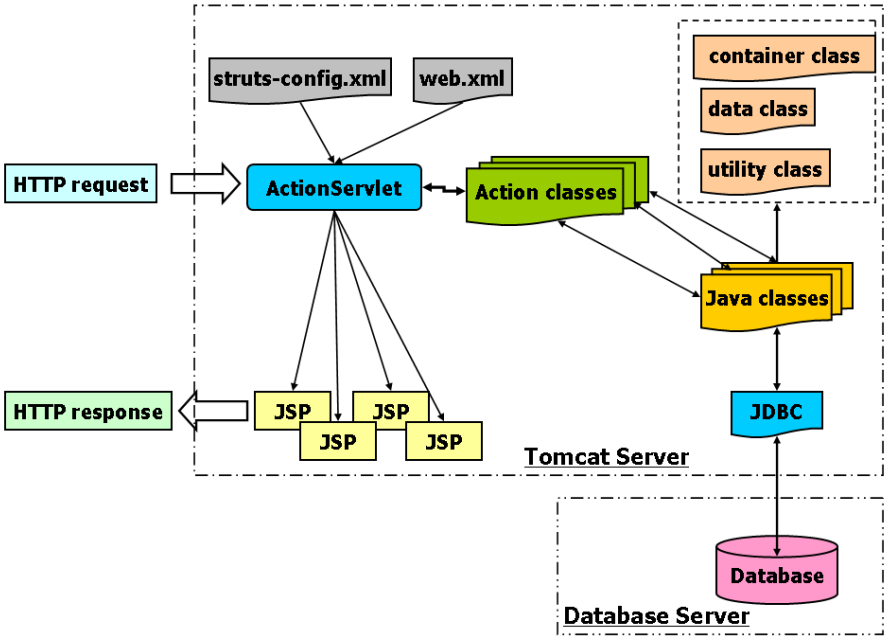



Figure 5.8: Implementation structure of the Recommendation Component

### 5.3.1 Implementation Structure

Figure 5.8 describes the structure of our implementation. Since TIP is implemented using the *Struts* framework [1] (see [26] for more details on the implementation of TIP 2.0), hereafter, we map our designed elements described in the previous section to the Struts 's structure. The following paragraphs explain the flow of our implementation.

- The user interfaces are presented as a typical html page in a web browser which a user will perceive as they connect to the system. These user interfaces are written in JSP.

- The duties of the *Recommendation Manager* are carried out by the Struts *ActionServlet*. It is in charge of deciding which Action class will process the *HTTP request* and which *JSP* will generate the *HTTP response*. Struts uses a configuration file to associate the Action class with the browser request. This configuration file is defined in a Web deployment file, *web.xml*. The file *struts-config.xml* configures the *ActionServlet*. The *ActionServlet* captures a client's *HTTP request* and passes it on to the *Action* class.

[1] struts.apache.org

- The *Action* class is the user-written controller component and it is in charge of processing the request. Here, the client's *request message* is used to select the filer algorithm from the *Filter Engine*. Meanwhile, a connection to the *Data Storage* is established by the *Data Collector* via *JDBC*. This control flow is done with help from other *Java classes*.

- The other *Java classes* hold the application process and data. We partition the Java classes in this implementation into three categories, the *utility classes*, the *data classes* and the *container classes*. We refer to the utility classes as a combination of the *Filter Engine*, and the *Location Engine*. The *Filter Engine* consists of 14 recommendation classes each of which represent the recommendation algorithm explained in Chapter 4 whereas the the *Location Engine* comprises users' location management classes. The *data classes* and the *container classes* are used to handle the user and sight information retrieved from the database.

- After the request is processed the response is sent back to Struts *ActionServlet*. The task of the *Notification* element is conducted by the internal mechanism of Struts. The *ActionServlet* then decides which JSP page it will forward the processed *HTTP request* to. Finally, the JSP with help from other Java classes generates the *HTTP response*. The *HTTP response* is sent to the client.

So far, we have described the structure of our implementation by mapping the six elements of the recommendation component with the Struts framework. The Struts *ActionServlet* is the main controller component whereas the *Action* classes control the flow of the implementation. With help from the Java *utility classes*, *data classes* and *container classes*, the response to the client's request is generated and displayed to the user via *JSP*. The subsequent section illustrates the implementation of the recommendation component using this structure.

### 5.3.2 Implementation

We begin this section with an overview of our implementation which comprises all the classes representing all of the recommendation algorithms we implemented. Then we select one of the algorithms and use it as a representative for the detailed explanation of the implementation.

Our implementation follows the structure described in the previous section. A UML diagram shown in Figure 5.9 describes all the classes used for generating recommendations to the users which are currently employed in the recommendation component. Here, we concentrate on the Action classes and the Java classes illustrated in the implementation structure shown by Figure 5.8. All of the Action classes are descended from *org.apache.struts.action.Action*. To generate recommendations using a particular algorithm, the corresponding action class calls a utility class which represents this recommendation algorithm. User and sight information retrieved from the database is stored in their corresponding data classes whereas the container class wraps up all of

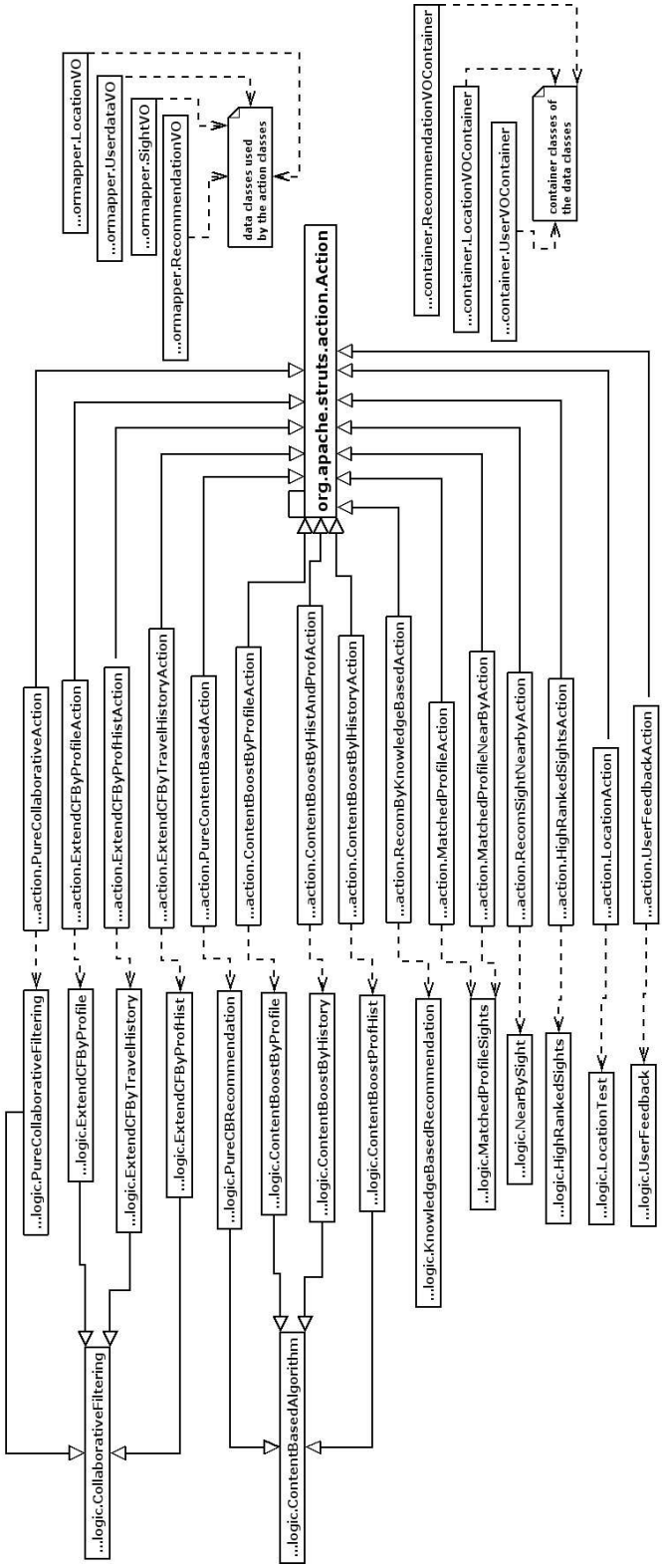



Figure 5.9: Overview of the implementation of the Recommendation Component

their related data class (e.g., the *SightVOContainer* concludes each of the sight information stored in every instances of the *SightVO* class).

As we can see from the diagram, the corresponding classes which represent the recommendation algorithms using the *collaborative filtering* paradigm extend the *CollaborativeFiltering* class while the corresponding classes of the *content-based recommendation* paradigm extend the *ContentBasedAlgorithm* class. The *KnowledgeBasedRecommendation* class corresponds to the *RecomKnowledgeBasedAction* action class which represents the recommendation algorithm using the *knowledge-based recommendation* paradigm. The *MatchedProfileSights* class correspond to two action classes: the *MatchedProfileAction* class and the *MatchtedProfileNearByAction*. The former class filter out sights that the active user did not define as their interests whereas the latter class take the user's current location as another constraint to filter out unrelated sights.

To manage a feedback score given to a particular sight by the active user, the *UserFeedbackAction* class calls its corresponding class, the *UserFeedback*. The given score is validated and inserted into the database if it is new. Otherwise, the existing feedback score is updated with the newly entered data. The *LocationVOAction* class takes control of the information of the active user's current location. Its corresponding class, the *LocationTest* validates the incoming data. If the user types in the name of a sight, it is mapped to its related coordinates. Then the coordinates are used in the recommendation algorithm, if required.

From this overview, we showed all the Action classes and Java classes which hold the key mechanisms of the recommendation component. In order to deliver a comprehensive explanation of our implementation we first select one of the recommendation algorithms, the *pure collaborative filtering* as a representative of the recommendation algorithm. Then we explain our implementation in detail by following the implementation structure given in Section 5.3.1. Figure 5.10 shows a UML class diagram which illustrates the implementation of the *pure collaborative filtering* algorithm. The pure collaborative recommendation runs as follows:

1. The *ActionServlet* which is an *HTTPServlet* receives a request from the browser and forwards it based on a configuration file: *struts-config.xml*. The processing logic is delegated to the *RequestProcessor* which asks the specified *Action* instance to handle the request via the method *processActionPerform*. In this case, it is the The *PureCollaborativeAction* class.

2. The *PureCollaborativeAction* class is the main class in our recommendation process. It is descended from *org.apache.struts.action.Action*. We overwrite the most important method of the Action class, the *execute()* method, as shown in the diagram.

3. The utility class named *PureCollaborativeFiltering* which extends the *CollaborativeFiltering* class is called by the *execute* method. Meanwhile, a connection to the database is established by implementing the *Datasource* interface which represents an engine for connection to the TIP database.

4. During the process, two data classes the *UserdataVO* and the *LocationVO* are used to hold the user and sight information retrieved from the database. The *RecommendationVO* class is used to store the information about the recommended sight whereas the *RecommendationVO-Container* is a container class for all of the recommendations.

5. When the entire process is finished, the *execute* method creates the corresponding response and returns an *ActionForward* instance describing where and how the control should be forwarded.

6. The response is forwarded to the corresponding destination as defined in the *struts-config.xml*, in this case, it is the *pureCF.jsp*.

7. If the recommendation process succeeds, the forwarded page retrieves the data from the *HelperBean* (*RecommendationVOContiner*) which holds the data required for displaying to the user. Otherwise, an error message is shown.

## 5.4 Summary

The design of the recommendation component implemented in TIP 2.5 consists of seven elements. These elements include:

- *the User Interface* which is deployed on the client side as a communication channel between the users and the recommendation component;

- *the Recommendation Manager* which works as a middle person who connects the user on the client side and the recommendation generating process on the server side;

- *the Data Collector* which works as a gateway to the information source during the recommendation generating process. The data collector retrieves required information directly from the data storage and prepares the information, if required, based on the information requirement message from the filter engine;

- *the Data Storage* which is depicted as a relational database. The main entities we consider are partitioned into two major parts. One is the sights and information about the sights. The other is the users and their personal information;

- *the Location Engine* which detects the current location of the user, maps the location name to its coordinates and calculate its geo-spatial information;

- *the Filter Engine* which is the heart of the recommendation component and comprises of all of the filter algorithms;

- *the Notifier* which works as a messenger who delivers the result of the filter engine to the recommendation manager.

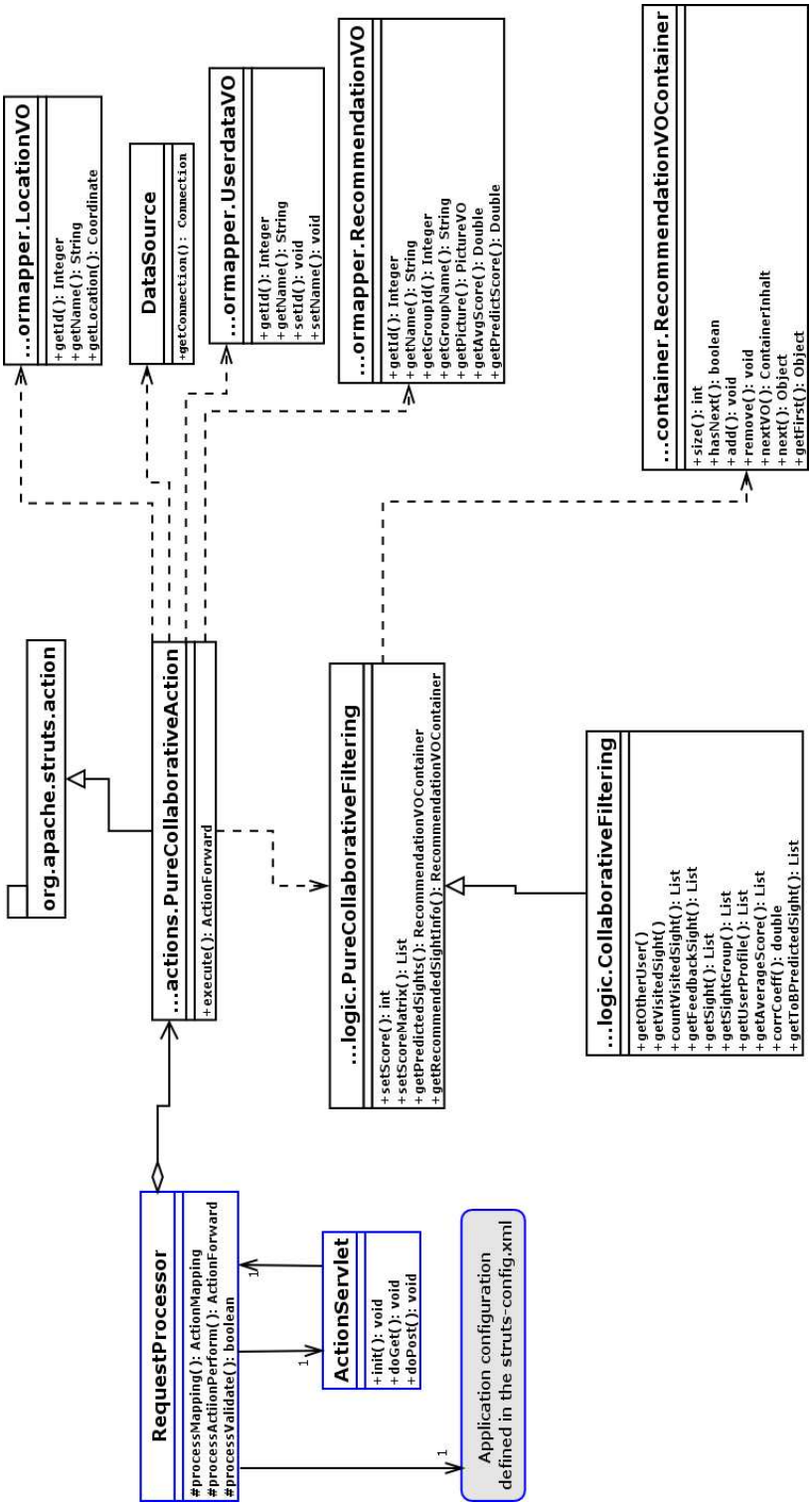



Figure 5.10: Implementation details of the *Pure Collaborative Filtering Algorithm*(UML) class diagram)

Since TIP is implemented using the *Struts* framework, we mapped the six elements described in the implementation architecture to its structure in order to form our implementation structure. We selected the *pure collaborative* algorithm as a representative and used it to explain details of our implementation. To verify the effectiveness of the implementation of our recommendation component, we propose the evaluation plan and conduct the analysis in the following chapters.

# Chapter 6

# Evaluation Framework

This project concentrates on not only the importance of the concepts, functionalities and features of the proposed recommendation models but also the effectiveness of their implementations. In this chapter, we design the evaluation plans for our implementations in order to verify that the limitations of the recommendation models identified in Chapter 1, Section 1.2 are remedied. First, we discuss the context of evaluation of recommendation systems. Then, we describe an outline of our evaluation for the implemented recommendation models. Throughout our discussion, we separate out our review of what has been done before in the literature from the introduction of our evaluation framework.

## 6.1 Context of Evaluation

Recommendations offer a way of taking a shortcut to the things we like without having to try many things we dislike or without having to acquire all the knowledge to make an appropriate decision. When we act upon recommendations from other people in daily life activities, we assume that the recommender has sufficient knowledge of our preferences or of the preferences of people like us or the recommender has knowledge of the available alternatives [11]. As recommendation systems have been increasingly used on many e-commerce web sites, appropriate evaluations to confirm these assumptions on a given recommendation have also been demanded.
Herlocker et al. [13] studied the key decisions in evaluating collaborative filtering recommender systems. They argued that conducting effective and meaningful evaluation of recommender systems is challenging. So far, there has been no published attempt to synthesize what is known about the evaluation of recommender systems, nor to systematically understand the implications of evaluating recommender systems for different tasks and different contexts. Therefore, they proposed the overview of the factors that should be considered for the evaluation. The following paragraphs describe these factors in more detail.

1. *Goal of evaluation*

   Different aspects might be focused on when evaluating recommendation algorithms. To validate a given recommendations algorithm, we may focus specifically on the *quality* of the recommendations. This can be conducted by measuring the *accuracy* of the filter algorithm, for instance, comparing errors between the predicted feedback score generated by the algorithm and the actual feedback score expressed by the user for a certain item. We could also consider the percentage of items for which a filtering algorithm can provide predictions or *coverage* [37]. Other options would be experiments on the response time of the algorithm to determine the *performance* of the system over a certain amount of the experimental data. In addition, since the recommendations are used to support decisions, it can also be valuable to measure the *user satisfaction*. For this reason, some commercial systems measure user satisfaction by the number of products purchased and not returned while the non-commercial systems may directly ask the users about the level of their satisfaction toward the system. According to the above arguments, we need to precisely identify for what aspect of the recommendation algorithm is important to be evaluated.

2. *Types of analysis and data sets*

   Several key decisions regarding data sets lie beneath successful recommendation algorithms. Different algorithms may perform better or worse using different data sets. Many collaborative algorithms have been designed specifically for a particular domain, e.g., movies, books, research papers etc. The algorithms designed for a domain where there are many more users than items (e.g., the Amazon.com data set has 29 million customers and several million catalog items [20]), such algorithms may be entirely inappropriate in a domain where there are many more items than users (e.g., a research paper recommender with thousands of users but tens or hundreds of thousands of articles to recommend).

   Apart from a selection of appropriate experimental data sets for which domain a given recommendation algorithm is employed, there are two further issues to be taken into account. One is considering to conduct a live user experiment or off-line analysis. The other is to generate synthesized or use natural data sets in the experiment. Conducting off-line analysis has the advantage that it is quick and economical to conduct large evaluations, often on several different data sets or algorithms at once. Typically, it is applied for analysis of predictive accuracy where predictions of the withheld values from the data set generated by the algorithm are compared to the actual values. Meanwhile on-line or live experiments are conducted in order to evaluate user performance, satisfaction, participation and other measures as conducted in [11]. An existing data set of a given recommendation system may not perfectly match the properties of the target domain and task. Accordingly, synthesizing a data set specifically to match those properties may be a good approach especially at the early stage of designing the algorithm [13].

However, it is very important to keep in mind that drawing comparative conclusions from a synthetic data set is risky in the long run, because the data may fit one of the algorithms better than another.

3. *Combination of measures used in comparative evaluation*

   Deciding what combination of measures to use in comparative evaluation is significantly challenging. Most of the current studies on evaluating recommendation algorithms focus on finding new filtering algorithms which appear to do better than the older algorithms they are compared to [11, 20, 27, 37]. The study reported in [13] suggested that algorithmic improvements in filtering algorithms may come from different directions than just continued improvement in accuracy differences. Possibly the best filtering algorithms should be measured in accordance with how well they can communicate their reasoning to users, or with how little data they can yield accurate recommendations.

So far, we learned that difficulties in evaluating recommendation systems, and in judging which criteria to use when making the evaluation, have led to a demand for appropriate system evaluations. Although study of evaluating recommendation systems is still underway, the implications from the study in [13] inspire us for the creation of our evaluation plan. The following sections illustrate an outline the of the evaluation for our recommendation models.

## 6.2 Outline of the Evaluation

In this section, we present an outline of the evaluation for our implementation by following the implications we identified in the previous section. We first briefly reaffirm the existing limitations described in Chapter 1. Then we verify the goals for which our evaluation will be conducted regarding these limitations. Finally, we illustrate our evaluation plan which comprises types of evaluation techniques and data sets.

### 6.2.1 Existing Limitations

The following paragraph concisely depicts the existing seven limitations we verified in Chapter 1. These limitations include:

- **New user problem** which addresses the lack of initial information for building a user model;
- **Cold start problem** which illustrates the sparsity of the coverage of feedback scores used in the collaborative filtering approaches, especially at the beginning of the system;
- **Specific preferences user or the *Grey Sheep* problem** whose preferences are sometimes relatively unusual compared to other users in the system. The inconsistency of their opinions leads to poor or inaccurate recommendations;

- **Over-specialization** which gives restricted recommendations to the user. This circumstance happens when the system can only recommend sights similar to those already rated sights which are highly scored against the the user's profile;

- **Transparency and user control** in which the concept has been applied but no evaluations have been conducted yet;

- **User satisfaction** which reflects the quality of the implemented recommendation algorithms used to build user models and the recommendation component;

- **Scalability** which explains the cost of computations regarding the number of users and items in the system.

These limitations indicate the essential combination of measures to be employed in our evaluation plan.

### 6.2.2 Goals of Evaluation

In order to make our evaluation strategy more straightforward, we first partition our goal of evaluation into three aspects based on the limitations listed above, *effectiveness* and *performance* of the implemented recommendation algorithms as well as the *presentation and interactions* of the recommendation models. We then explain how each of these aspects address the existing limitations. The subsequent paragraphs describe these three aspects in more detail:

1. **Effectiveness of the algorithms**

   The focus of this aspect is on the first four limitations: the *new user problem*, the *cold start problem*, the *specific preferences user problem* or the *Grey Sheep* problem, and the *over-specialization* problem. We believe that verification and comparison on the implementation results will point out some preliminary indications as to the specific strengths and weaknesses of the algorithm. This will lead to remedies which we design to correct a shortage of required information in the recommendation process.

2. **Performance of the algorithms**

   Giving recommendations in a mobile tourist information system tends to encounter *scalability* problems when the system is increasingly utilized by the users. This problem leads to the system's performance issue. In this aspect, we concentrate on which of the recommendation algorithms outperforms the others with a changing number of users and the number of sights. We believe that examination of the time required to generate the recommendations and the ability to provide predictions to the user with many of the sights the user has not yet rated or *coverage*, will indicate the performance of the recommendation algorithms.

3. **Presentation and interaction of the recommendation models**

   The interface and interaction model provided by the system influence the last two limitations, *transparency and user control* and the *user satisfaction*. We strongly believe that fluid and traceable recommendations are important for the user's acceptance of a mobile tourist information system. However, our main focus in this project is on the design and implementation of the recommendation models. We therefore determine the following criteria for the design of the interface and interaction model for the recommendation component in TIP [16].

   (a) *Acceptability*
   Since the interaction with the system is not a on-off situation, but rather a continuous way that slowly adapts the system's behavior to the user, therefore the fulfilment of this criterion needs to be tested using the user test groups and exemplary recommendation lists. We plan to utilize this prototype for a certain period of time in order to collect both the users' and the sights' information. Then the usability test will be conducted. For this study, it will be reasoned based on previous studies.

   (b) *Tractability*
   This is a long term functional test regarding user acceptance. The system's reasoning, e.g., the basis of the recommendations, must be transparent to the user so that they are able to control and manipulate the system's reactions. Otherwise the users will not accept the system for a long term usage. The fulfilment of this criterion requires a longitudinal user study. A preliminary set of reasonable expectations will be argued for in a first-cut analysis. Still, further validation based on a user study will be conducted later on in the next project.

In summary, to confirm that implementation of the recommendation component in TIP successfully addresses the seven limitations, we need to conduct a combination of measures. We believe that evaluations of our recommendation models should measure not only how effective they filter out unrelated information but also how well they can convey their reasoning to users, and/or with how little data they can yield accurate recommendations. The subsequent section describes our evaluation plan.

### 6.2.3 Evaluation Plan

This section illustrates the evaluation plan which includes the employed evaluation techniques and the data sets. According to the goals of evaluation we classified in the previous section, we separate our evaluation plan into two aspects. In the first evaluation plan, we focus on effectiveness of the designed algorithm and the acceptability of the presentation and interaction of the recommendation models. We refer to this as the *qualitative evaluation*. The second plan is concerned with performance of the algorithms which we refer to as the *quantitative evaluation*.

- **Qualitative evaluation**

  We propose to verify the *effectiveness* of the recommendation algorithms and the fluidity of the recommendation component by using an explanatory study through the two test scenarios we introduced in Chapter 1. We will investigate whether the system can effectively respond to the user's request and provide useful explanations and justifications for giving recommendations to the users. Our intention is to test both new and experienced users. We establish our test setting based on the existing tourist information in the TIP system. The users' information is the natural data gained from the current usage of the system by the TIP development team. The sight information is supported by the Hamilton City Council. More detail about about the entire qualitative evaluation is given in Chapter 7.

- **Quantitative evaluation**

  We propose to examine *performance* of the recommendation algorithms from *response time* to a user's request which indicates *computational complexity* of the algorithms. We also verify *coverage* of the recommended items which indicates the predictions the system can provide regarding the items the user has not yet given their feedback. Our focus in mainly on the recommendation algorithms using the knowledge of the *Collaborative Filtering* paradigm. The test setting consists of the off-line synthesized users and sights information. More detail about the entire quantitative evaluation is given in Chapter 8.

We believe these two evaluation plans depict a good combination of measurements for our implementation since we try to balance our examinations on both the *qualitative* aspects (e.g., effective recommendations and well-defined recommendation delivery techniques) and the *quantitative* aspect (e.g., complexity of the algorithms). Therefore, we expect that the results from these two evaluation plans will provide an overview which will be very useful for further study on the recommendation component for TIP.

## 6.3 Summary

This chapter is concerned with the evaluation framework fro the recommendation component we implemented in the project. We started by conducting our investigation on the existing literature. Currently, there has been no published attempt to synthesize what is known about the evaluation of recommender systems, yet evaluation of these systems is still underway. Nevertheless, we learned that judging the efficiency of recommendation algorithms requires several measurements. We therefore applied the results we obtained from this investigation to evaluate our system against seven limitations identified at the beginning of this project. We categorized these limitations into three subjects: *effectiveness*, *performance*, and *presentation* of the recommendation component. We proposed two main evaluation plans. One is the *qualitative evaluation* in which we will examine effectiveness and presentation of the algorithms.

The other is the *quantitative evaluation* which concentrates on performance of the algorithms. We expect the results from a combination of these two measures will be useful for further improvement of the recommendation component in TIP. More detail on these two evaluation plans will be described in the subsequent two chapters.

# Chapter 7

# Qualitative Evaluation

In this chapter, we conduct the qualitative evaluation on the implementation of the recommendation component. Our objective is to verify the effectiveness of the recommendation algorithms and the fluidity of giving the recommendations to the users. We first introduce the experimental data used in the setting of the evaluation. Then we show in detail the effective use of the implemented recommendation algorithms through the user scenarios and screen shots captured from the TIP system.

## 7.1 Setting of the Evaluation

In this test setting, we carry out our evaluation using an explanatory study. We use the currently employed data set of the TIP system. This data set consists of the sight information of public art located in Hamilton, New Zealand and the information of the current users used by the TIP development team. The sight information comprises 83 items of HCC-owned and the private-owned public arts in Hamilton. These sights are classified into two major sight groups (HCC-owned and private-owned) and eight sub sight groups (mural, painting, wall hanging, stained glasses, drawing, tapestry, tiles, and mosaic). The users information consists of 69 users. Each of the users owns one sight group profile and one topic profile. The user's sight group profile contains at least one sight group of interest. There are 107 feedback scores and 143 records of the users' travel histories. The input of GPS data is simulated by direct insertion of the required data in the user interface and the database. We use examples of clusters that have been built in the TIP system such as:

- sights that are of the group *arts* and have subgroups as *privately-owned public art* or *Hamilton City Council-owned public art*.
- sights that have the same value of attribute 'address', e.g., Hamilton Garden, Waikato Museum etc.
- sights of the same group located in a certain area, e.g., all *sculpture* around the Central Library and Garden Place, all painting in the Central Library.

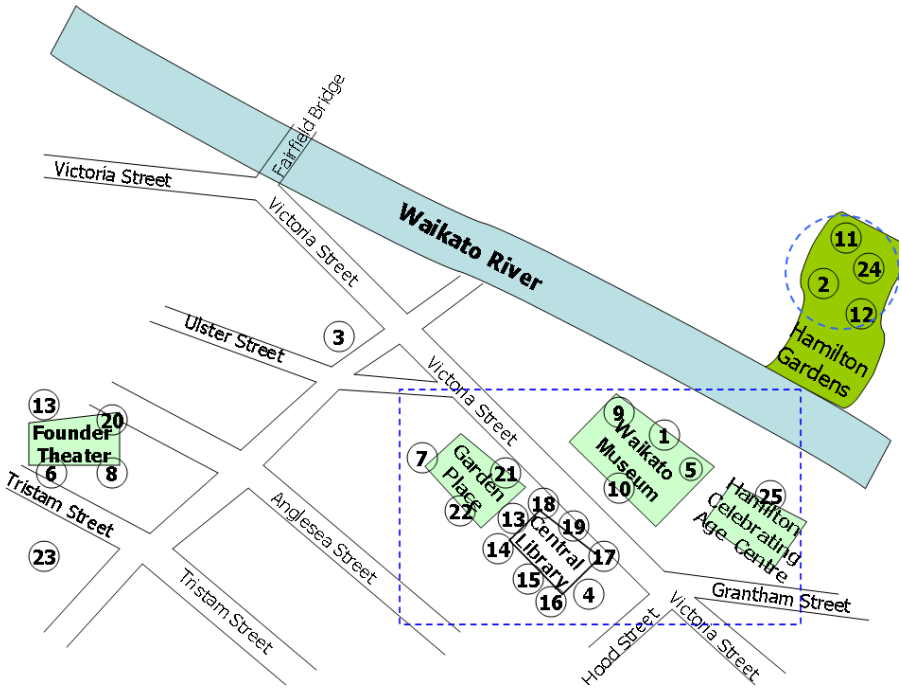



Figure 7.1: A map of the example data sights

Figure 7.1 shows a map of some of the example sight data used to test the recommendation component. Our test setting consists of two areas of interests. The dashed square describes the first while the dashed circle describes the second area of interest for this chapter. The following types of sights are used in our test scenario. The numbers in the parenthesis refer to the number of sights in Figure 7.1.

- **Sculpture**: Ripples (1), Little Bull (2), The Farming Family (3), The Librarians (4), Millennium Family (5), Hilda Ross Fountain (6), Garden Place Fountain (7), Centennial Fountain (8), In Memory You are Known (9), Museum Fountain (10), Hamilton Gardens Mural (11), Little Bull (12), Giraffe (13), Waves (23), The Torso (24)
- **Painting**: Founders Theatre Mural (13),'Odd Numbers Side, Victoria Street, Hamilton, New Zealand 1997' (14), 'Civic Square 1993' (15), 'This Was Our Library, Hamilton 1993' (16), 'Horizons' (17), 'Bush' (18), 'Little Owls'(25)
- **Tapestry**: Great Wall of China (19)
- **Wall Hanging**: Hamilton Landscape(20)
- **Mosaic**: Whariki (21),Peace Wall(22),

## 7.2 Results of the Explanatory Study

This section describes the results of the explanatory study. Here, we show how the system can help the users to get through their request for recommendations.


(a) Sight group selection (b) Topic selection

Figure 7.2: Joey's sight groups and topic profiles

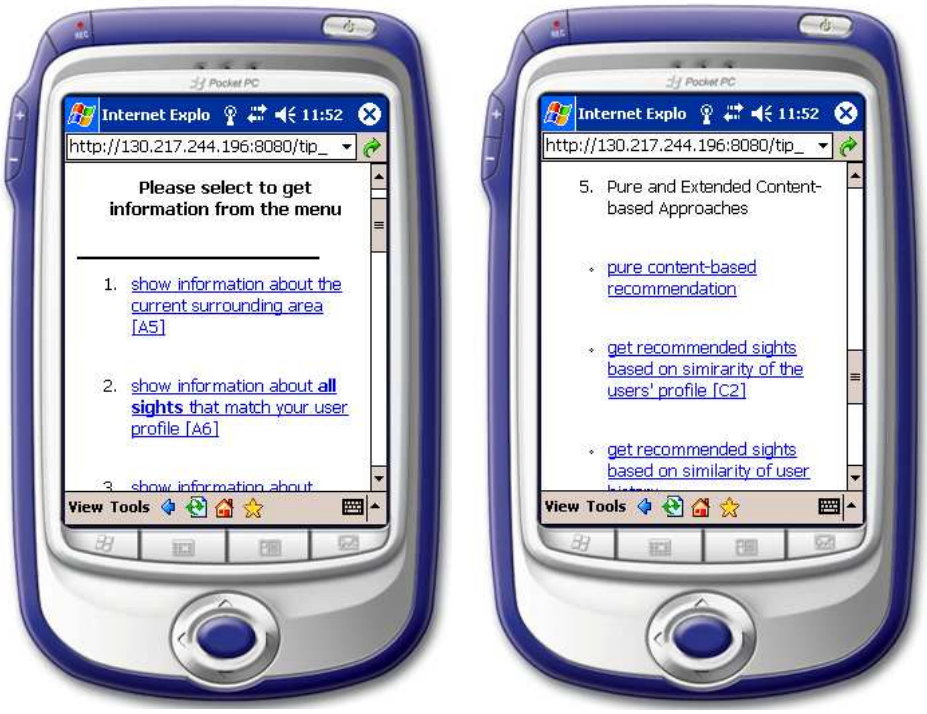
Then, the analysis of these results is given in the subsequent section. For clarity, screen shots of the employed recommendation models are captured from a personal computer (PC) using the Microsoft Pocket PC 2003 Emulator instead of the actual mobile devices. We captured the actual coordinates of the test sights and used them to simulate the signal of the GPS data in this test setting.

We revisit our application scenarios introduced in Chapter 1, to see if the implementation of these recommendation models would satisfy Joey and Anne. We start with our new user, Joey. Joey defined his interests to the system when he first registered. This is to let the system know what he is interested in so that the system can provide him with personalized information. Joey defined in his sight group profile that he is interested in *mural, painting, sculpture, and wall hanging* as shown in Figure 7.2(a). He also informed the system types of information about the sights he is interested in so that he will not receive all available sight information but only the selected ones. Figure 7.2(b) illustrates Joey's sight information selection for his topic profile.

The next morning, Joey starts his travel at the Waikato Museum. After he finishes his visit, he wants to get recommendations for his next visits. He sends his location to the TIP system. The system detects that Joey is now on the river bank below the museum and his current location is near to the sculpture named *Ripples*. In general, the system's main screen presents a picture of the sight as well as a link which the user can click for its general information as shown in Figure 7.3(a). Figure 7.3(b) shows the general information and the name of the artist according to the topic information Joey defined in his topic profile. He can also get more detailed information by clicking on given link.


(a) TIP main screen (b) sight information screen

Figure 7.3: TIP main screen and sight information screen of *Ripples*




(a) The first three major options (b) The content-based recommendation options

Figure 7.4: Recommendation menu

Since this is the first time he has used the TIP system Joey does not yet feel confident that the system will respond to his requests correctly. He wants to do some testing on the system before he really uses it. Joey then decides to check the recommendations service of the system. In this prototype of TIP, we provide seven major recommendation options to the user as shown in Figure 7.4. Among these options, the content-based and the collaborative recommendation option each comprise four other minor options which indicate the type of user information used in the recommendation generating procedure (see Figure 7.4(b)). Hereafter, each of these options is explained.

Joey first clicks on option number 2 *show information about all sights that match your user profile* ( see Figure 7.4(a)) in order to prove if the system will deliver sights that he is interested in. The system takes the information he has registered in his sight group profile. Then the sight information is filtered and a set of matching sights is returned to him.

Figure 7.5 illustrates the recommendation given by this option. There is a brief explanation of the information used to generate recommendations on the top of the screen. Each of the recommendations consists of the name of the sight, the sight group this sight belongs to and a link which the user can click to get sight information and a picture. According to Joey's request, all sights belonging to the four sights groups *(mural, painting, sculpture, and wall hanging)* defined in Joey's sight group profile are given.

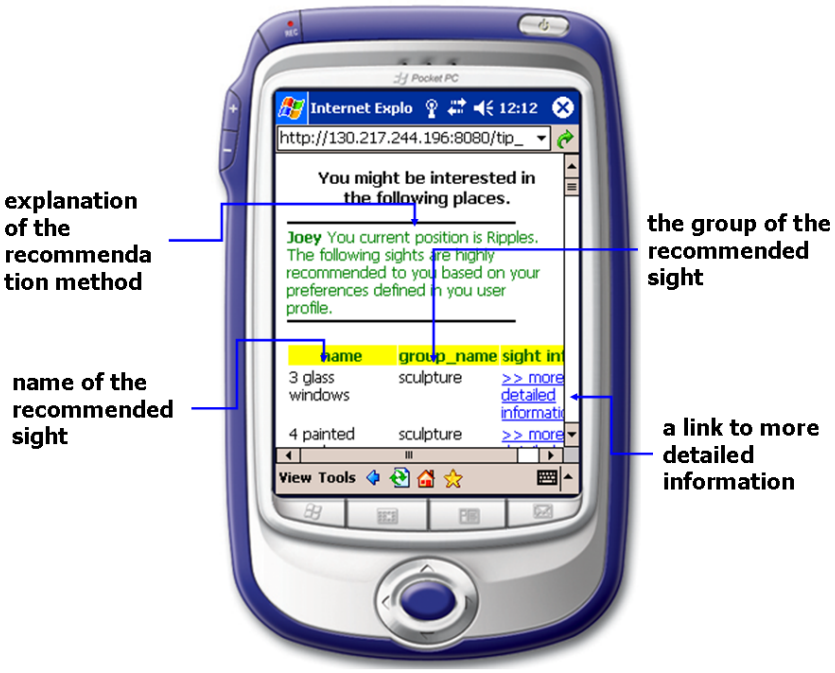



Figure 7.5: A list of all sights that match Joey's sight group profile

After reading through the set of recommended sights, Joey gains confidence in the system. He is convinced that the system does take his interests into account to generate the recommendations for him. However, the given recommendations contain too many sights. Therefore, he tries another option in order to narrow down these choices by selecting option number 3 *show information about nearby sights that match your user profile*. In this test setting,

we define sights that are located no farther than 500 metres from the user's current positions as nearby sights which can be changed later on. Consider our area of attention indicated by a dashed line in Figure 7.1; Joey is now at sight number 1 on the map. When he selects this recommendation option, the information about his current position together with the information about sight groups he is interested in, defined in his user profile, are used to filter sight information.


Figure 7.6: Two sculptures that match Joey's sight group profile and located within a walking distance from his current location

By using the user's current location together with the user's interests, only the sights that the user is interested in and are located near to the user's current location are suggested. This will help the user not to be overwhelmed and irritated by the returning number of sights from the system. As a result, two nearby sculptures, *In Memory You are Known* and *Millennium Family* number (5) and (9) on the map (Figure 7.1), are recommended as shown in Figure 7.6.

Joey notices that the number of recommended sights has been considerably decreased compared to a given recommendations from the first option. Moreover, all these nearby sights match his preferences. He is satisfied with the results returned from the two recommendation options that he used for testing. Joey finally decides to use the TIP system. He then revisits the main screen. From this point, he can give his impression of this sight as feedback to the system or he can ask for more recommendations from the system. Figure 7.3 (a) shows the user feedback screen where the user can give their impression to their current visit sight. Feedback scores range from 1 to 10 and can be selected from a drop down list. Since Joey is very impressed by the sculpture,

he selects 10 as his feedback score.

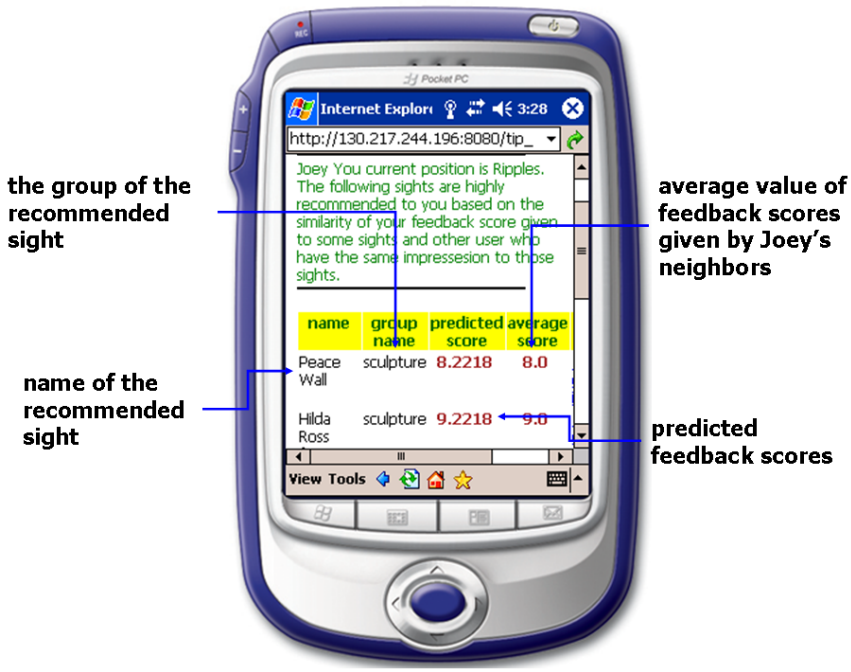



Figure 7.7: A set of sights generated by the extended collaborative filtering using implicit feedback from Joey's sight group profile

Typically, the collaborative filtering and the content-based recommendation options are not applicable to a new user since the user has not yet given enough feedback scores to be used in the calculation algorithms. However, the implementation of the extended *collaborative filtering using information in the user profile* (option 4-2) and extended *content-based using information in the user profile* (option 5-2) allow Joey to be able to get recommended sights from other like-minded users.

To generate the recommendations created by option 4-2 shown in Figure 7.7, Joey's missing feedback scores used to find his neighbours are implicitly extracted from his interests stored in his sight group profile. This technique effectively fixes the *cold start* problem for the collaborative filtering algorithm as well as decreases the user attempts to give their feedback in order to get the recommendations. For each sight in the given recommendations, a predicted feedback score together with an average score is described. This aims to provide a reason to the user why these sights are recommended.

The recommendations created by option 5-2 are different since the information stored in the user sight groups profile is not used to replace the missing feedback scores. Rather, this information is exploited to find recommendations from other users who share the same interests in the sight groups with Joey. This technique is based on two ideas. One is to recommend the sights in the sight groups that other like-minded users (neighbours) are interested in even though these sight groups are not defined in the user's sight group profile. The idea is to lessen the *over-specialization* issue which is the major drawback of the content-based recommendation. Herlocker et al. [13] referred to


Figure 7.8: A set of sights generated by the extended content-based algorithm using similarity between Joey's and other users' sight group profiles

this type of recommendation as *serendipitious recommendation*. They claimed that a serendipitious recommendation helps the user find surprisingly interesting items they might not have otherwise discovered. The other is to solve the lack of feedback scores given to sights in the sight groups of interests for the new user. The recommendations generated by this option are illustrated by Figure 7.8. Sights in the groups that these similar users are also interested in are recommended to Joey. In this case, the sight group *Stained glasses window* is of interest. Therefore, the sights *Te Deum East Window, leadlight windows, and Glass-Panel-1* are recommended. The recommendations include the predicted score of each sight so that Joey would understand why these three sights are suggested to him. In Joey's application scenario, we have confirmed that our extended recommendation models can lower the problem of not having enough feedback scores for the new user required for both the collaborative filtering the content-based recommendation. Moreover, the problem of having excessively restricted content-based recommendation is also decreased.

Hereafter, we will examine if the implemented recommendation models also satisfy Anne, our experienced user. Anne is now at Hamilton Gardens. Since she is interested in arts, when she sees the *Hamilton Gardens Mural (11)*, she wants to get some information about this sight. She sends here current location to the system and gets her required information. After that, Anne decides to get recommendations from TIP. Anne would like to visit the sights that are similar to some sights she had a good impression of in her past visits so she selects option 5-1 (see Figure 7.4(b)), the *pure content-based recommendation*. The system takes the information in her sight groups profile,


(a) Recommended sights

(b) Past visited sight and the feedback scores given by Anne

Figure 7.9: Recommendation generated by the *pure content-based recommendation model* given to Anne

her travel history, and her feedback scores to filter out the dissimilar sights. The returned recommendations consist of a set of the recommended sights shown on the upper part of the screen (see Figure 7.9(a)) and the sights from her past visits together with her feedback scores given to these sight shown on the lower part (see Figure 7.9(b)).

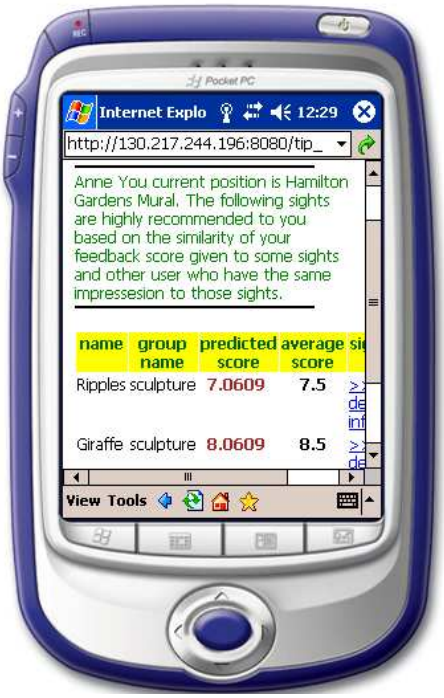
After reading through the recommendations, she is interested in seing some painting. She clicks the link to get information about the location of these painting. She finds out that they are in the Hamilton Central Library (e.g., number (16), (17), (18) as shown in Figure 7.1). The next place that Anne will visit is the Waikato museum so she decides to stop by the central library after that.

If Anne would like to get recommendations from other users who have similar travel history and/or preferences in the sight groups, she can select other content-based recommendation options (option 5-2 to 5-4). Figure 7.10 describes the recommendations generated by other users who have similar interests (see Figure 7.10(a)) and the recommendation generated by other users who have both similar interests and travel histories (see Figure 7.10(b)). Anne is satisfied with the given recommendations which are given by other users who have similar interests in the sight group. She finds it is interesting not to get recommendations only on the sights that she defined as her interests. The recommendations given by other users who have similar interests and travel histories are interesting enough for her to follow.


(a) Recommendations from other users who have similar preferences


(b) Recommendations from other users who have similar preferences and travel history

Figure 7.10: Recommendation generated by the *extended content-based recommendation model* given to Anne




(a) The pure collaborative filtering


(b) The extended collaborative filtering using information in the sight group profile

Figure 7.11: Anne's collaborative filtering recommendations

Since Anne has given her feedback to most of the sights she visited, she thinks it might be worth taking recommendations from others who have similar opinions about these sights. Thus, Anne selects the *pure collaborative filtering* option (4-1) in the recommendation menu. Figure 7.11(a) illustrates the results of this recommendation option. Two sculptures *Ripples* and *Giraffe* are given. Although Anne did not define her interest in sight group sculpture, she is interested in the given recommendations. After she looks through the detailed information about the sculptures, the takes the *Ripples* sculpture as another stop when she visits the Waikato museum.

As the experienced user, Anne is able to select other available recommendation options (see Figure 7.4), e.g., the *knowledge-based recommendation* (option 6), the *extended collaborative filtering* (option 4-2 to 4-4). Figure 7.11(b) describes the results of the extended collaborative filtering using implicit feedback from Anne's sight group profile to replace her missing feedback scores.


(a) Recommended sights (b) Past visit sights in the sight groups

Figure 7.12: Knowledge-based recommendation generated based on information in Anne's travel history

The knowledge about Anne's past visit together with information about her interests is used to generate the *knowledge-based recommendations* as shown in Figure 7.12. The given recommendations illustrated by Figure 7.12(a) are generated based on the information in her travel histories. Since Anne already visited two of the paintings and two murals, others paintings as well as other murals are suggested. The recommendations include name of the sights and sight groups Anne visited as shown in Figure 7.12(b). The last recommendation option is to recommend popular sights which are highly scored by most of the users as expressed by Figure 7.13. For each recommended sight,the total

number of scores and the total number of high scores ($\geq 7$) given by the users is presented.


Figure 7.13: Highly scored sights

From our exploratory study based on the example scenario mentioned above, we have found that the recommendation algorithms created to be employed in the recommendation component have efficiently generated recommendations for both the new and the experienced user of the TIP system. In other words, a combination of the parameters for generating recommendations (e.g., user sight group profile, user context, user travel history and user feedback score) and the recommendation methodologies can effectively lessen the restrictions found in each of the pure recommendation paradigms (e.g., lacking of the feedback scores of a new user, excessively restricted recommendations).

## 7.3 Analysis and Summary

We validated the effectiveness of our implementation in this chapter. The evaluation was conducted using the currently employed data set of the TIP system. We illustrated the evaluation results through the two application scenarios: the new and the experienced user. For clarity, screen shots captured from a personal computer were used instead of the actual mobile devices. In this section, we analyze the outcomes of our evaluation in two aspects, the *effectiveness* of the implemented recommendation algorithms and the *fluidity* of the user's interactions.

### 7.3.1 Effectiveness of the Recommendation Component

As we mentioned in Chapter 6, the main goal of the qualitative evaluation is to confirm that our implementation is able to address these four limitations: *new user problem*, *cold start problem*, *specific preferences user problem* or *Grey Sheep problem*, and *over-specialize problem*. To be more specific, we consider the *effectiveness* of our implementation as *an ability to provide recommendations to the user by using other types of the user information (e.g.,* user preferences *and* user travel histories*) to replace the missing information required for the recommendation generating procedures*. The outcomes of the implementation explained in the previous section indicate positive implications as follow.

1. The problem of not being able to provide recommendations to the *new user* due to a shortage of the user's feedback scores for the collaborative filtering algorithms or the so called *cold-start problem* is rectified. Using information about the user's interests to fill in the missing feedback allows the user to get recommendations from other similar users. As we can see from the new user's test scenario (refer to page 79 and Figure 7.7), by using information in Joey's sight group profile the system can generate recommendations using the collaborative filtering algorithm although he just starts to use the system and gives only one feedback score to the sight. Meanwhile, the user who has been using the system but is not keen to give their opinion can also get this advantage since the information in their travel histories can be used as their implicit opinions about the sights.

2. Since the users are not strongly encouraged to provide their feedback scores in order to get recommendations from the system, the problem of inaccurate recommendations given to a user who has inconsistent opinions with any group of users or the *Grey sheep* user can be lessened. However, we regard this benefit as a short term remedy. Langley et al. [3] suggested a solution for the grey sheep problem which was to use a user model to represent a group of users based on their similar tastes. A simplistic method starts with gathering users who are highly correlated. Then each of the users is grouped into a neighbourhood. From this, an average or model of users for each of these groups is built. The *unusual* opinion of the *grey sheep* user will be drowned out in favour of the more usual opinions of members of the neighbourhood. Accordingly, we no longer concentrate on similarity between the individual and other users in the system. Rather, we choose a user who represents the center of the tastes of the group as the active user to whom we generate recommendations. This assumption is appealing and will be implemented as a proof of concept in the next version of the recommendation component.

3. *Over-specialization*, which is the main drawback of the content-based recommendation model, is fixed by the idea of providing *serendipitious* recommendations. We proposed to use two other types of the *user-user* similarity, the *similarity on sight groups of interest*, and the *similarity on sight of interest* to extend the *content-based recommendation*. From

our test scenario, both Joey (refer to page 80 and Figure 7.8) and Anne (refer to page 81 and Figure 7.10(a)) get recommendations on the sight group *stained-glass windows* from other users who have similarity in the sight groups of interest although they both did not define their interests in this sight group. Moreover, as the experienced user Anne is able to select to get recommendations from other users who have similar travel histories (refer to page 81 and Figure 7.10(b)). We believe that this type of recommendation helps the users to find surprisingly interesting sights they miss from using the ordinary content-based recommendation algorithm.

To summarize, using the users' personal information to replace their missing feedback scores required for the recommendation generations gives appealing outcomes. The results of the implementation show that the *new user* and the *cold start* problems for the recommendation algorithm using the collaborative filtering paradigm are effectively rectified. We find out that the *grey sheep* problem, which is also a concern for the collaborative filtering algorithms, is more complicated. Although no requirements for giving feedback scores to the system in order to get recommendations can lessen the inconsistency of the feedback given by these users for now, we believe this problem still requires more tangible solutions in the long run. Accordingly, we adopt the idea proposed in [3] to be employed in the next revision of the recommendation component. Meanwhile, the idea of finding neighbourhoods based on other two types of the user-user similarity, the *similarity on sight groups of interest* and the *similarity on sight of interest* in order to generate *serendipitious* recommendations to the users can remedy the *over-specialization* problem of the recommendation algorithm using the content-based recommendation paradigm.

### 7.3.2 Fluidity of the Interactions

We consider the *fluidity of the interactions* between the user and the system as a key factor for the *transparency and user control* and the *user satisfaction* issues. As we explained in our goal of evaluation, we analyze the fluidity of the users' interaction to the system in two measures, the *acceptability* and *tractability*. The following paragraphs illustrate our first-cut analysis on the fluidity of interactions of the recommendation component.

1. *Acceptability*

   We believe that acceptability from the users depends on issues of initial personalization costs (e.g., for creating user profiles) and the quality of the recommendations themselves (we leave out the issues of privacy and trust in this study). In the case of the initial costs, avoiding a requirement for explicit user profiles is desirable. However, we need to carefully balance between enhancing ease of use (by not explicitly asking the users to define their interests) and enhancing the accuracy of the algorithm. The study conducted in [36] indicates that the users do not mind giving a little more input to the system in order to receive more accurate recommendations. According to this statement, we believe that the users are willing to provide information about their preferences if this will help

them to get useful recommendations from the system. However, we still need to conduct a study on how much information is appropriate for asking the users to provide as their input to the system.

In the case of the quality of the recommendation, user understanding of why a particular sight is recommended is a crucial issue for the acceptability of the system. In this prototype, we believe that providing clearly laid out recommendation options and giving a brief explanation of the selected recommendation algorithms (see Figure 7.5) reflect the quality of the recommendation to the users. This idea is supported by Swearinggen et al. [36]. Since the presence of longer descriptions of individual items correlates positively with both the perceived usefulness and ease of use of the recommender system, our recommendations comprise two types of information the user may find useful. One is the *basic information* which includes sight name and group name. The other is *community rating information* which consists of the predictions and the average score. A link for more information about a sight and its picture is also given.

2. Tractability

   *Tractability* is a long term functional test regarding user acceptance which leads to the issue of *trust* in a system's recommendations. The study conducted in [35] illustrates two factors which strongly affect levels of the users' trust, familiarity with the recommended items together with explanation and justifications for the recommendations. In this case, the initial costs for creating user profiles are rewarded since the users can understand why the sights are recommended and can clearly uncover a link between the input and their output. Accordingly, they perceive that the system understands their tastes an are inclined to trust the system more. The basic information about the sights (e.g., name, sight group, a link to more detailed information including picture) and the community rating also aid users in decision making to visit the sights.

In summary, analysis of the implementation results on the *effectiveness* of the recommendation component and the first-cut analysis on the *acceptability* and *tractability* of the recommendation component are appealing. We confirm the effective use of the users' personal information such as their user profiles and travel histories to generate the recommendations. We also verify the fluidity of the system interaction which leads to acceptability and tractability (user's trust to use the system in the long run). The next stage of the TIP project with regard to the qualitative evaluation is to conduct further system validation based on long term user studies.

# Chapter 8

# Quantitative Evaluation

In this chapter, we conduct the quantitative evaluation on the implementation of the recommendation component. Our focus is on the performance of the recommendation algorithms using the collaborative filtering paradigm. We begin this chapter with the experimental setting in which we illustrate the data sets, the employed methodologies, and the experimental platform. The experiments concentrate on three subjects, *complexity*, *response time* and *coverage* of the recommendation algorithms. We first analyze the complexity of the algorithms. Then we illustrate the experimental procedures, results and analysis of the results for the response time and coverage.

## 8.1 Experimental Setting

In this discussion on the experimental setting, we first introduce a typical data set generated for our experiments. Then, we illustrate the employed methodologies which we divide into two categories based on the outline stated in our evaluation plan described in Chapter 6: the *response time* and the *coverage*. Finally, we explain the experimental platform.

### 8.1.1 Data Set

Herlocker et al. [13] suggested that when evaluating a recommendation in a new domain where there is significant research on the structure of user preferences, but no data sets, it may be appropriate to evaluate algorithms against synthetic data sets to identify the promising ones for further study. Based on this statement, we propose to conduct our experiment using synthesized data.

We design the recommendation service in the TIP system based on two assumptions. First, there are many more users than items in the tourist information system domain. Second, the system will be utilized in a particular area, for instance, downtown Hamilton, the Waikato museum etc. Accordingly, a typical test data set used in our experiment consists of 1,000 users and 100 sights. We divide the test data sets into a training and a test portion. For this purpose, we introduce a variable $p$ used to determine which percentage of the data is used as a training and a test portion. A value of $p = 0.7$ indicates 70%

of the data is used as a training portion and 30% of the data is a test portion. Hereafter, we carry out every experiment on the test data set with the variable $p = 0.8$. The data set is converted into a *user-item* matrix $F$ that has $m$ rows (e.g., users) and $n$ columns (e.g., sights). Each of the sights is rated by at least one of the users. The number of sight groups is fixed at 10 whereas each of the sights is randomly assigned to one single group. The number of sight groups allocated in a user profile for each user is fixed at 5. These 5 sight groups are randomly assigned to the user's profile. The number of sights stored in the user travel history is generated as 20% more than the number of user feedback scores based on the assumption that all of the sights that the user has rated are counted as their travel history. Furthermore, the users may not give their feedback to every sight they have visited.

For our experiments, we also take another factor into consideration, *sparsity level* of data sets. We adopt a determination of the sparsity level of the data set proposed in [33] throughout our experiments. Consequently, the sparsity level for the data matrix $F$ is $1 - \frac{nonzero\,entries}{total\,entries}$. We set the sparsity level of our test data sets at 90%.

### 8.1.2 Methodologies

A number of different measures have been proposed and utilized to evaluate the performance of the various filtering algorithms employed by recommendation systems [37]. Performance metrics related to time and storage requirements can be divided into three categories, *response time*, *storage requirement*, and *complexity*. In this experimental evaluation, we concentrate mainly on the *response time* which is related to the *complexity* of the implemented recommendation algorithms. Meanwhile, we consider *coverage* as an indicator to performance of the prediction mechanism of the recommendation algorithms.

#### 8.1.2.1 Response Time

*Response time* is a widely used performance measurement, which is utilized for various purposes and in different domains. In the case of recommendation systems, it is defined as the time that elapses between a user's stated request and the system' response to that request [37]. In our test scenario, the users submit their request to the system at time $t_1$. The system accepts the request, processes the input and after a successful completion of the required tasks, it will provide a response at time $t_2$, where $t_2 > t_1$. Then, the response time $t_r$, is defined as the difference between $t_2$ and $t_1$. As a result, $t_r = t_2 - t_1$.

#### 8.1.2.2 Coverage

*Coverage* is a measure of the percentage of items for which a filtering algorithm can provide predictions [37]. A low coverage value indicates that the employed recommendation algorithm is not able to assist the user if they have not yet rated many of the items. On the contrary, a high coverage value indicates that the recommendation algorithm is able to provide adequate help in the selection of the item the user is expected to enjoy more. In our test setting,

we define coverage as *the number of sights for which predictions can be formed as a percentage of the total number of sights the active user has not yet rated.* Equation 8.1 illustrates the equation used to calculate the coverage value

$$Coverage = \frac{\sum_{i=1}^{m} np_i}{\sum_{i=1}^{m} n_i} \tag{8.1}$$

where

$n_i$ is the total number of sights for which user $u_i$ has not yet given feedback scores,

$np_i$ is the number of those sights for which the recommendation algorithm is able to generate a prediction, with $np_i \leq n_i$,

$m$ is the total number of the test users.

### 8.1.3 Experimental Platform

We run all of our experiments on a Windows based PC with AMD Athlon XP 2700+ processor having a speed of 2.16 GHz and 512 MB of RAM.

## 8.2 Experiments

In this section, we explain the experiments, their results and analysis of the results. The experiments focus on four collaborative filtering recommendation algorithms using the *neighbourhood-based* method since these algorithms grow significantly with the number of *users* and *items* in the database. First, we theoretically analyze complexity of each of these recommendation algorithms. Then, we examine the response time required by the algorithms in order to confirm the results of the analysis. Finally, we compute coverage values provided by the algorithms in order to illustrate performance of their prediction mechanism.

### 8.2.1 Complexity

Vozalis et al. [37] claimed that some existing recommendation algorithms offer theoretically promising mathematical techniques in order to generate their results but nonetheless require complex calculations. Therefore, it is vital to evaluate the complexity of the designed algorithms. This is to prevent the implementation from using an expensive algorithm which causes an unacceptable delay time in responding to the users' requests. In this section, we discuss the complexity of the recommendation algorithms using the collaborative filtering paradigm which we have employed in the recommendation component.

The analysis concentrates on the *user-user* similarity calculation step since it turns out to be a performance bottleneck. We initially present the worst case computation and then give an approximation of the algorithms under the real conditions.

A scenario of a typical collaborative filtering algorithm is that $m$ is the number of users and $n$ is the number of sights in the database. The computation complexity of the *user-user* similarities is $O(m^2n)$, since we need to compute the similarity between each pair of users according to the subset of their co-rated sights.

Considering a scenario of the extended collaborative filtering algorithms in our implementation, another two variables used in the computation need to be taken into account. One is $k$, the number of sight groups provided by the system which the users can define as their interests. The other is $h$, the number of sights the user has visited, which is recognized by the system as their travel history. The computational complexity of the extended collaborative filtering algorithm using the *user sight group profile* is $O(m^2n^k)$ as we need to verify whether a sight with *no-feedback* score belongs to any of the sight groups stored in the user profile. Meanwhile, the complexity of the extended collaborative filtering algorithm using *user travel history* is $O(m^2n^h)$ since we need to compare a no-feedback sight with the sights stored in the user travel history. The complexity of the latter algorithm is greater because we first retrieve sight information from the database. Then, for each sight, we count the number of visits to the sight given by the users.

In general, the number of sight groups $k$ is rather static and much less than the number of sights in the database, ($k \ll n$). Although the number of sights visited by the user $h$ is increased as the user uses the system more, it is also much less than the number of sights stored in the system, even for enthusiastic user. Based on these two reasons, we conclude that the complexities of the algorithms we have implemented in this study are highly related to the increasing number of users and sights in the database. In other words, these four algorithms also have an $O(m^2n)$ complexity.

To confirm this analysis, in the next section we carry out the experiments on the complexities of these four recommendation algorithms regarding the system response time.

### 8.2.2 Response Time

The experimental scenario of the system response time is set up to practically depict the level of complexity that the implemented algorithms demonstrate when the active user requests a recommendation. Our main focus is on the impact of the increasing number of the two major variables, the *users* and the *sights*.

#### 8.2.2.1 Experimental Procedure

Our experimental procedure employs the data set described in the experimental setting section. Here, we carry out the experiments in the real life situations where it is typical for the recommendation system to encounter a high sparsity

level of the *user-sight* matrix. Thus, we define the experimental variables as follows:

- $m'$ is the number of users with whom the active user has at least one co-rated sight, with $m' \ll m$;
- $n'$ is the number of feedback given to sights by the active users, with $n' \ll n$;
- $k'$ is the number of sight groups defined in the user profile of the active user and other users, with $k' \leq k$;
- $h'$ is the number of sight visited by the active user, with $h' \geq n'$ since the user may not give their feedback to every sight they have visited.

Hereafter, we respectively refer to these four variables as, the *co-rated users*, *feedback sights*, *sight groups* and *visited sights*.

We follow the criteria setting for a selection of the active user's neighbourhood and a recommended sight described in Chapter 4, Section 4.2.1.2 and Section 4.2.1.3. A user is considered to be a neighbour of the active user if their associated similarity factor is greater than or equal to 0.7 (in a range of -1 to 1) whereas a sight is recommended to the user if it holds a predicted feedback score greater than or equal to 7. To experimentally determine the impact of these two variables on the response time, we selectively vary their values, as summarized in Table 8.1.

Table 8.1: Variable setting for the experiment on the system response time

| **Description** | **Variable** | **Varied Range** |
|---|---|---|
| Experiment 1 | number of co-rated users $m'$ | 100 to 500 in an increment of 100 |
| Experiment 2 | number of feedback sights $n'$ | 10 to 50 in an increment of 10 |
| Experiment 3 | - number of sight groups $k'$<br>- number of feedback sights $n'$ | 10 to 40 in an increment of 10<br>10 to 50 in an increment of 10 |

As shown in the above table, experiment 3 is concerned with changing two variables, the sight groups ($k'$) and feedback sights ($n'$). In this case, we choose to carry out one experiment on the influence of the changing number of the sight groups and the visited sights. This is to confirm our assumptions on the complexity of the implemented recommendation algorithms where the number of the sight groups and the number of the visited sights have low influence compared to the number of the co-rated users and the feedback sights. The subsequent sections illustrate our experiments, their results and analysis of the results.

#### 8.2.2.2 Results and Analysis

1. *Experiment 1: Changing of the number of co-rated users* ($m'$)

   In the experiment, we examine the influences on the system response time of the increasing number of users with whom the active users have at least one co-rated sight ($m'$). We present the results in Figure 8.1. It can be observed from the chart that the averged response time of these four algorithms is linearly elevated in accordance with the increasing number of the co-rated users. It is obvious that the four algorithms's behaviour can be separated into two sets. The response time of the *extended collaborative filtering using user profile* is very close to the *collaborative filtering using the default value*. Meanwhile, the response time required by the *extended collaborative filtering using user travel history* and the *extended collaborative filtering using user profile and user travel history* is much higher and it is considerably increased when the number of the co-rated users is increased. This is because these two algorithms require more access to the database since they both retrieve the visited sights information and count the number of visits of each user in the *user-sight* matrix $F$. The results confirm the theoretical analysis of the complexity of the algorithms carried out in the previous section. The response time is increased in accordance with the increasing number of the co-rated users and the complexity is $O(m^2)$.

2. *Experiment 2: Changing of the number of feedback sights* ($n'$)

   The second experiment is concerned with the influences on the system response time of changing the number of feedback sights ($n'$) . It can be seen from Figure 8.2 that the averaged response time of these four algorithms is elevated when the number of co-rated sights increases. The chart also shows that the rate of increase is less than the first experiment. It is likely that we can also separate these four algorithms into two groups according to the rate of increase of their response time (similar to the test results of the first experiment). However, it is not as obvious as the results shown in Figure 8.1 since the complexity of the first experiment is $O(m^2)$ while in this experiment the complexity is $O(n)$. The results also confirm the theoretical analysis on influences of the increasing number of sights on the complexities of these algorithms.

3. *Experiment 3: Changing of the number of sight groups* ($k'$)

   In the third experiment, we focus on the influences of the increasing number of sight groups ($k'$) defined in the active user profile and the feedback sights ($n'$). We therefore focus mainly on the following recommendation algorithms:

   - the *extended collaborative filtering using user profile*;
   - the *extended collaborative filtering using user profile and user travel history.*

   Figure 8.3 and Figure 8.4 show the results of the experiment on the *extended collaborative filtering using user profile* and the *extended col-*

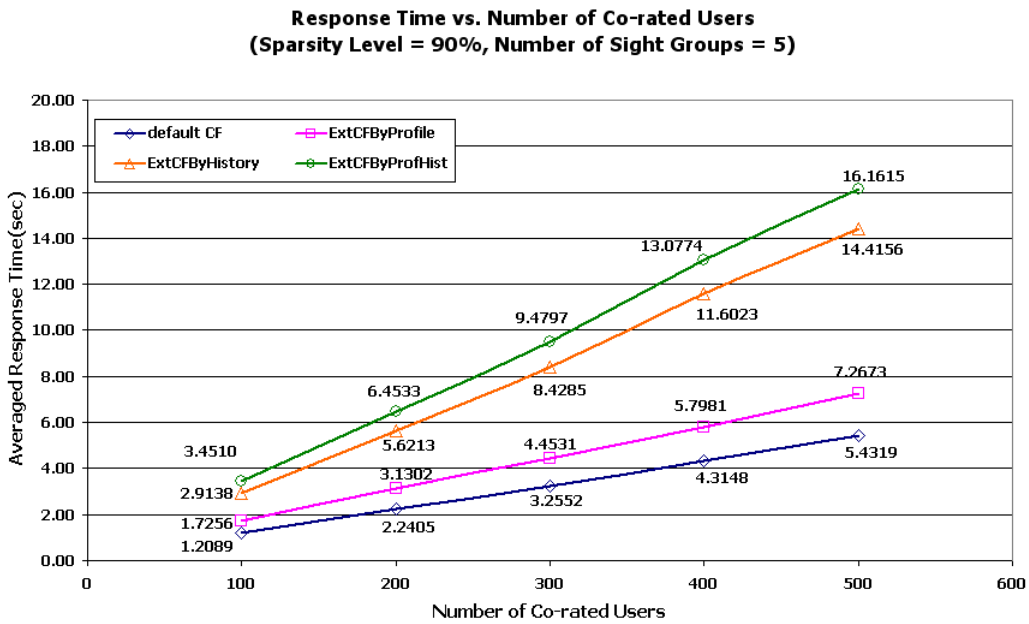



Figure 8.1: Response time vs. increasing number of users with whom the active user has at least one co-rated sight

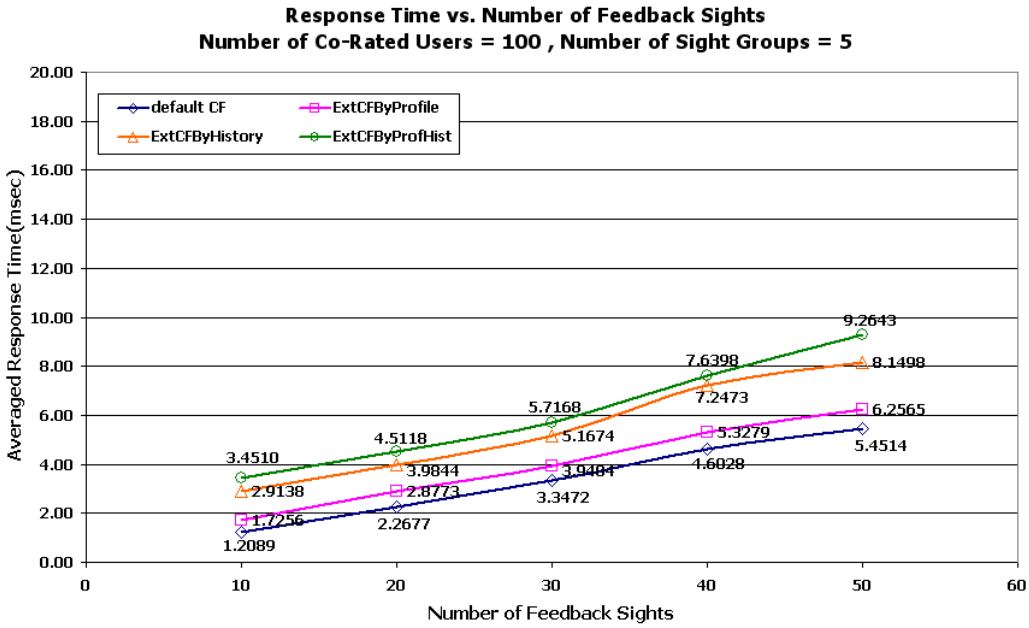



Figure 8.2: Response time vs. increasing number of feedback sights of the active user and another users

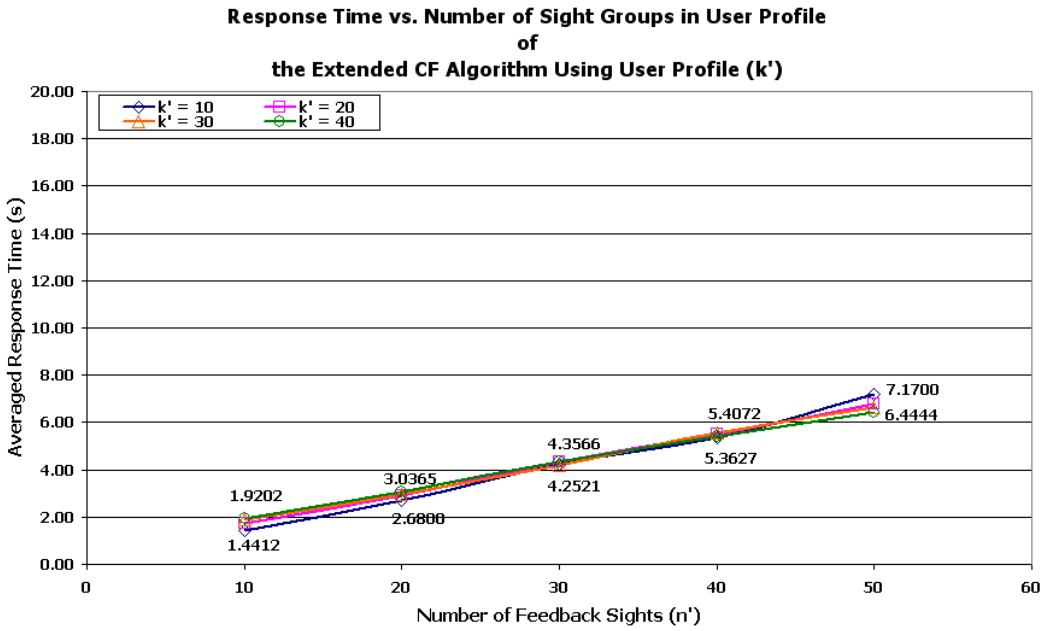



Figure 8.3: Response time vs. increasing number of sight groups stored in the active user's profile (Extend CF using user profile)

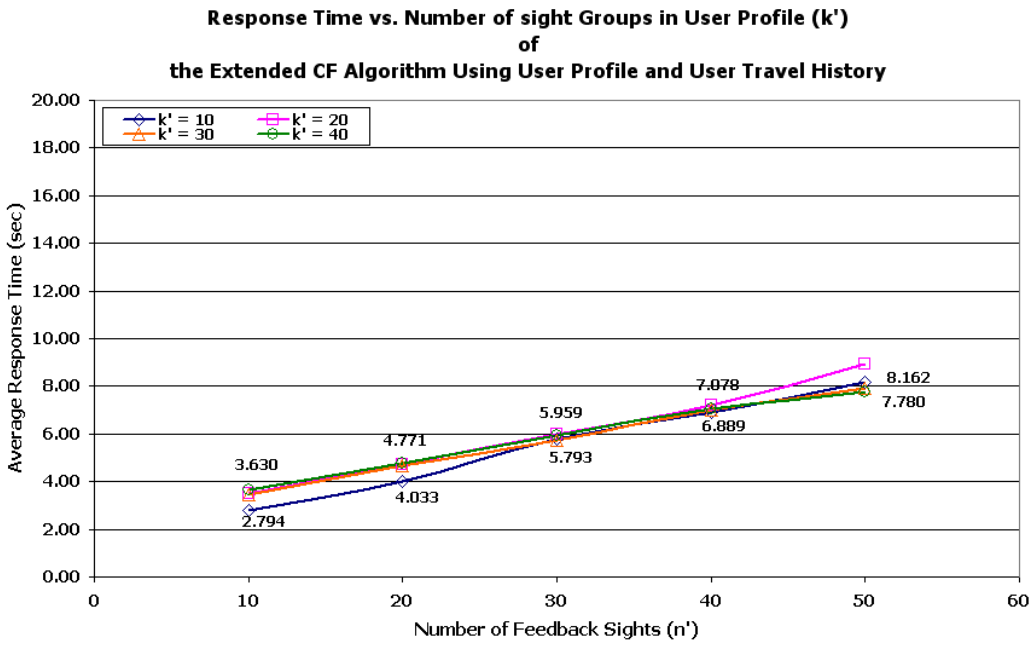



Figure 8.4: Response time vs. increasing number of sight groups stored in the active user's profile (Extend CF using user profile and travel history)

*laborative filtering using user profile and user travel history* respectively. It can be observed from the charts that the increasing number of sight groups $k'$ has less impact on the growth of the system response time than the increasing number of feedback sights $n'$. The numbers shown in the charts indicate response time with the lowest ($k' = 10$) and the highest ($k' = 40$) setting of the number of sight groups. Comparing the response time of the *extended collaborative filtering using user profile* shown in Figure 8.2 (*e.g.*, $k' = 5\, and\, n' = 50$) and Figure 8.3 (e.g., k'= 40 and n' = 50), the response time $t_r$ is $t_r \approx 6.25s$ and $t_r \approx 6.44$ respectively. The response time of the *extended collaborative filtering using user profile and user travel history* algorithm is somewhat different where the response time shown in Figure 8.2 (*e.g.*, $k' = 5\, and\, n' = 50$) and Figure 8.4 (e.g., k'= 40 and n' = 50), $t_r$ is $t_r \approx 7.78s$ and $t_r \approx 9.26$ respectively.

#### 8.2.2.3 Conclusion

We draw conclusions from these experiments as follow:

- The increasing number of users and sights is highly influential on the computational complexity of the recommendation algorithms using the collaborative filtering paradigm. The results of the first two experiments confirm the theoretical analysis on the implemented algorithms. In experiment 1, the response time of these four algorithms is increased in accordance with the increasing number of co-rated users or the complexity is $O(m^2)$. Meanwhile, the results of experiment 2 show that the increasing number of feedback sights requires $O(n)$ response time.

- Considering the increasing number of users and sights in the *user-user* similarity computation step, the recommendation algorithms using the default value and the user profile to replace the *no-feedback* values in the matrix $F$ reveal less complexity than using the user travel history (as shown in Figure 8.1 and Figure 8.2).

- The results of experiment 3 show that the increasing number of feedback sights overshadows the computational complexity of the extended recommendation algorithms using user profile. Although we focus more on the increasing number of users and sights in our experiment, it is essential to point out that more extensive experiments on each of these four variables are required. These experiments may be conducted as pilot experiments before the system employment (using synthetic test data sets or using natural data to construct an actual user model) when the system is completed.

### 8.2.3 Coverage

We examine *coverage* values generated by each of the algorithms in order to verify the system's performance with regard to a prediction mechanism of the recommendation algorithms. In other words, we concentrate on the algorithm's ability to provide predictions over the number of un-rated sights of the active user. We made a general assumption here:

*employing the users' personal information (e.g., user profile, travel history) to replace a missing feedback score required by the collaborative filtering algorithms is superior to using default or null values.*

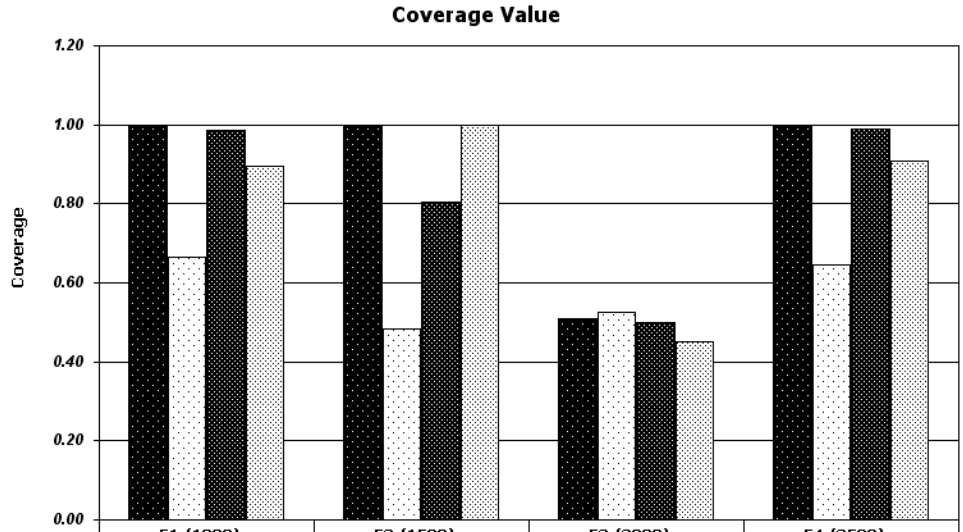
Typically, this is not necessarily true. In fact, we define this assumption in order to demonstrate the behaviour of the extended collaborative filtering using personal information from the user compared to the default value. The experimental scenarios are set up to depict the ability to provide predictions over unrated sights of these four algorithms when the active user requests recommendations.

#### 8.2.3.1 Experimental Procedure

As the design of the extended collaborative filtering primarily aims to solve the problem of the missing feedback scores, we therefore concentrate more on the system's ability to give recommendations when the user-sight matrix is very sparse. Therefore, we examine coverage values generated when the sparsity level of the matrix $F$ is $\approx 93\%$. The experimental data set comprises 3000 users and 100 sights. The number of sight groups allocated in a user profile for each user is randomly generated and varied from 1 to 10 sight groups. Each of the users provide their feedback scores to at least 1 sight. We apply the same setting for the number of visited sights stored in the users' travel histories (20% more than the number of the user's feedback scores). We employ the same criteria used to select the active user's neighbourhoods ($similarityfactor \geq 0.7$) and a recommended sight ($prediction \geq 7$). To experimentally determine the coverage value of the algorithms, we randomly select the 4 test data sets. The number of all users in the data set is varied from 1000 to 2500 in an increment of 500. Here, we concentrate on two subjects, coverage value and percentage of the users who can get recommendations from the algorithms.

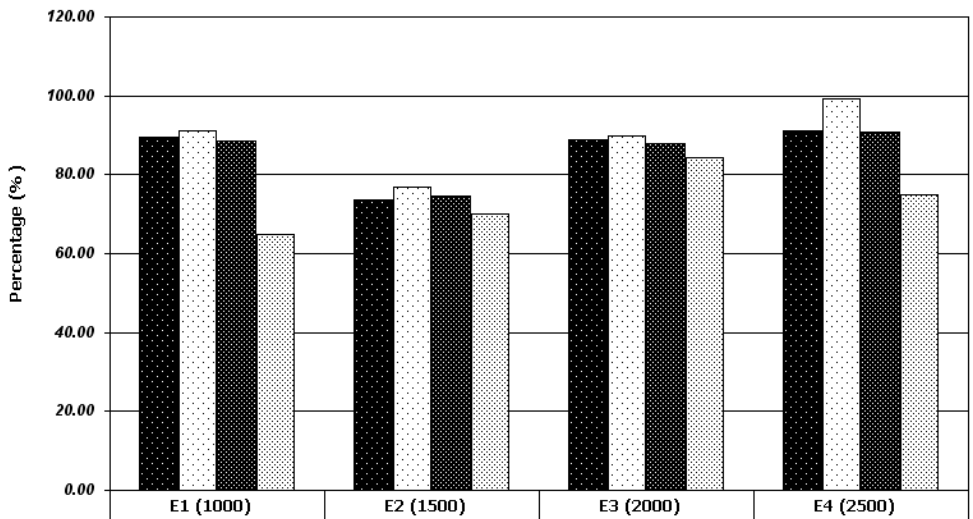
#### 8.2.3.2 Results and Analysis

In this section, we present our experimental results. Figure 8.5 illustrates the coverage values of the four recommendation algorithms. It can be seen from the chart that among these four algorithms the coverage value of the extended algorithm using the user profile is the lowest in every test data set ($\approx 0.58$ on average). Meanwhile the extended algorithms using default valued ($\eta = 5$) has the highest coverage value ($\approx 0.88$ on average). Coverage values provided by the extended algorithm using travel history and a combination of user profile and travel history reveal similar coverage values ($\approx 0.82$ on average). The results also show one significant aspect: the number of test users is not as important as the quality of their information stored in the data set. The results of the experiment using the third test data set of 2,000 users show the lowest coverage value for all of the four algorithms. Meanwhile, the first (1,000 users) and the fourth (2,500 users) experiments reveal similar characteristics whereas the results of the second experiment (1,500 users) are slightly different. This is because the third data set has the lowest distribution of the sights that have




| | E1 (1000) | E2 (1500) | E3 (2000) | E4 (2500) |
|---|---|---|---|---|
| DefaultCF | 0.9985 | 1.0000 | 0.5108 | 0.9988 |
| ExtCFByProfile | 0.6652 | 0.4833 | 0.5238 | 0.6441 |
| ExtCFByHistory | 0.9862 | 0.8057 | 0.4990 | 0.9904 |
| ExtCFByProfHist | 0.8941 | 1.0000 | 0.4497 | 0.9067 |

Figure 8.5: Coverage values of the four recommendation algorithms




| | E1 (1000) | E2 (1500) | E3 (2000) | E4 (2500) |
|---|---|---|---|---|
| DefaultCF | 89.50 | 73.50 | 88.75 | 91.20 |
| ExtCFByProfile | 91.00 | 77.00 | 90.00 | 99.17 |
| ExtCFByHistory | 88.50 | 74.50 | 87.75 | 90.80 |
| ExtCFByProfHist | 65.00 | 70.00 | 84.20 | 75.00 |

Figure 8.6: Percentage of the users who get recommendations from the algorithms

been rated by the users. In other words, the number the users who have rated some particular sights are very high whereas some are very low compared to another three data sets. This leads to a condition where some users have a very high number of neighbours and sights to use in computed predictions but the number of the generated predictions is low.

Figure 8.6 describes the percentage of users who get recommendations from the system. It can be observed from the chart that the percentage of users who get recommendations form the system is high ($65\% - 99\%$). Among these four algorithms, the extended algorithm using the user profile shows the highest percentage in every data set ($91\% - 99\%$). Meanwhile, the extended algorithm using a combination of user profile and travel history gives the lowest value in every data set ($65\% - 75\%$). The extended collaborative filtering algorithm using the default value and the travel history shows a similar percentage value and give the highest value of $\approx 90\%$.

#### 8.2.3.3 Conclusion

We draw conclusions from these experiments as follow:

- Using the default value to replace the missing feedback scores reveals the best performance since it provides both a high coverage value ($\approx 0.87$) and the percentage of users who get recommendations ($\approx 85.7\%$);
- The extended collaborative filtering algorithm using the user profile shows the highest percentage of users who get recommendations ($\approx 99\%$) but fails to provide a high coverage value. It reveals the coverage value of $\approx 0.58$ on average, which is the lowest among these four algorithms.
- The extended algorithms using user travel history and a combination of user profile and user travel history shows a similar coverage value ($\approx 0.58$) but the percentage of users who get recommendation provided by the former algorithm is higher ($\approx 12\%$).

## 8.3 Summary

In this chapter, we carried out the quantitative evaluation of the implementation of the recommendation component. The main focus was on performance of the recommendation algorithms using the collaborative filtering paradigm. We therefore concentrated on the computation complexity of the *user-user* similarity calculation since it turns out to be the bottleneck of the entire process. We began with the analysis of the approximate complexities of each of the algorithms. We concluded that these algorithms require an $O(m^2n)$ complexity for $m$ users and $n$ sights. Then we conducted the experiments on the *response time* of each of the algorithms according to changing of the number of users and sights in the system. We also carried out experiments on the coverage where we focused on coverage value and percentage of the users who get recommendations from the algorithms. The subsequent paragraphs explain our conclusions.

1. The results of the experiment regarding the system's response time confirm our theoretical analysis. The complexity of the four recommendation algorithms is an $O(m^2)$ for $m$ users and an $O(n)$ for $n$ sights.

2. The results of the experiments show that using the default value to replace the missing feedback scores produces the best performance since it requires the lowest complexity and provides the highest coverage value. Furthermore, the percentage of the users who get recommendations is also high.

3. The extended collaborative filtering algorithms using user profile fails to give high coverage values but shows the highest percentage of users who get recommendations. Meanwhile, its complexity is similar to using the default value.

4. Although the extended algorithms using user travel history produces a high coverage value and percentage of users who get recommendations, it requires high computational complexity. This is because these algorithms requires more access to the database than other algorithms.

5. The extended algorithm using a combination of the user profile and user travel history also provides high performance regarding the coverage issue. It fails to provide a quick response to the user's request since it shows the highest computational complexity.

# Chapter 9

# Conclusion

In this chapter, we summarize what we have accomplished in this project. We first review our aim for the project and our achievements. We then explain the problems we have encountered. Finally, we suggest possible future work.

## 9.1 Summary

As detailed in Chapter 1, our goal for the implementation of the recommendation component in the TIP system was to acquire recommendation algorithms which will help the users find sights to visit without being overwhelmed by the results returned by the system. The outcomes from the first prototype implemented in TIP 2.0 indicated seven important limitations. We classified these limitations into three categories:

- **Applied recommendation algorithms issue**

  These limitations are triggered by the existing drawbacks of the recommendation algorithms adopted and employed in our implementation. The recommendation algorithms using the collaborative filtering paradigm suffer from lack of initial information for building a user model of a new user, the so called the *new user* problem. This limitation then stimulates the *cold start problem* where the coverage of the feedback scores, highly required for generating recommendations, is very sparse. Consequently, the system cannot provide recommendations or generates poor recommendations for the users. Another problem is concerned with the inconsistent opinions given by the *grey sheep* users. These opinions are sometimes relatively unusual compared to other users in the system leading to poor or inaccurate recommendations. The recommendation algorithms using the contented-based recommendation paradigm encounter the *over-specialization* problem where the system can only recommend sights similar to those highly rated by the user. The recommendations are too restricted and may not match user requirements.

- **User acceptance issues**

  User acceptance is concerned with two limitations, *transparency and user control* and *user satisfaction*. These two issues are highly related to the

interface and interaction model employed by the system. Thus, it is important to examine the ability of the system to provide reasoning, the perceived usefulness and ease of use to the users.

- **System performance issue**

  As some recommendation algorithms offer theoretically promising mathematical techniques to generate their results but nonetheless require complex calculations, it is important to verify their computational complexities. In the tourist information system domain, one focus is on the increasing number of users who register with and use the system as well as the number of sights stored in the system.

To address the above limitations, we partitioned our tasks in this project into three stages: *design, implementation* and *evaluation* of the recommendation component.

### 9.1.1 Design

We introduced personalized recommendations utilizing the user's personal information (e.g., preferences, travel histories) in order to remedy the existing limitations triggered by the employed recommendations paradigms. Our main focus was:

> *to find appropriate recommendation approaches which can compliment the existing advantages and balance the current drawbacks of providing recommendations in a mobile tourist information system.*

Based on the background knowledge and the analysis of five system parameters used to generate recommendations (e.g., user profile, context of a user, context of a sight, user travel history, and user feedback) described in Chapter 2, we proposed the outline of some advanced recommendation models in Chapter 3. Here, we combined these five parameters and the general principle of the three recommendation paradigms (collaborative filtering, content-based recommendation, and knowledge-based recommendation) to generate our recommendation models.

In Chapter 4, we explained the design of the recommendation algorithms based on the outlines we created. We started by defining terms used in the filter algorithms. We divided the design of the algorithms into three categories based on their base recommendation paradigm. We then introduced equations for and functionalities of these filter algorithms. For each of the algorithms, we adopted the basic computing equations and modified the parameters used in these equations according to our proposed idea.

### 9.1.2 Implementation

The implementation described in Chapter 5 focused mainly on the employment of the designed algorithms in the recommendation component. As TIP is implemented using a client-server approach with a central database, the user's

personal information (e.g., preferences, feedback, travel histories) is stored in the database and a recommendation component is embedded as another type of filter engine on the server side. The implementation architecture of the recommendation component consists of seven elements: the *user interface*, the *recommendation manager*, the *data collector*, the *data storage*, the *location engine*, the *filter engine*, and the *notifier*. The *user interface*, written in JSP, consists of html pages which work as a communication channel between the user and the system throughout their recommendation request session. Another six elements embedded on the server side cooperate to generate recommendations.

The recommendation architecture is implemented in TIP using the *Struts* framework where the entire recommendation process is controlled by the Struts *ActionServlet*. We generated 14 Action classes, each of which has a corresponding Java utility class representing, each of the recommendation algorithms. To generate the recommendations, the action class selected by the ActionServlet calls its corresponding utility class as well as establisheing a connection to the database by implementing the *DataSource* interface. User and sight information retrieved from the database is stored in their corresponding data classes and wrapped up by their corresponding container classes. When the recommendation generating process is finished, a corresponding response is created. Then the *ActionForward* instance, describing where and how the control should be forwarded, is returned. The response is passed on to the corresponding destination, as defined in the *struts-config.xml* configuration file. Recommended sight information is displayed to the user via a corresponding JSP, if the recommendation request succeeds. Otherwise, an error message is given.

### 9.1.3 Evaluation

We proposed our evaluation plans in order to confirm that the design and implementation of the recommendation component we carried out in this study can remedy the existing limitations. In Chapter 6, we designed our evaluation plan based on the implications gained from an investigation of the existing recommendation systems evaluation techniques. Our goal was to measure the effectiveness of the filter algorithms with regard to quality of the generated recommendations and the system performance. We also focused on the ability to convey to the users the reasoning used by the recommendation component. Accordingly, we proposed a combination of measures. One was the *qualitative evaluation* which examined a remedy on the applied recommendation algorithms and the user acceptance related issues. The other was the *quantitative evaluation* which analyzed the system performance related issues.

#### 9.1.3.1 Qualitative Evaluation

We carried out the *qualitative evaluation* using the explanatory study as described in Chapter 7. The evaluation concentrated on addressing the first two categories of the existing limitations, the *applied recommendation algorithms* and the *user acceptance* related issues.

- **Applied recommendation algorithms**

  To illustrate that the designed recommendation algorithms successfully remedy the drawbacks of the recommendation paradigms we adopted and employed in our implementation, we carried out and illustrated our experiment through the two test scenarios introduced in Chapter 1. The test setting consisted of the actual user data created and utilized by the TIP development team and the sight information supported by the Hamilton City Council. The results of the evaluation showed positive outcomes as follows:

  - the idea of using other types of user information to replace their missing feedback scores, highly required by the collaborative filtering algorithms, effectively remedies the problem of not being able to provide recommendations to the *new user* and the *cold-start* problem;
  - the *grey sheep* problem can also be rectified by this idea since the users are not strongly encouraged to provide their feedback scores to the system. Inaccurate or poor recommendations, caused by inconsistent opinions given by these users, will be lessened. However, we regarded this as a short term solution. We have adopted an idea proposed by [3] to be employed as a long term solution in the next version of the recommendation component;
  - the *over-specialization* problem which is the main drawback of the content-based recommendation model, is fixed by providing the users *serendipitious* recommendations . We applied the concept of the collaborative filtering algorithm in the extended content-based recommendation algorithms. We proposed two new *user-user* similarity calculations. One was the *similarity on sight groups of interests*. The other was the *similarity on sight of interest*. Recommendations given by these neighbourhood of the active user will help them find surprisingly interesting sights they might miss from using the ordinary content-based recommendation algorithm.

- **User acceptance**

  To address the user acceptance issue, we conducted a first-cut analysis on the fluidity of interactions between the users and the recommendation component. We partitioned the analysis into two subjects, *acceptability* and *tractability*.

  - *Acceptability*

    Acceptability from the users depends on issues of initial personalization costs (e.g., for creating user profiles), and the quality of the recommendations. As supported by the study conducted in [36], we believe the user is willing to give a little more input to the system in order to receive more accurate recommendations. To support user understanding of why a particular sight is recommended, we delivered both *basic information* (e.g., name, group name, a link

to detailed information and picture) and *community rating information* to the users. A brief explanation on the recommendation option chosen by the user is also given. We believe this information aids users in decision making to visit sights of genuine interest to them. This reflects the quality of our implementation.

– *Tractability*

  Tractability is a long term functionality test regarding user acceptance which leads to the issue of *trust* in a recommendation system. We believe that asking the user to define their interests to the system will convey the reason why they receive a given recommendation from the system. Accordingly, they perceive that the system understands their preferences and they are inclined to trust the system more. The basic information about a sight and its community rating also aids the user in decision making to visit the sights of genuine interest.

To confirm this analysis, we planned to conduct a usability test in a further study on both how much information is appropriate to ask from the users regarding their recommendation requirements and the types of information the users find useful to support their decision making .

#### 9.1.3.2 Quantitative Evaluation

The *quantitative evaluation* was conducted as described in Chapter 8. Our focus was on the third category of limitations, the *system performance*. We carried out the experiments to verify the system computational complexity, response time and coverage of the four recommendation algorithms using collaborative filtering paradigm. The analysis of the computational complexity of each of the algorithms showed the number of users and number of sights was highly influential, resulting in a complexity of $O(m^2n)$ where $m$ is the number of users and $n$ is the number of sights in the system.

We set up an experimental scenario to measure the system response time in order to practically depict the level of complexity that the implemented algorithms demonstrate when the active user requests a recommendation. Our main focus was on the impact of the increasing number of the two major variables, the *users* and the *sights*, on the *user-user* similarity calculation step, since it turns out to be a performance bottleneck. The results have confirmed that complexity of the algorithms with regard to the increasing number of the users is $O(m^2)$ whereas the increasing number of sights reveals a complexity of $O(n)$. The experiments on the *coverage* values generated by each of these algorithms were carried out in order to verify the system's performance with regard to a prediction mechanism of the recommendation algorithms.

Our conclusion from the results of the system performance with regard to *complexity*, *response time* and *coverage* was that, among the four algorithms, using the default value to replace the missing feedback score outperforms the others. It required the lowest complexity and produced the highest coverage value. Furthermore, the percentage of the users who get recommendations was also high. Meanwhile, the extended collaborative filtering algorithms using

user information (e.g., user profile and/or user travel history) also showed appealing results. Further studies on users' satisfaction are required to examine these extended recommendation algorithms, in order to find recommendation algorithms with appropriate response time and high user satisfaction.

To summarize, our study has addressed all the limitations we identified at the beginning. The lack of user information used for building a user model is effectively remedied. The system's user interfaces have been re-designed in order to clearly convey the reasoning behind given recommendations and a feeling of being in control is facilitated for the users. Scalability of the system has been examined and analyzed. The results have uncovered some appealing aspects for further study. The overview of the conclusion of what we have accomplished in this thesis is shown in Table 9.1.

Table 9.1: Limitations identified at the beginning of the study. Three of the limitations can only ever be improved but not solved. Limitations marked with ++ have been solved, limitations marked with + have been improved, limitations marked with +− have been partly solved/improved and require further study.

| **Description** | **Solved** | **Improvements** |
|---|---|---|
| New user problem | ++ | |
| Cold start problem | ++ | |
| Grey Sheep problem | +− | |
| Over-specialization | ++ | |
| Transparency and user control | | + |
| User Satisfaction | | + |
| Scalability | | +− |

## 9.2 Problems Encountered

Some significant problems encountered with regard to the design and implementation of the recommendation component in the TIP system and from the review of the evaluations are as follows:

- As the current prototype of the TIP system is implemented using the central database approach, the design of a recommendation algorithm requires a careful consideration of the structure of the database. Using a hierarchical structure to classify a sight into its sight groups and sight related information to its topic and subtopic, is theoretically promising. The database administrator, whose task is to collect sight information from several sources and then insert it into the database, can easily carry out their tasks. In practice, it prompts two significant problems for the

implementation of the recommendation component. One is the representation of sight group and topic selection given to the users. Creating a user profile has high initial cost if the user is required to follow a hierarchical selection. These tedious tasks will give the users a bad impression of the system which may prevent them from using the system in the long run. The other problem is that matching a particular sight to its sight group requires more complex queries which leads to an expensive implementation with regard to response time and computational complexity.

- We carried out the evaluations using synthesized data sets because gathering natural data to construct the complete system is still underway. We utilize these simulated data sets as a way of testing the implemented algorithms . Herlocker et al. [13] argued that the synthesized test data gives us an easy way to test algorithms for obvious flaws, but they in no way accurately model the nature of real users and real data. Accordingly, drawing comparative conclusions from the experiments using synthetic data sets only is risky because the data may fit one algorithm better than another. Although the current results from the experiments on the synthesized data sets are solely observational, extensive tests using natural test data should be performed to compare and confirm the results.

## 9.3 Future Work

For future work we propose to follow three main directions:

- First, the proof-of-concept implementation of the recommendation component in the TIP system needs to be tested extensively. The results and review from the first qualitative and quantitative evaluations have shown some promising points for further study (e.g., impacts of the increasing number of sight groups stored in the user profile on complexity of the implemented algorithms; employment of the idea proposed in [3] as a long term measure for the *Grey Sheep* problem.)

- Second, collecting natural user information from the actual usage of the system and fully employed actual GPS signal and mobile devices is significant since this will uncover the mobility behaviour of the recommendation component.

- Finally, long term user studies with regular TIP system users are required. The study will be based on an integration of experimental tests of both the user interface and the recommendation component.

# Bibliography


[1] Scottish citylink online bus ticket booking. available at `http://www.citylink.co.uk/howtobuy.htm` Accessed on 19/4/2005 4:42 p.m.

[2] Yahoo! alerts. available at `http://alerts.yahoo.com/` Accessed on 18/4/2005 10:30 a.m.

[3] L. Adam, A. Pietek, and J. McCrae. Collective information. available at `http://www.imperialviolet.org/suprema_cf.htm` Accessed on 11/2/2006 6:16 p.m.

[4] I. Amendola, F. Cena, L. Console, A. Crevola, C. Gena, A. Goy, S. Modeo, M. Perrero, I. Torre, and A. Toso. UbiquiTO: A multi-device adaptive guide. In S. A. Brewster and M. D. Dunlop, editors, *Mobile HCI: Mobile Human-Computer Interaction*, volume 3160 of *Lecture Notes in Computer Science*, pages 409–414, Glasgow, UK, September 2004. Springer.

[5] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communication of the ACM*, 40(3):66–72, March 1997.

[6] R. Burke. Knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce. AAAI 99*, Orlando, Florida, 1999.

[7] R. Burke, K. Hammond, and B. Yong. The findme approach to assisted browsing. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):32–40, 1997.

[8] H. Chen and A. Chen. A music recommendation system based on music data grouping and user interest. In *Proceedings of the tenth international conference on Information and knowledge management*, Atlanta, Georgia, USA, October 2001.

[9] K. Cheverst, K. Mitchell, and N. Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Communication of the ACM*, 45(5):47–51, 1997.

[10] M. H. Goker and C. A. Thompson. The adaptive place advisor: A conversational recommendation system. In *Proceedings of the 8th German Workshop on Case Based Reasoning*, Lammerbuckel, Germany, March 2000.

[11] C. Hayes, P. Massa, P. Avesani, and P. Cunningham. An on-line evaluation framework for recommender systems. In *Workshop on Personalization and Recommendation in E-Commerce*, Malaga, Spain, May 2002.

[12] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the SIGIR'99*, Berkley, California, USA., August 1999.

[13] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[14] A. Hinze and G. Buchanan. The challenge of creating cooperating mobile services: Experiences and lessons learned. In *Proceedings of Twenty-Ninth Australasian Computer Science Conference (ACSC 2006)*, Hobart, Australia, January 2006.

[15] A. Hinze and S. Junmanee. Advanced recommendation in a mobile tourist information system. Technical Report 4/2005, University of Waikato, Computer Science Department, Hamilton, New Zealand, August 2005. 591 project report.

[16] A. Hinze and S. Junmanee. Travel recommendations in a mobile tourist information system. In *Proceedings of Information Systems and its Application ISTA'2005*, Palmerston North, New Zealand, May 2005.

[17] A. Hinze, K. Loeffler, and A. Voisard. Contrasting object-relational and rdf modeling in a tourist information system. In *Proceedings of the AusWeb*, Gold Coast, Australia, July 2004.

[18] A. Hinze and A. Voisard. Location and time-based information delivery in tourism. In *Proceedings of Advances in Spatial and Temporal Databases, 8th International Symposium*, Santorini Island, Greece, July 2003.

[19] S. Junmanee. Advanced recommendation in a mobile tourist information system. Dissertation, Computer Science Department,University of Waikato, Hamilton, New Zealand, February 2005.

[20] G. Linden, B. Smith, and J. York. "amazon.com recommendations": Item-to-item collaborative filtering. *IEEE Internet Computing*, 07(1):76–80, January/February 2003.

[21] K. Loeffler. User adapted information delivery in context-aware systems. Master's thesis, University of Berlin, Berlin, Germany, January 2004.

[22] P. Melvile, R. Mooney, and R. Nagarajan. Content-boosted filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Canada, July 2002.

[23] S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88, January 2004.

[24] R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, San Antonio, Texas, USA, June 2000.

[25] Q. Nguyen, D. Cavada, and F.Ricci. On-tour interactive travel recommendations, in information and communication technologies in tourism 2004. In *Proceedings of the International Conference*, Cairo, Egypt, January 2004.

[26] P. Ottlinger. Mobile tourist information system. Master thesis, University of Berlin, Berlin, Germany, February 2004.

[27] M. Papagelis and D. Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. In *Proceedings of Cooperative Information Agents VIII, 8th International Workshop,CIA 2004*, volume 3191 of *Lecture Notes in Computer Science*, pages 830–831, Erfurt, Germany, September 2004. Springer.

[28] C. Paris. Information delivery for tourism. *IEEE Intelligent System*, 17(6):61–63, November/December 2002.

[29] C. Paris. Travel recommender systems. *IEEE Intelligent System*, 17(6):55–57, November/December 2002.

[30] A. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. MaNee, J. A.KinStan, and J. Reidl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of International Conference on Intelligent User Interfaces IUI'02*, San Francisco, CA ,USA, January 2002.

[31] F. Ricci and F. Misser. Personalized product recommendation through interactive query management and case-based reasoning. In *Proceedings of CHI 2003 Workshop on Designing Personalized User Experiences for eCommerce: Theory, Methods, and Research*, Fort Lauderdale, April 2003.

[32] F. Ricci and F. D. Missier. *Designing Personalized User Experiences for eCommerce*. Kluwer Academic Publisher, the Netherlands, 2004.

[33] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of 10th International World Wide Web Conference ,WWW10*, Hong Kong, May 2001.

[34] S. Simon. Measures of association and agreement:pearson correlation. available at `http://www.childrens-mercy.org/stats/definitions/correlation.htm` Accessed on 5/5/2005 1:33 p.m.

[35] R. Sinha and K. Swearingen. The role of transparency in recommender systems. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 830–831, London, UK, April 2002.

[36] K. Swearingen and R. Sinha. Interaction design for recommender systems. In *Proceedings of Designing Interactive Systems, DIS 2002*, Minneapolis, Minnesota, USA, April 2002.

[37] E. Vozalis and K. Margaritis. Analysis of recommender system's algorithm. In *Sixth Hellenic-European Conference on Computer Mathematics and its Applications(HERCMA)*, Athens, Greece, September 2003.

[38] A. Zipf. Adaptive context-aware mobility support for tourists. *IEEE Intelligent System*, 17(6):57–59, November/December 2002.

# Appendix A

# Database Structure of TIP

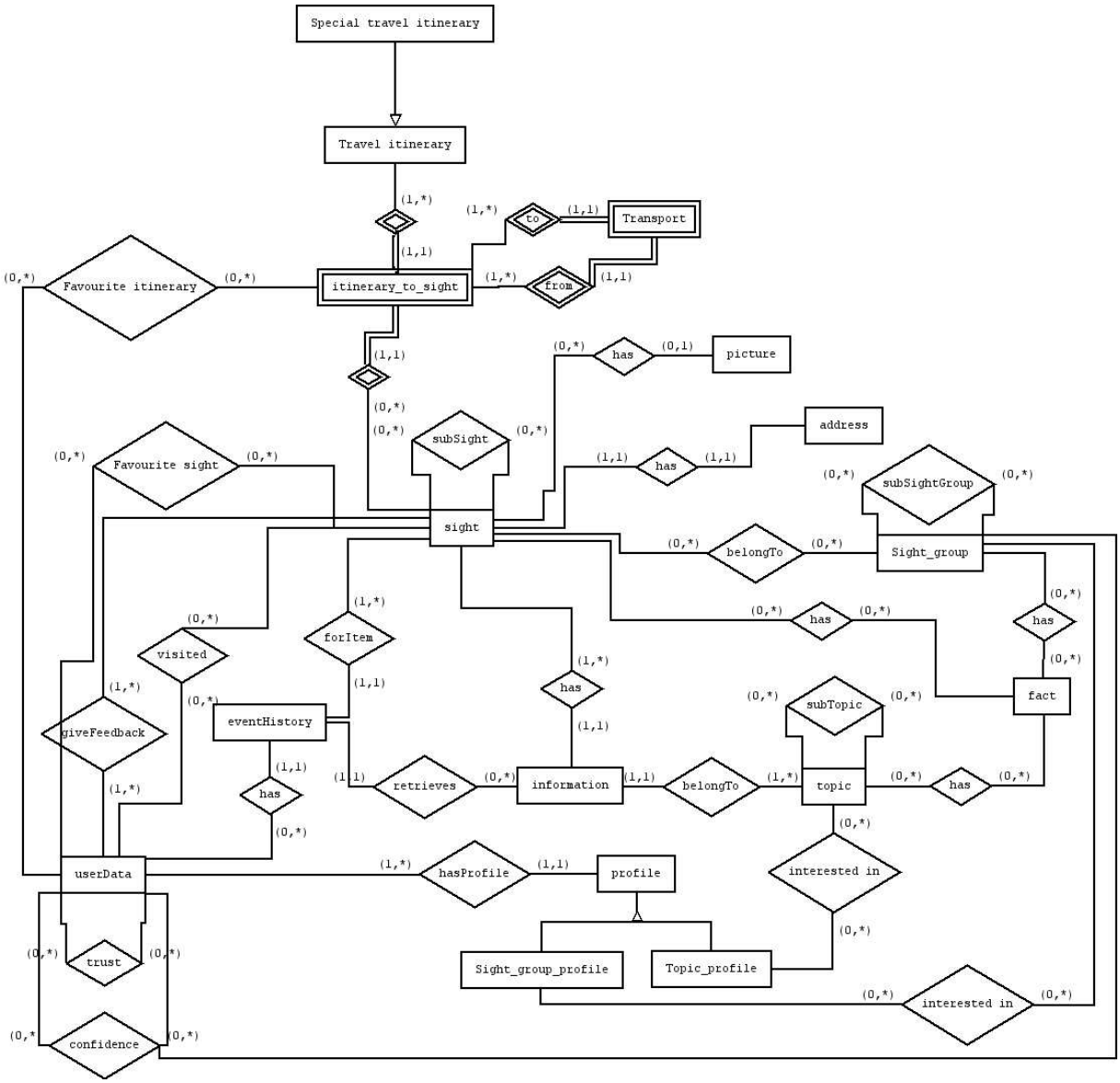



Figure A.1: Entity relationship diagram of the TIP database