

**An Investigation on  
Personalized Collaborative  
Filtering for Web Service  
Selection**

Kenneth Karta

*This report is submitted as partial fulfilment  
of the requirements for the Honours Programme of the  
School of Computer Science and Software Engineering,  
The University of Western Australia,  
2005*

# Abstract

The current Web service architecture addressed service *discovery* problem, but not service *selection*. If we treat services as special type of products from the service providers, existing techniques used for product selection can be applied to service selection. Among many existing techniques for product selection, one dominant approach is to use recommender systems. For example many e-commerce Web site, such as Ebay, Amazon and Epinions all have recommender system support to ease the burden of product selection. Recommender systems are classified into different types, which include content-based, collaborative and hybrid based systems which perform recommendation based on user/item data set. There is also multidimensional recommender system where the system supports multiple dimensions, such as *user,item*, and its quality attributes.

This research investigates the fundamentals of different types recommender systems. In particular, the collaborative filtering based approach using both two dimensional and multidimensional data. This study is to facilitate the development of a framework that supports service selection. In this project we are interested in applying multidimensional collaborative recommender system for Web service selection. But before we can propose the framework, we first perform experiments on the similarity measures used in collaborative filtering to fully understand the working process of normal collaborative filtering system. Pearson correlation and Vector Similarity are tested and compared using MovieLens data set. This experiment is important to decide which of the similarity measures perform better in two dimensional data set.

Because the Quality of Service is often characterized by multiple criteria, and different people/users may put strong emphasis on certain criteria than the others, two dimensional approach is over-simplified. Therefore we looked at the multi dimensional approach on product selection as well. An experiment using the multidimensional data collected in this research from Epinions.com is performed to study how it works on product selection. Finally, we draw from the experience of both two dimensional and multidimensional product selection and propose a framework for Web service selection which uses the multidimensional recommender system approach.

**Keywords:** Collaborative filtering, Multidimensional Recommender system, Recommender System, Web Service, Quality of Service.

**CR Categories:** I.2 [Artificial Intelligence], I.2.8 [Problem Solving, Control

Methods, and Search], H.3 [Information Storage and Retrieval], H.3.5 [Online Information Services].

# Acknowledgements

First i would like to thank Dr. Wei Liu for the encouragement and support through out the honours year. Without her help, this project would not have gotten as far as it is.

I would also like to give thanks to my family for all the support throughout the year. Finally i would also like to thank my fellow Honours student for the good times in the lab.

# Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Definition and Motivation . . . . .	1
1.2 Organization . . . . .	2
<b>2 Literature Review</b>	<b>3</b>
2.1 Recommender System . . . . .	3
2.1.1 Content-based Recommendation . . . . .	3
2.1.2 Collaborative recommendation . . . . .	4
2.1.3 Hybrid Recommendation . . . . .	7
2.2 Web Service . . . . .	8
2.2.1 Web Service Selection . . . . .	9
2.2.2 Quality of Service Ontology . . . . .	11
<b>3 Multidimensional Recommendation System</b>	<b>13</b>
3.1 Multiple Dimension . . . . .	13
3.2 Rating Estimation Technique for Multidimensional Recommender System . . . . .	15
3.2.1 Reduction-Based Approach . . . . .	15
3.2.2 Multi Level Rating Estimation . . . . .	16
<b>4 Experimental Evaluation on Collaborative Filtering Similarity Measure</b>	<b>18</b>
4.1 Aim . . . . .	18
4.2 Data set . . . . .	18

4.3	Evaluation Metrics . . . . .	19
4.4	System Configuration . . . . .	20
4.5	Implementation . . . . .	20
4.6	Methodology . . . . .	21
4.7	Experimental Results . . . . .	22
<b>5</b>	<b>Experimental Evaluation on Multidimensional Recommender System</b>	<b>27</b>
5.1	Aim . . . . .	27
5.2	Data set . . . . .	27
5.3	Evaluation Metrics . . . . .	28
5.4	System Configuration . . . . .	28
5.5	Implementation . . . . .	28
5.6	Methodology . . . . .	29
5.7	Experimental Results . . . . .	29
<b>6</b>	<b>Web Service Selection Using Collaborative Filtering Framework</b>	<b>31</b>
6.1	Web Service Selection Framework . . . . .	31
6.1.1	Web Service Directory Facilitator Agent . . . . .	31
6.1.2	User Registry Agent . . . . .	33
6.1.3	Quality of Service Ontology Agent . . . . .	33
6.1.4	Ranking Registry Agent . . . . .	33
6.1.5	Web Service Selection agent . . . . .	34
6.2	An Example: Selecting a Web Service . . . . .	34
6.3	Comparison With Other Framework . . . . .	35
<b>7</b>	<b>Conclusion and Future Work</b>	<b>36</b>
7.1	Conclusion . . . . .	36
7.2	Future Work . . . . .	37
<b>A</b>	<b>Original Honours Proposal</b>	<b>38</b>



# List of Tables

2.1	Agents and the services they have rated . . . . .	10
4.1	Standard deviation on different dataset . . . . .	26
5.1	Result of Collaborative recommender system run on various di- mension . . . . .	29

# List of Figures

2.1	Web Service Architecture . . . . .	8
2.2	Overview of Service Selection System [10] . . . . .	9
4.1	MAE and recall graph using ua.base and ua.test data set . . . . .	22
4.2	MAE and recall graph using ub.base and ub.test data set . . . . .	23
4.3	MAE and recall graph using u1.base and u1.test data set . . . . .	23
4.4	MAE and recall graph using u2.base and u2.test data set . . . . .	24
4.5	MAE and recall graph using u3.base and u3.test data set . . . . .	24
4.6	MAE and recall graph using u4.base and u4.test data set . . . . .	25
4.7	MAE and recall graph using u5.base and u5.test data set . . . . .	25
6.1	The Web Service Selection Framework . . . . .	32
A.1	Web Service Architecture . . . . .	38

## CHAPTER 1

# Introduction

### 1.1 Problem Definition and Motivation

A Web service is a software system designed to support interoperable machine-to-machine interaction over the Internet. It provides a standard means of inter operating between different software applications, running on various of platforms or frameworks [4]. Some example of Web services include hotel reservation, flight booking, online shopping and many more.

Web service architecture consists of Web service *consumer*, Web service *provider* and a *service registry*. Web service provider register their service to the public service registry called Universal, Description, Discovery and Integration (UDDI). Web service consumer then searches the registry to find a service that matches their requirement. The ability to search for appropriate Web services based on functional descriptions is known as Web service *discovery* [4]. The current Web service architecture addresses the service discovery problem, but not service *selection*. Web service selection deals with selecting a service implementation from those that are discovered from the given non-functional description [10]. Web service selection is important because it deals with selecting services that are best related to the consumer, while services that are returned from service discovery are not ranked and thus not in particular useful to help the consumer with a unique selection.

If we treat Web services as special type of products from the Web service providers, existing technique used for product selection can also be applied to Web service selection. Recommender system is the commonly used technique for product selection. There are different types of recommender systems, content-based, collaborative and hybrid recommender system, where each recommender system has its own strength and weakness. There are now many e-commerce Web sites that uses recommender systems to provide recommendation to the user, Ebay, Amazon and Epinion all have recommendation services to ease the burden of product selection. Current recommender systems usually perform rec-

ommendation only based on a two dimensional user/item data set, however the recommender system can be extended to work on a multidimensional environment to provide more accurate and personalized recommendation. This type of recommender system is known as multidimensional recommender system.

The multidimensional recommender system uses user, item as well as item's quality attributes for recommendation. In Web service, this quality attributes are referred to as the Quality of Service (QoS). Performance, reliability, availability and security are some examples attributes of QoS. There are very limited work available on using recommender system for Web service selection as it is a new emerging area, Therefore, our aim is to present a framework on Web service selection using multidimensional recommender system. Due to no freely available data on Web service, we are unable to implement the actual framework. On the contrary, Web service is similar to product selection where both involve mapping a set of products/services to a ranking of the products/services in that set. Therefore any implementation on product selection can technically be implemented on service selection as well. Before we present the framework, we will conduct an experiment on collaborative recommendation similarity measures. The Pearson Correlation and Vector Similarity will be tested using the MovieLens [7] data set. This experiment is important to decide which of the similarity measures perform better in two dimensional data set and critical to enable us fully understand the working of a typical recommender system. We will then perform an experiment on product selection using multidimensional recommender system. There are no freely distributed multidimensional data set available, therefore we gathered our own data from the Epinion.com Web site and use it in the experiment. Build on the experiment of running two dimensional and multidimensional collaborative filtering, we proposed an infrastructure for enabling recommender system based Web service selection.

## 1.2 Organization

Firstly, we discuss the different type of recommender system, which include content-based, collaborative and hybrid recommender system in Chapter 2. In Chapter 2 we will also discuss Web service and its quality of service. Chapter 3 describes previous work on multidimensional recommender system. An experiment evaluation on collaborative filtering similarity measures will be discussed in Chapter 4. While Chapter 5 will discuss an experiment run on multidimensional recommender system. We will then present our framework on Web service selection using Collaborative filtering in Chapter 6. Finally, the dissertation will be summarized and concluded in Chapter 7.

## CHAPTER 2

# Literature Review

## 2.1 Recommender System

With the rapid growth of information on the Web, software tools are needed to assist users to find information they are looking for. Recommender system can help user to solve the problem by providing them with personalized suggestions. In a typical recommender system existing users provide recommendations or rankings as input, which the system then aggregate and directs to appropriate recipients. Recommender system has been used by many e-commerce companies to do various filtering including recommending Websites, books, DVD, and many other products. For example, the Amazon.com recommendation system starts by finding a set of customers whose purchased and rated items overlap the users purchased and rated items. The system aggregates items from these similar customers, eliminates items the user has already purchased or rated, and recommends the remaining items to the user. In this section, we will look at different types of recommendation systems including *content-based*, *collaborative* and *hybrid recommendation*.

### 2.1.1 Content-based Recommendation

A content-based system recommends items using a profile built up by analyzing the content of items that the user has rated. The content-based filtering first analysis items rated by an individual user and use the content of the item as well as the provided ratings to build up the user's profile, it then uses the profile to compare to other items that the user has not rated yet to recommend additional items of interest [6].

In a content-based recommendation, the rating  $R(u,i)$  of item  $i$  for user  $u$  is usually obtained from the ratings  $R(u,i')$  assigned by the same user  $u$  to other item  $i'$ , where  $i'$  are items that are "similar" to item  $i$  in terms of their content. For example, in a movie recommendation scenario, in order to recommend a movie to

user  $u$ , the content-based recommender system has to understand the user preferences by analysing the user profile and find the commonalities among the content of the movies user  $u$  has rated highly in the past. Then, only the movies that have high degrees of similarity with the user profile, would be recommended [1].

Content-based recommendation technique has its weaknesses. Generally, content-based technique have difficulty in distinguishing high quality and low quality information that is on the same topic, and as the number of items grow, the number of items in the same category increases, further decreasing the effectiveness of content-based technique [6]. The second problem that content-based encounter is *over-specification*. The over-specification problem occurs when the system can only recommend item scoring highly against the user's profile, the user is restricted to seeing items similar to those already rated. The last problem is common to most recommender system, it is the *eliciting user feedback problem*. Rating items is a time-consuming process for the users, so the fewer the rating required the better. But for the content-based approach to work, a user's ratings are the only factor influencing the performance, and there seem to be no way to reduce the quantity without reducing the performance [3].

## 2.1.2 Collaborative recommendation

Collaborative filtering uses a database about user preference to predict items that a new user might like. The task of Collaborative filtering is to predict the utility of services to the active user based on the user's previous likings or database of user votes from a population of other users. A Collaborative Filtering algorithm can be used for one of the two purposes [12]:

1. Prediction: is a numerical value expressing the predicted likeliness of an item for the active user.
2. Recommendation: is a list of  $N$  items, that the active user will like the most. Also known as the Top- $N$  recommendation.

Collaborative Filtering algorithms can be classified into two main categories, Memory-based (user-based) and Model-based (item-based). We will have a look at each of them in detail in section 2.1.2 and 2.1.2.

### Memory-Based Collaborative Filtering

Memory-based algorithm uses the entire user-item database to generate prediction. The database usually consist of a row indicating the users, and column

indicating the item to be rated. The memory-based algorithm employ statistical analysis to find a set of users called as *neighbours* that have similarity with the active user rating. Once a set of neighbours is formed, the systems use different algorithms to combine the preference of neighbours to produce a prediction on the top-n recommendations for the active user [12]. The user database consists of votes  $V_{i,j}$ , which correspond to the vote of item  $j$  made by user  $i$ . If  $I_i$  is the set of items on which user  $i$  has voted, we can define the mean vote for user  $i$  as [5]:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} V_{i,j} \quad (2.1)$$

To predict vote of a active user  $a$  for item  $j$ ,  $p_{a,j}$  can be calculated using the following formula [11]:

$$p_{a,j} = \bar{v}_a + \frac{k \sum_{i=1}^n w(a,i)(v_{i,j} - \bar{v}_i)}{\sum_{i=1}^n |w(a,i)|} \quad (2.2)$$

where  $n$  represent the number of user in the database that has nonzero ranking. In the formula,  $k$  is a normalizing factor such that the absolute values of the weights sum to unity. The function  $w(a,i)$  can represent, distance, correlation or similarity between user  $a$  and  $i$ . Typically, two types of similarity measures can be used here, the Pearson correlation and the Vector similarity.

**Pearson Correlation:** This general formulation of collaborative filtering first appeared in the context of GroupLens project, where the Pearson Correlation coefficient was defined as the basis for the weights [5], the correlation between users  $a$  and  $i$  is:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \quad (2.3)$$

where the summations over  $j$  are over the items for which both user  $a$  and  $i$  have recorded votes.

**Vector Similarity:** The similarity between two documents is often measured by treating each document as a vector of word frequencies and computing the cosine of the angle formed by the two frequency vectors. We can adopt this formalism to collaborative filtering, where users take the role of documents, titles take the role of words, and votes take the role of word frequencies. Note that under this algorithm, observed votes indicate a positive preference, there is no role for negative votes, and unobserved items received a zero vote. The relevant weights are now:

$$w(a, i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}} \quad (2.4)$$

The squared terms in the denominator serve to normalize votes, so that users that vote on more titles will not a priori be more similar to other users.

### Model-Based Collaborative Filtering

The Model-based collaborative filtering provides item recommendations by first developing a model of user ratings. The model based algorithm takes a probabilistic approach. It treats the collaborative filtering process as a computation for the value of user rating, given the user previous ratings on other item. The first step in the algorithm is to look into the set the active user has rated and computes how similar they are to the target item. Once the similarity has been computed, the prediction is computed by taking a weighted average of the target user's ratings on these similar items [12].

One important step is to compute the similarity between items and then to select the most similar item. To compute the similarity between item  $i$  and  $j$ , the isolation of user who have rated both items is needed. Again the correlation-based similarity and the vector-based similarity measures are the popular methods for finding the similarity measures between items. Both of these techniques are the same as what used in memory-based algorithm mentioned above.

Once the isolation of the set of most similar items based on the similarity measures are completed, the next step is to look into the target user's ratings and use a technique to obtain predictions. One example of the prediction technique is called Weighted sum.

**Weighted sum:** this method computes the prediction on an item  $i$  for a user  $u$  by computing the sum of the ratings given by the user on the items similar to  $i$ . The prediction  $P_{u,i}$  can be denoted as:

$$P_{u,i} = \frac{\sum_{all\ similar\ items\ j} (s_{i,j} * R_{u,j})}{\sum_{all\ similar\ items\ j} (|s_{i,j}|)} \quad (2.5)$$

Where  $S_{i,j}$  denotes similarity between item  $i$  and  $j$ . and  $R_{u,j}$  denotes the ratings of user  $u$  on item  $j$ . Basically this approach tries to capture how the user rates the similar items. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range.

Both the content-based and collaborative recommender system suffer from the *new user problem*, the problem is caused when a new user is introduced into the

system. A user need to rate sufficient number of item before the recommender systems can provide a reliable recommendation. Since a new user will have none or very few ratings, the systems would not be able to produce an accurate recommendations to the users [1].

In addition to the *new user problem*, collaborative recommendation also suffer from the *new item* problem, since they rely solely on rating data to make recommendations. Therefore the collaborative recommender system would not be able to recommend new items until it is rated by a substantial number of users. The *sparsity* of ratings is another common problem that collaborative recommender system encounter. This problem refers to situation where the user ratings data are sparse and insufficient to identify similarities between users, causing the recommender system to produce inaccurate recommendation [14].

### 2.1.3 Hybrid Recommendation

In a Hybrid recommendation approach, the content-based and collaborative filtering are used together for recommendation. There are many different type of Hybrid recommender system, Balabanovic and Shoham approach taken in the Fab system was, for each user has to have a profile gathered based on the content analysis, the profile is then compared with the other users to determine similar users for collaborative recommendations. By using this approach, users will receive items both when they score highly against their own profile, and when an item is highly rated by user with a similar profile [3].

Two special cases arise when using this type of hybrid filtering system. The first case, if the content analysis component returns just a unique identifier rather than extracting any features, then the hybrid filtering will reduce to pure collaborative recommendation. The second case, if there is only a single user, then it will reduce to pure content-based recommendation.

Another approach to build a hybrid recommender system is to implement a separate collaborative and content-based recommender system. Then we can combine the outputs (ratings) obtained from individual recommender system into one final recommendation using either a linear combination of ratings or a voting scheme. Alternatively, we can choose to use the one that is “better ”than the other based on some recommendation quality metrics. These metrics can be a recommendation with a higher level of confidence or a recommendation with higher consistently with past ratings of the user [1].

By having content-based and collaborative recommendation together, the hybrid system exhibits the following advantages [3] :

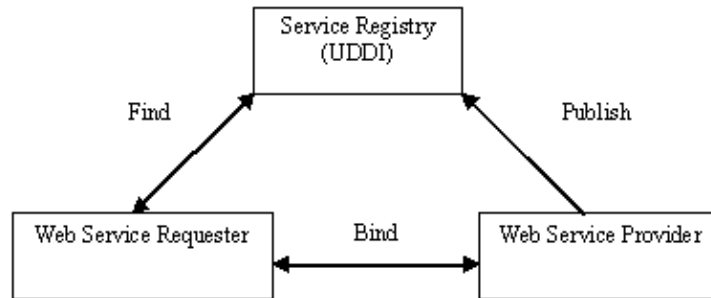


Figure 2.1: Web Service Architecture

- By using collaborative recommendation, it can use other people's experience as a basis rather than incomplete and imprecise content analysis.
- By using content-based recommendation, it can deal with item unseen by others.
- The profile that was built from content of items, can be used to make good recommendation to users, even if there are no other users similar to them.
- Collaborative recommendation can be made between users who have not rated any of the same items, as long as they have rated similar items.
- By utilizing group feedback, the system will only require fewer cycles to achieve the same level of personalization.

## 2.2 Web Service

Web service is any software that makes itself available over the Internet and uses a standardized XML messaging system. XML is used to encode all communications to a Web service. For example, a client invokes a Web service by sending an XML message, then waits for a corresponding XML response. Because all communication is in XML, Web services are not tied to any one operating system or programming language.

Figure 2.1 describes the structure of Web services. Web services providers uses Web Service Description Language (WSDL) to describe the service that are provided. The service providers publish their services to a public service registry using Universal, Description, Discovery and Integration (UDDI). Service consumer discover service in the registry and obtain a URL for the WSDL file

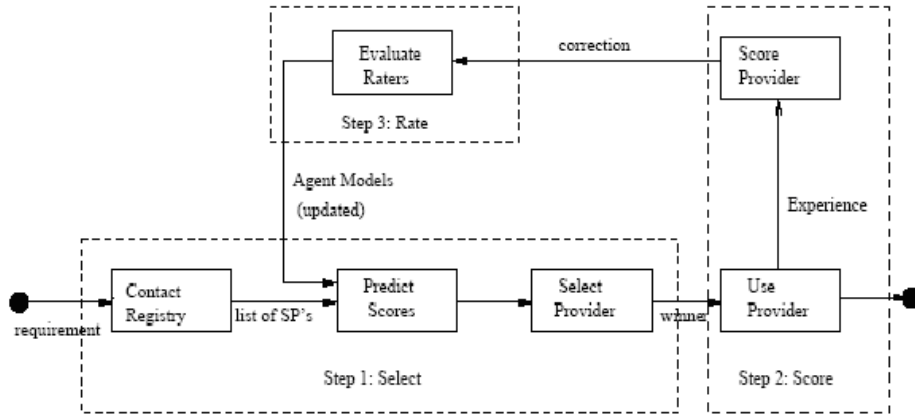


Figure 2.2: Overview of Service Selection System [10]

that describe the service. The Service then can be invoked using the XML based Simple Object Access Protocol (SOAP) [2]. To establish the binding between a service requester and a service provider, service discovery and service selection should be supported by the Web service infrastructure. Service discovery deals with finding the service implementation that meet a specified description. While service selection deals with choosing a service implementation from among those that are discovered for the given description. But the current infrastructure of Web services does not support a critical need of selecting the right service types, due to the lack of means in expressing and collecting a service's non-functional attributes. Namely, its Quality of Services (QoS). QoS can be objective (encompassing reliability, availability and request- to response time) or subjective (focusing on user experience on using the services). With the help of QoS policy, Web service selection can be improved. By using QoS, user can specify the desired quality's of services. In this section Web service discovery, Web service selection and Quality of Service will be explained briefly.

### 2.2.1 Web Service Selection

Web service discovery deals with finding Web services implementations that meet some specified description. This discovery is done by searching the service with matching description and service profile on the UDDI. Service selection deals with choosing a service implementation from among those that are discovered for the given description. An example of a Web service selection system will be discuss next.

	<b>a</b>	<b>b</b>	<b>c</b>	<b>d</b>
N1	0.3	0.5	-	-
N2	0.2	0.4	0.1	0.5
N3	-	0.3	0.2	-

Table 2.1: Agents and the services they have rated

Singh and Sreenath [10] has developed a system that uses Collaborative Filtering for Web service selection. Their system works by firstly sending a query to the UDDI registry to acquire the list of service providers that are available. The consumers need to query other consumers to estimate the service quality offered by different service providers, which will return the list of service providers with a score associating with each provider. The scores indicates the evaluation made by other consumers; a null score represents that the consumer does not know about the given provider.

The raters and their score are stored in a two-dimensional matrix, with the column representing the different providers and the rows representing the rates from each users. From Table 2.2.1, let  $N1$ ,  $N2$ ,  $N3$  be the three consumers who have rated 4 service providers,  $a$ ,  $b$ ,  $c$ ,  $d$ . Not all consumers have rated the providers. By adapting the formula for memory-based algorithm,  $S_{a,j}$  represent the score given by  $N_a$  to provider  $j$ .  $\bar{S}_a$  represent the average score given by  $N_a$  to all providers and  $R_{a,i}$  represent  $N_a$ 's rating of provider  $i$ . Below is the formula.

$$S_{a,j} = \bar{S}_a + \frac{\sum_i ((S_{i,j} - \bar{S}_i) * R_{a,i})}{\sum_i R_{a,i}} \quad (2.6)$$

Finally, the provider that obtains the highest score ( $S_{a,j}$ ) is chosen.

After using the provider, the user can evaluate the providers by giving score to individual features that are important to the user. The score will then be used to calculate the overall score by calculating the weighted sum. Figure 2 described the overall architecture of the system developed by Singh and Sreenath [10].

Singh and Sreenath tested the system using an artificial dataset as there are no extensive real dataset of scoring of open services are available. There are some good result from the experiment, however the system was not actually implemented yet. Even though the system developed by Singh and Sreenath was logical, they do not consider quality of service attributes of Web services in their recommendation system.

## 2.2.2 Quality of Service Ontology

Quality of Service concentrates on the non-functional requirements for different Web services. The QoS specification is an ontology that allows the matching between services request with service quality description semantically and dynamically. The semantic matching allows the service agent to match consumers to services using the provider's advertised QoS policy and the consumer's preferences [9]. Maximilien and Singh [8] have the same objective QoS attribute with the W3C Working Group [13] which include:

- **Performance:** The performance of Web services is how fast the service request can be completed.
- **Reliability:** The Reliability of Web services represent on the ability of the service to perform request under stated condition for a specific time interval.
- **Scalability:** This represent the capability of a service to handle more computing capacity or to process more user' request in a given time interval.
- **Availability:** This represents the probability of a service to be up and running.
- **Security:** Capture the level and kind of security that the service provides. Key component in security include Authentication, Encryption and Authorization.

For the complete list of the attribute of Quality of Services, it can be found at [13].

There are also subjective Quality of services, which are divided into 2 categories, which are:

- **Action-based Web service:** Provide an action type services to the consumer. Some subjective quality of attributes this type of service provide include, accuracy determine whether the action perform by the Web service are accurate, correctness determine whether the functionality of the Web service are correct and failure recovery determine how well the service handle failure.
- **Information-based Web service:** Provide an information type services to the consumer. Some subjective quality of attributes this type of service provide include, completeness determine whether the information provided

by the service is complete, and relevancy determine whether the information provided by the service is relevant to the consumer.

In product selection, sometimes it is not sufficient to make recommendation based on a single ranking value. Therefore there in product selection there are other several quality attributes to help with more personalized recommendation. In Web service selection, Quality of service attributes can also be used to help with making recommendation to the consumer. However, when there are additional attributes / dimensions, we need a different recommender system to process additional data. The Multidimensional recommender system that is discuss in the next Chapter will solve this problem.

## CHAPTER 3

# Multidimensional Recommendation System

On the previous chapter, we have discussed recommender system using collaborative filtering. The collaborative filtering approach focused on a two dimensional model, usually recommending items to users (or vice versa). However, in many applications, it is not sufficient just to recommend items to users based on singular ranking values. For example, customer preferences for selecting a movie to watch can be significantly depended on the time, place and companion they would like to watch with. In other words, the recommender system may recommend a different movie to a user depending on whether he is going with friends on a Saturday night or with his parents on a weekday. Therefore to provide useful recommendation in such applications, other dimensions such as time, place and companion have to be implemented into the system. Multidimensional recommender system provides the solution to this problem. In this section, we will explain the multidimensional recommender system developed by Adomavicius et al [1].

### 3.1 Multiple Dimension

Traditionally, content-based, collaborative and hybrid recommender system made a recommendation by using two type of entities, users and items. First the user would rate item that the user has previously have purchased, indicating how the user like the item. Based on these ratings, the system then estimate new item that the user has not rated before. The rating function of recommender system can be described by the formula bellow:

$$R : User \times Items \rightarrow Ratings \quad (3.1)$$

As mentioned before, some application does not fit into the two dimensional recommender system. From the previous example, there are more than two

entities that determined what movie to be recommended to the users. Beside the user and items entities, *time*, *place* and *companion* need to be considered when making recommendation. Therefore the use of Multidimensional recommender system need to be used in this situation.

To design the multidimensional system, let  $D_1, D_2, \dots, D_n$  be the dimensions, we can defined the recommendation space  $S$  to be:

$$S = D_1 \times D_2 \times \dots \times D_n. \quad (3.2)$$

Let Ratings be a rating domain representing the set of all possible rating values. Then, the rating function  $R$  is defined as:

$$R : D_1 \times \dots \times D_N \rightarrow Ratings \quad (3.3)$$

For example, consider the recommendation space User \* Item \* Time, where the User dimension can be defined as User = Name \* Address \* Age, where it represent the set of users having certain names, addresses and certain age. Similarly, the Item dimension could be defined as Item = Item name \* Price \* Type, where it can represents a set of items defined by their names, price and type. Finally the Time dimension can be represented as Time = Year \* Month \* Date. The Rating function  $R$  then can be defined as how much user  $u$  liked item  $i$  at time  $t$ . For example, John rated a vacation that he took to Japan on December 7-14, 2003 as 6 (out of 7). The set of attributes describing a particular dimension is called a *profile*. For example, the item's profile would consist of item's name, price and type.

The rating function  $R$  used in the both the 2D and multidimensional system is either represents a rating defined by the users or an algorithm to estimate the value of the ratings. The rating function will be discussed later on in this chapter.

When using multidimensional recommender system, an important question regarding what dimensions should be included in the system need to be resolved. To understand the issue involved, consider a simple case where a dimension  $X$  having two possible value  $X = a$  and  $X = b$ . If the distribution of rating for both  $X = a$  and  $X = b$  were the same, then dimension  $X$  would not have any impact on the recommendation made. For example, using the recommending a movie scenario, assume the dimension  $X$  to be *Place*, where it has only two value: *Theater* and *Home*. Also assume that the rating for movie watched in the theater ( $Place = Theater$ ) is the same as the rating for watching it at home ( $Place = Home$ ). In this case, the place where movies are watched(at home or in theater) does not effect movie watching experiences. Therefore the *Place* dimension can be removed from the multidimensional model [1].

Traditional recommender systems only provide recommendation of one particular type: “recommend top N items to users”. On the other hand, a multidimensional recommender system has the ability to recommend more than one dimensions to the users. For example, in the movie recommender system, the system can recommend a movie and a place to see it to a person at a certain time. Alternatively, the system can also recommend a place and a time to see a movie to a person with a companion. From this example we can see that the multidimensional system can be more personalized than the standard 2D system.

## 3.2 Rating Estimation Technique for Multidimensional Recommender System

An important question when dealing with multidimensional recommender system, is how to estimate an unknown rating. There are many method proposed for two-dimensional recommender system as described in section 2.1. However not all of these methods are directly extensible to the multidimensional case, because extra dimensions have complicated the problem.

In this section, we will discuss the rating functions that can be used in a multidimensional recommender system. We will first describe the *reduction-based* approach and followed by *multi-level rating estimation technique*.

### 3.2.1 Reduction-Based Approach

The reduction-based approach reduces the multidimensional recommendation problem to the traditional two-dimensional *User x Item* recommendation space [1]. Therefore, after the reduction to the 2D space, all the techniques suitable for 2D recommender system such as Content-based, Collaborative and Hybrid can be used.

To understand how the reduction is done, assume that:

$$R_{User \times Content}^D : U \times C \rightarrow rating \quad (3.4)$$

represent a two-dimensional estimation function that, given existing ratings  $D$  where it contains records for each of the user specified ratings, can calculate a prediction for any rating. Similarly, a three-dimensional rating function can be defined as

$$R_{User \times Content \times Time}^D : U \times C \times T \rightarrow rating \quad (3.5)$$

where  $D$  contains records  $(User, Content, Time, Rating)$  for the user specified ratings. Then the reduction of three-dimensional prediction function can be expressed as a two-dimensional prediction function as

$$\forall (u, c, t) \in U \times C \times T, R_{User \times Content \times Time}^D(u, c, t) = R_{User \times Content}^{D[Time=t](User, Content, Rating)}(u, c) \quad (3.6)$$

Where  $D[Time=t](User, Content, Rating)$  represents a rating set obtained from  $D$  by selecting only those records where  $Time$  dimensions has the value of  $t$  and keeping only the corresponding value of  $User$  and  $Content$  dimensions as well as the rating itself. In other words, by performing two relational operations, selection followed by projections, we can obtain  $D[Time=t](User, Content, Rating)$  from the set of three-dimensional ratings  $D$  [1].

Here is an example to illustrate how the reduction-based approach works. Assume that we want to predict whether **Robert** would like to read the **Dow Jones report** in the **morning**. In order to calculate  $R_{User \times Content \times Time}^D(\text{Robert}, \text{Dow Jones Report}, \text{Morning})$ , the reduction-based approach would proceed as followings: first, it would eliminate the  $Time$  dimensions by only selecting the morning ratings from the set of all ratings  $D$ . This will then reduce the problem to the standard  $User \times Items$  case on the sets of mornings ratings. Then by using any 2D recommendation technique described in section 2.1, we can calculate how **Robert** like the **Dow Jones Report** based on these morning ratings. The main idea for this approach is, if we want to predict a “morning” rating for a certain user and a certain item, we should only made the recommendation based on the previously specified “morning” ratings.

### 3.2.2 Multi Level Rating Estimation

When dealing with multidimensional recommender systems, the aggregate rating problem will arise. An example of this problem can be represented by the following scenario. Consider the movie recommender system problem, assume that Peter has provided the rating for movies he has seen ( $R(\text{Peter}, \text{Matrix})=7$ ),  $R(\text{Peter}, \text{Batman}) = 3$ ), and he also has rated the whole category *action* genre ( $R(\text{Peter}, \text{action})=6$ ). From this data, how can we use the cumulative rating of action movies  $R(\text{Peter}, \text{action})= 6$  to estimate ratings of other individual action movies that Peter has not rated yet?

This problem can be addressed as follows. Let  $R_a(\text{Peter}, \text{action})$  be the actual aggregate rating that Peter has assigned to the actions movies (e.g., 6 in the example above). Let  $R_c(\text{Peter}, \text{action})$  be the aggregate rating computed from the individual ratings  $R(\text{Peter}, \mathbf{x})$  assigned to all the actions movies, using the following expression [1]:

$$R(\text{Peter}, \text{action}) := \text{AGGR}_{x.\text{genre}=\text{action}} R(\text{Peter}, x) \quad (3.7)$$

Let  $X_r$  be the set of action movies that Peter has rated, and  $X_{nr}$  be the set of action movies that Peter has not rated yet, whose ratings we try to estimate. Then, one way to compute ratings to the action movies  $X_{nr}$  that Peter has not rated yet is to minimize the difference between the actual rating  $R_a(\text{Peter}, \text{action})$  and the computed rating  $R_c(\text{Peter}, \text{action})$ . Formally [1]:

$$\begin{aligned} & \min |R_a(\text{Peter}, \text{action}) - R_c(\text{Peter}, \text{action})| \\ & = \min_{R(\text{Peter}, x)_{x \in X_{nr}}} |R_a(\text{Peter}, \text{action}) - \text{AGGR}_{x \in X_r \cup X_{nr}} R(\text{Peter}, x)| \end{aligned}$$

In other words, we want to assign the ratings of movies in  $X_{nr}$  so that they yield the computed aggregated rating that is the closest to the rating  $R_a(\text{Peter}, \text{action})$  that was assigned by Peter himself [1].

## CHAPTER 4

# Experimental Evaluation on Collaborative Filtering Similarity Measure

### 4.1 Aim

The aim of this experiment is to understand the process of two dimensional collaborative filtering by comparing the two similarity measures, the Pearson Correlation and Vector similarity using Matlab program. In particular we look at the performance of the two on predicting user ratings on MovieLens dataset [7]. By doing so, pros and cons of similarity measures are investigated to give full understanding of collaborative filtering based recommender system. So that further experiments on multidimensional can be carried out in Chapter 5.

### 4.2 Data set

The data set we used here are provided by the MovieLens group [7]. MovieLens is a web-based research recommender system that started in 1997. Each week hundreds of users visit MovieLens to rate and receive recommendations for movies. The database now has over 43000 users who have expressed opinions on 3500+ different movies. The data set used in this experiment consist of 100,000 ratings from 943 users on 1682 movies. The ratings use a scale of 1 to 5, where 1 for dislike and 5 for strongly like. To be able to conduct experiment, the data set are divided into *training set* and *test set*.

There are two types of data set used in the experiment:

- First type of data sets are obtained by dividing the entire data set randomly into 5 partitions. For each partition, the data set are split 80%/20% into

training and test data set. Throughout the paper we term this data set as u1.base, u1.test, u2.base, u2.test, u3.base, u3.test, u4.base, u4.test and u5.base, u5.test.

- Second type of data sets are obtained by dividing the entire data set randomly into 2 partitions. For each partitions the data set are split into a training set and test set, with exactly 10 ratings per user in the set. Throughout the paper we term this data set as ua.base, ua.test and ub.base, ub.test

### 4.3 Evaluation Metrics

Recommender systems research has used several types of measures for evaluating the quality of a recommender system. The two type of metrics used in this experiment are [12]:

- *Statistical accuracy metrics* evaluates the accuracy of the system by comparing the numerical recommendation scores against the actual user ratings for the user-item pairs in the test data set. *Mean Absolute Error* (MAE) is one example of this metrics. MAE is a measure of the deviations of recommendations from their user-specified values. For example, for each rating-prediction  $\langle p_i, q_i \rangle$  this metric treats the absolute error between them  $|p_i - q_i|$  equally. The MAE first computed the sum of these absolute errors and then computed the average. The general equation for MAE is:

$$MAE = \frac{\sum_{i=1}^N |p_i - q_i|}{N} \quad (4.1)$$

The lower the MAE the more accurately the recommendation system predicts user ratings.

- *Quality accuracy metrics* the purpose of this metrics is to measures the number of correctly predicted votes against the test data set, *recall* is one example of this metrics. we can compute the *recall* of the recommended system as:

$$recall = \frac{Number\_of\_hits}{n} \quad (4.2)$$

where *Number\_of\_hits* represents the number of correctly predicted vote, and *n* represent the total number of recommendation made. A recall value of 1.0 indicates that the algorithm was able to always predict the correct

rating, whereas a recall value of 0.0 indicates that the algorithm was unable to correctly recommend any of the ratings from the test set.

## 4.4 System Configuration

The Collaborative filtering code used here has been implemented in Matlab. We run the test on a Pentium 4 2.80 GHz computer with 512M RAM. Tests were run on Matlab Version 7.01 on Microsoft Windows XP Professional.

## 4.5 Implementation

The code used in this experiment were developed using Matlab and is available in Appendix B. All the algorithm used in the code has been explained in the previous chapters, we will also refer to the exact formula when discussing the algorithm.

The purpose of this experiment is to compare the two similarity measures, either the Pearson correlation or Vector similarity, and determine which will perform better in predicting ratings based on the **MovieLens** data set. We have implemented both algorithms in Matlab.

The Pearson correlation (`pcorrel`) method uses the Equation 2.3 mentioned in section 2.1.2. Some other methods were also created so that it can be reuse in other algorithms. Some of these methods include, `meanVote` which is a method to calculate the average vote for a particular user (Equation 2.1), and the method `sArray` which is used to create the list of item that any 2 users have voted before. This method will take in any two users and compute the similarity between them using Pearson correlation and return a float value as the output.

The Vector similarity (`vSim`) method uses the Equation 2.4 which is also mentioned in section 2.1.2. In this method, there is no need to calculate a mean vote for a particular user, therefore the `meanVote` method was not used, however the `sArray` method was still used in this method. Similar with the `pcorrel` method, this method will take in any two users and compute the similarity between them using the `Vector similarity` and return a float value as the output.

Both similarity methods, either `pcorrel` or `vSim` can be used in the Collaborative filtering algorithm to predict a rating that the active user have not voted before. The `predictWeight` method is a memory based collaborative algorithm that is used to predict a single movie rating for the active user. The method input variable are the active user and one movie item that the rating would like

to be predicted. The main algorithm used in this method uses Equation 2.2 in section 2.1.2. The output from this method is a prediction of a single movie rating for the active user. We can use this method repetitively to predict a large number of users and items.

For the experiment we ran the `predictWeight` method using both the `pcorrel` and `vSim` method. The test set contain movie item with the actual rating provided by the user. We input the user and the movie item and use the `predictWeight` function to predict the rating, we then compare the predicted rating with the actual rating. By comparing the output with the actual result, we can calculate the **Mean Absolute Error (MAE)** and `recall` as discussed in section 4.3. The method `evalCF` does this computation automatically.

Finally all the methods that have been created and discussed in this section can be found at the appendix B.

## 4.6 Methodology

Given the data set provided from MovieLens, we need to split the data set as mentioned in section 4.2. In Matlab environment, we will read in the appropriate data sets, and acquire the training and testing data sets.

In the initial step of the algorithm, three parameters need to be supplied by the user.

1. The number of user or the number of lines that the algorithm will process to predict the rating.
2. The type of similarity algorithm that will be used in the algorithm. The two types of similarity algorithm that can be selected are either Pearson correlation or Vector similarity.
3. The training and testing data set for the algorithm. There are 7 pairs of training and testing set in total in our experiment, 5 pairs according to the 80%/20% split(u1-u5), 2 pairs base on the 10 rating per user (ua, ub)

The developed code made used of both the prediction and similarity measures formula discuss on section 2.1.2. After the algorithm has been run, it will return a value range from 1-5 to represent the rating predicted.

For the experiment in order to find out the performance of the similarity measures, we uses all the 7 pairs set available. For the first type of pairs set,

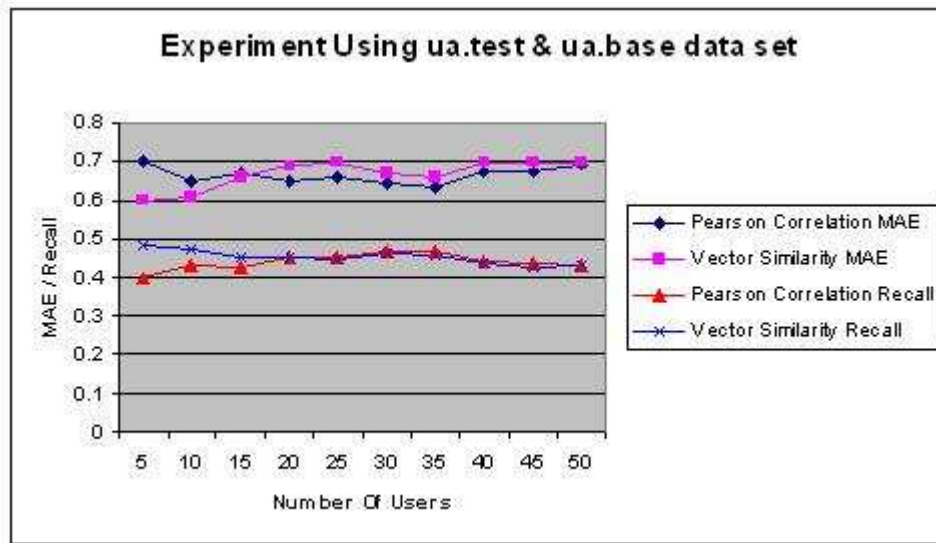


Figure 4.1: MAE and recall graph using ua.base and ua.test data set

which include (u1.base, u1.test) - (u5.base, u5.test), the number of lines read into the system is the input. In this case, the number of lines read represent the number of ratings need to be predicted.

For the second type of pair set which is ua.base, ua.test, ub.base and ub.test, the number of users will determine how many prediction the code will made. For this test set, each user will have 10 actual ratings. Which means the code need to predict 10 ratings for each user. Therefore if the number of user input is set to 50, the code will predict 500 ratings.

## 4.7 Experimental Results

In this section we present our experimental results on the two similarity measures used by memory based collaborative filtering for generating predictions on Movielens data set. In assessing the quality of the similarity measures, two matrices are used, MAE and recall. A lower value in MAE indicates a better result, whereas a higher value in recall represent a better result.

As we can see from the graph, both Pearson correlation and Vector similarity have similar performance on most the MovieLens data set. We can only observe that Pearson correlation perform better than Vector similarity when using the ua.base and ua.test data set. While Vector similarity perform better than Pearson

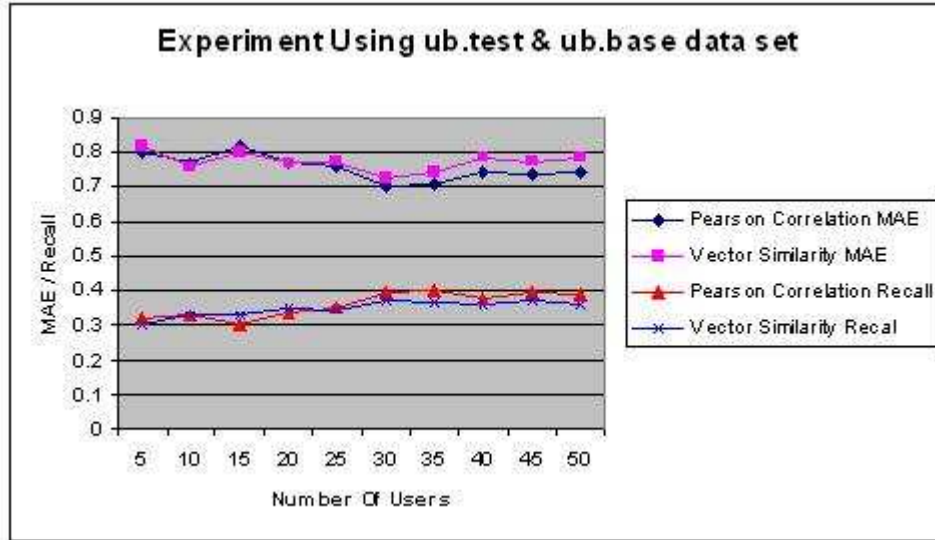


Figure 4.2: MAE and recall graph using ub.base and ub.test data set

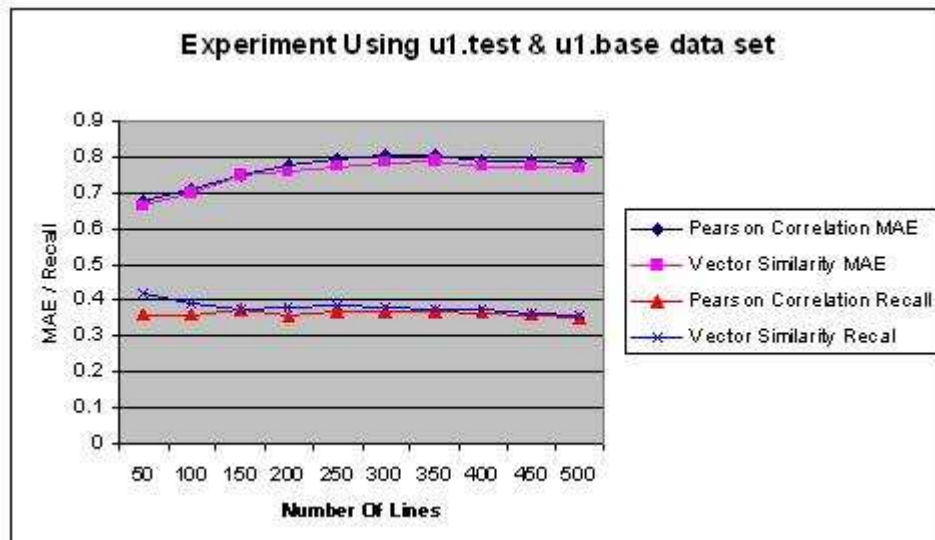


Figure 4.3: MAE and recall graph using u1.base and u1.test data set

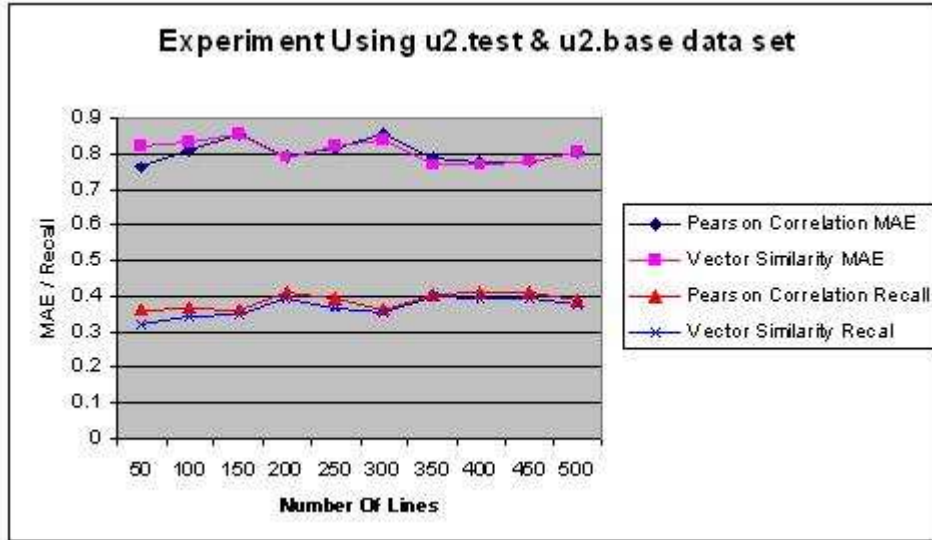


Figure 4.4: MAE and recall graph using u2.base and u2.test data set

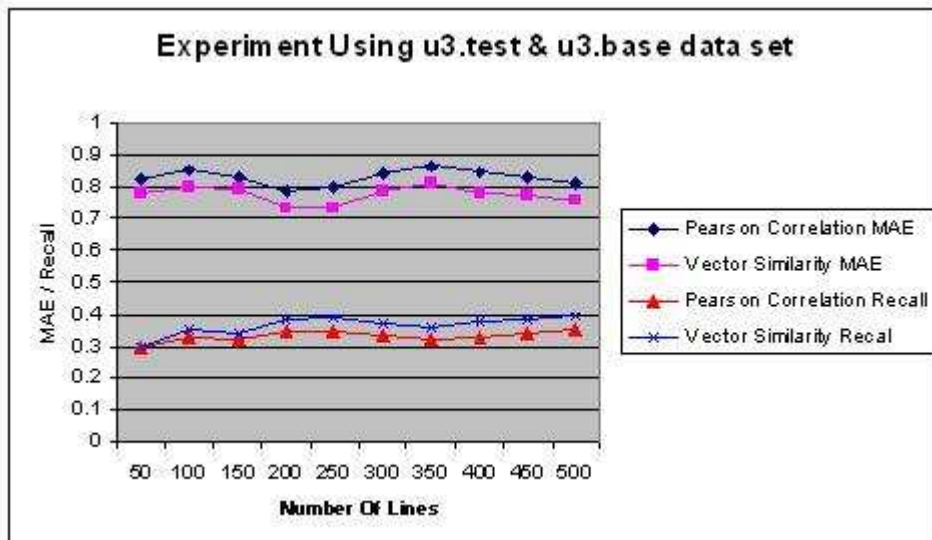


Figure 4.5: MAE and recall graph using u3.base and u3.test data set

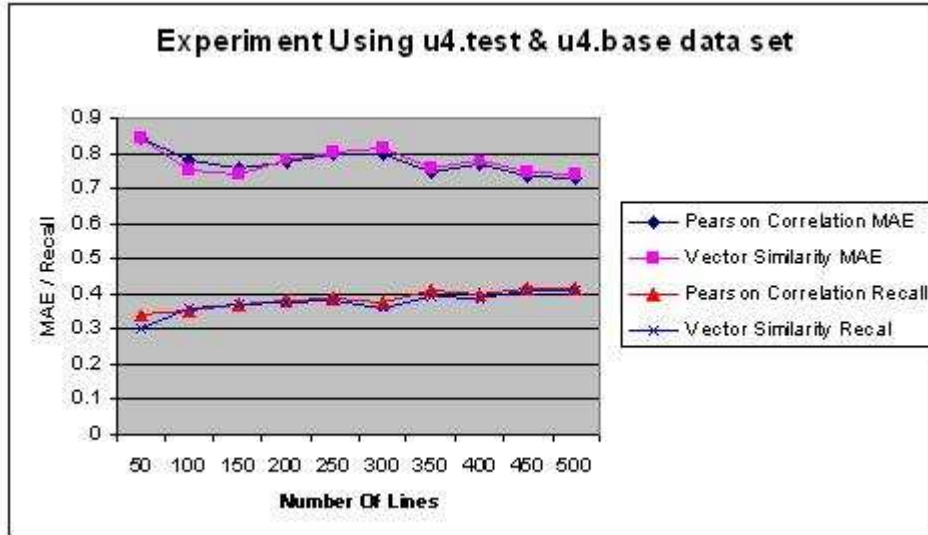


Figure 4.6: MAE and recall graph using u4.base and u4.test data set

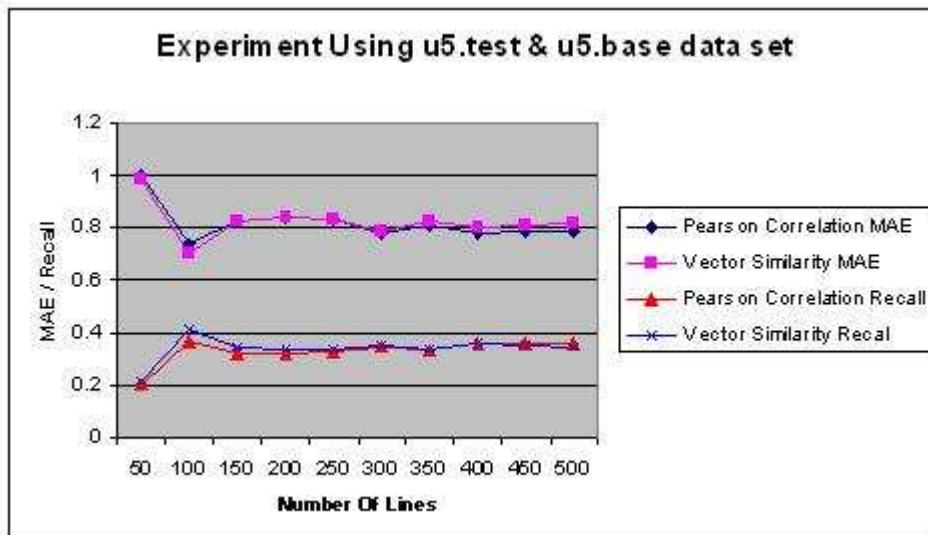


Figure 4.7: MAE and recall graph using u5.base and u5.test data set

Dataset	Standard Deviation
ua.base	1.2225
ub.base	1.2232
u1.base	1.2087
u2.base	1.2155
u3.base	1.2204
u4.base	1.2157
u5.base	1.2171

Table 4.1: Standard deviation on different dataset

correlation when using the u3.base and u3.test data set. But overall, there is no trend where the Pearson correlation outperforms vector similarity or vice versa. The result obtained in this experiment actually corresponds to another result that Sarwar et al obtained from their similar experiment [12].

Even though from this experiment we found out that there is really no much difference between the similarity measures, but there might be some cases where one would perform better than the other, and this case might be the way the user voted for different items. Pearson correlation put less weight on the rating close to the mean overall rating, therefore it may perform better when the data set have higher variation. Variation in the dataset can be calculated by using standard deviation. Table 4.7 present the standard deviation on the different dataset we used. The Pearson correlation equation state that, each of the rating will be subtracted from the mean vote of the user. When there is no variation in the voting (which usually means that the rating it self is about the same as the mean vote), one part of the equation will be equivalent to zero and result the output to be zero as well. When correlation between the two users is zero, it will render the collaborative filtering useless.

## CHAPTER 5

# Experimental Evaluation on Multidimensional Recommender System

## 5.1 Aim

The aim of this experiment is to run the multidimensional recommender system on a data set containing ratings in more than one dimensions, and observe the result when applying the recommender system to the different dimensions of the data set. We can also then use this experiment to explain the framework we propose for Web service selection outline in the next chapter.

## 5.2 Data set

For this experiment we cannot use the same Movielens data set. We need another data set that provide more dimensions as compare to the Movielens data set. Due to there were no freely distributed multidimensional data set available, we have to create our own multidimensional data set.

To create the multi dimensional data set, we use data gathered from an e-commerce Web site Epinions.com. Epinions.com provides varieties of products and enables the customer to rate items available on the Web site. For this experiment we concentrated on one particular type of items, which is digital camera. We have chosen this particular item because, beside the usual product rating dimension, the digital camera provided other dimensions as well. The other dimensions include *ease of use*, *durability*, *battery life*, *photo quality* and *shutter speed*. A user who want to rate an item, have to give opinions for all the dimensions using a scale from 1-5 (dislike-like).

Each camera type has a HTML file that contain all the ratings. We used both shell scripting and Java programming to extract the information needed from the HTML document. To extract the data, we first need to parse the HTML

document for detecting sentences containing the various dimensions. This was done using the Linux shell command *grep*. After the parsing is done, all the sentences detected containing the various dimensions will be saved to an output file.

The next step is to process the output file to construct the data set. A Java method is created to execute this step, the purpose of this method is to read the output file (produce from the previous step) and do more text processing to extract the value of each rating dimension, and then write it to a file in a way so that it represent a database structure.

The process above are repeated for different types of digital cameras. Due to time restriction for this project, we have only extracted ratings from 100 different types of digital cameras. The data set contain 1469 users who have rated at least one of the 100 available cameras. Even though the data set seem large enough for this experiment, however we need to remove all the users who have only rated for one item. A user who voted for one item could not be used in a multidimensional recommender system. Due to the removal of the users, the data set is left with only 56 users. This dramatically reduce the size of the data set. We will observe the effect of this data set when running the experiment later on.

### 5.3 Evaluation Metrics

The evaluation metrics used in this experiment, were the same as the one used in the previous experiment discussed in section 4.3.

### 5.4 System Configuration

The System Configuration used in this experiment is the same as previous experiment. Refer to section 4.4 for more details.

### 5.5 Implementation

For this experiment, the reduction based multidimensional recommender system is used. As explained in section 3.2.1 the reduction based approach make used of any techniques worked for two dimensional recommender system. Therefore we can use the Collaborative filtering recommender algorithm that have been developed previously in Chapter 4.

However, the code need to be modified so that it can be used in this experiment. The modification made are mostly for the way the system read in the data sets as the input. This modification are made so that all the dimensional ratings can be read into the system.

## 5.6 Methodology

First we need to extract a test set from the data set. This is done by selecting one set of ratings from each user. The remainder of the ratings that are not selected for test set become the training set.

For this experiment, we are going to apply the Collaborative recommender system on different dimensions that the data set have, including an extra dimension which is the average rating from **ease of use**, **durability**, **battery life**, **photo quality** and **shutter speed**. Again the setup for the Collaborative recommender system will be the same as previous experiment discussed in section 4.6. One difference for this experiment is, due to the small size of the data set, all the ratings inside the test and training set will be used.

## 5.7 Experimental Results

	MAE	Recall
Product Rating Dimension	0.9107	0.4107
Ease of Use Dimension	0.7321	0.5
Durability Dimension	0.6429	0.4286
Battery Dimension	1.1429	0.25
Photo Quality Dimension	0.8929	0.3036
Shutter Speed Dimension	1.25	0.2679
Average	0.8429	0.0357

Table 5.1: Result of Collaborative recommender system run on various dimension

The purpose of running this experiment is not only to determine the performance of multidimensional recommender system, but also on showing how the multidimensional recommender system can be used on product selection. Consider a scenario where a user need recommendation for buying a digital camera. Depending on the user preference, user might prefer some functionality over the others. An example, a user might prefer a camera with a better battery life

rather than its shutter speed. Multidimensional recommender system does not restrict recommendation based on product dimension alone, it provide the user with other dimensions that are available, which might give a more accurate recommendation to the user. From the result of this experiment, we can see that some dimension have perform better than the usual product rating. Based on the experiment's result, if a user wanted a digital camera recommendation based on the durability of the digital camera, the system would recommend item more accurately. A crucial point to note, the result obtained in this experiment might not be very accurate, due to the sparsity of the data. The data set used for this experiment were also not large enough compare to the one used in the previous experiment.

## CHAPTER 6

# Web Service Selection Using Collaborative Filtering Framework

So far we have discussed recommender system for product selection, which can be extended for Web service selection. In this chapter we will propose a Web service selection framework using multidimensional collaborative recommender system.

## 6.1 Web Service Selection Framework

The current infrastructure of Web services has no support on the implementation of Web service selection. Therefore based on the experiences of using multidimensional collaborative recommender system for product selection, we propose to extend the current infrastructure of Web service to support multidimensional collaborative Web service recommendation. The framework itself consist of several components, which include Web service Directory Facilitator agent, User registry agent, Quality of service ontology agent (QoS), ranking registry agent and Web service selection agent. Figure 6.1 present an overview of the framework.

### 6.1.1 Web Service Directory Facilitator Agent

A directory facilitator agent is needed to play the same role as a UDDI registry to allow Web service provider to register their services. Same as normal Web services, Web service providers register their services to the new repository agent, the agent will then maintain all registered services available, and Web service consumer can look up services in the new registry. The extra work the new directory facilitator agents need to do is to communicate with the ranking agent and the QoS agent, which will be explain later on in this chapter.

### Web Service Selection Framework

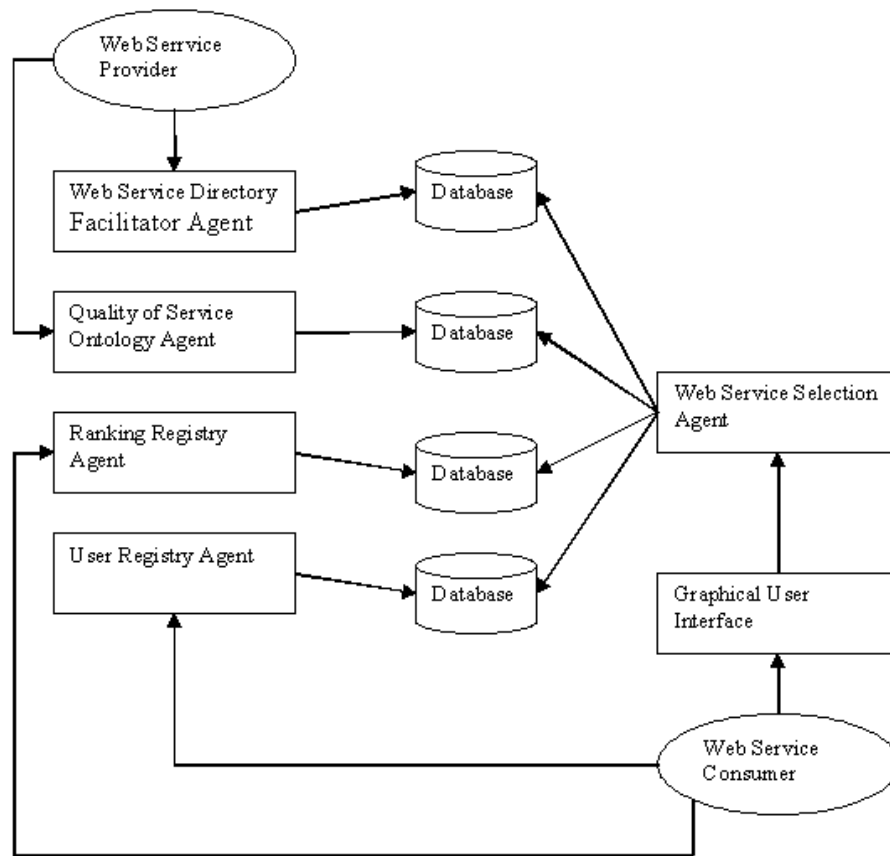


Figure 6.1: The Web Service Selection Framework

### 6.1.2 User Registry Agent

Mechanism for collecting information about users are very important in building a collaborative recommender system's dataset. For example, the MovieLens dataset. Therefore, to build Web service recommender system, user's registering is equally important. The user registry agent in this framework, allow users/consumers to register with the system, and only when registered the user can then rate Web service they have used before. The multidimensional service recommendation system can use the user profiles/preferences to make personalized recommendation.

### 6.1.3 Quality of Service Ontology Agent

The purpose of the Quality of Service Ontology(QoS) agent is to provide attributes/ dimensions for the collaborative filtering to work on. For an example, in the multidimensional recommender system experiment using the digital camera data set, there are five dimensions that the recommender system can work on. This attributes include dimensions such as photo quality, shutter speed and battery life. For Web service selection framework, the Web service's quality of service attributes can be of various dimensions, either objective or subjective, these quality of service attributes are discussed in section 2.2.2. This agent will be working closely together with the Web Service Directory Facilitator Agent. When a service provider register their services to the directory, the Web service will automatically have the basic QoS attributes defined. In addition to that, the service provider are allowed to register any additional quality of service attributes the service might have to the the QoS agent. The QoS agent will then save all attributes associated with the Web service. The attributes will then be used for rating purposes by the user. The ratings will can then be used by the recommender system to provide recommendations to other users.

### 6.1.4 Ranking Registry Agent

The ranking registry agent provide storing mechanism for all the ratings on Web service ranked by the users. This agent provided profile that represent the user preferences on Web service they like. The agent also provide user profile that are similar to the active user, so that it can be used by the recommender system to provide recommendation to the active user. Therefore the ranking registry will be an important component for the Web service selection framework.

### 6.1.5 Web Service Selection agent

The last component of the framework is Web service selection agent. The purpose of this agent is to use multidimensional recommendation technique to recommend services to the consumer. As mention in Chapter 5, the reduction based multidimensional recommender system, work by reducing the multidimensional problem into a two dimensional environment. The system then apply collaborative filtering technique to the two dimensional data to obtain recommendation. The new Web service agent will act similarly to those in product selection. Instead of using product rating and it's quality attributes, the agent uses service rating and Web service's Quality of Service attributes to perform recommendation to the service consumer.

## 6.2 An Example: Selecting a Web Service

In this case study, we will explain the process on how the new framework work on finding a specific Web service. First, the Web service provider would register their Web services to the new directory facilitator agent. Similarly with UDDI, the agent will store information about the Web service and how it can be accessed. When the service provider register their services, they may also register any additional quality of service attributes to the QoSO agent. After the service has been registered to the system, other user or consumer are now able to rate the Web service, and all the rating will be saved in the ranking registry agent. But before any user can rate services, they first need to register with the user registry agent.

When a service consumer search for a specific Web service, the Web service selection agent will be used. After the request made by the consumer, the Web service selection agent will query the directory facilitator agent. Where the directory facilitator agent will perform the service discovery method and return a list of services matching the search criteria. Service selection is then done to the list of available service by using multidimensional collaborative recommender system.

Before service selection is done, the consumer need to define the criteria for the selection. As a simple example, the consumer criteria might be, the Web service reliability must be excellent. The Web service selection agent will then make a selection based on this criteria. The multidimensional recommender system reduce the multidimensional data to **service** and **reliability** dimension only. The recommender system then will return the best prediction from the list of result to the consumer.

The consumer should also be allowed to select more criteria for service selection. When selecting more criteria for service selection, the consumer should prioritize the criteria. The recommender system would then perform reduction based on the prioritized list of criteria. A more complex example in this case, the customer criteria might be, the Service rating must be excellent, the performance and reliability of the Web service it self must be exceptional as well. Therefore this criteria actually involve **servicerating**, **performance** and **reliability** environment. To perform this criteria, the multidimensional recommenders system must first reduce the problem into a two dimension first according to the prioritize list. Therefore the reduction are done in the following step: first (**service**, **servicerating**), second (**service**,**performance**) and lastly (**service**, **reliability**) dimensions. When the system reduces to **service** and **servicerating** dimension. The system does not actually recommend any service at this stage, the result of selection on the **service** and **servicerating** dimension, will then be reduce again to the **service** and **performance** dimension. The reduction is done for **service** and **reliability** dimension as well after the previous reduction has been complete. Finally the system will return the best prediction from all the criteria to the user. An important thing to note is that, at any stage after the first reduction, when the reduction does not return any matching services the system should return the best prediction made on the last reduction.

### 6.3 Comparison With Other Framework

The Web service selection framework we proposed have some similarities and differences to the one that is proposed by Singh and Sreenath [10]. The similarities for both our framework is that we uses collaborative filtering technique for the recommender system. In addition, both our framework make used of the rating provided by the service consumer to make recommendation for other consumer. Even though there are some similarity with out framework, there are however major differences. Most importantly our framework support multidimensional rating for recommendation. The multidimensional rating which compose of Web service Quality attributes, provide the recommender system with more information to work with to produce more personalized recommendation to the consumer. Other functionality our framework provide include user registration, where registered user has the rights to rate Web services.

## CHAPTER 7

# Conclusion and Future Work

## 7.1 Conclusion

The current infrastructure of Web service addresses the service discovery problem, but not service selection. Web service discovery deals with finding Web services implementations that meet some specified description. While Web Service selection deals with choosing a service implementation from among those that are discovered for the given description. If we treat Web services as special type of products from the Web service providers, existing technique used for product selection can also be applied to Web service selection. Therefore understanding how these techniques work on product selection is important in the research for creating a Web service selection framework.

In this dissertation, we have conducted experiments on performing both two dimensional and multidimensional recommendation using collaborative filtering algorithms. On the two dimensional recommender system experiment, we tested two type of similarity measures, Pearson correlation and Vector similarity on the MovieLens dataset. The experiment result suggest that both similarity measures have similar performance based on the MovieLens data set. Therefore, both similarity measures can be used in the multidimensional collaborative recommender system.

The second experiment we conducted was to demonstrate the way multidimensional recommender system work on product selection. This experiment was conducted using the dataset gathered from the Epinion.com Web site. By experimenting with multidimensional recommendation on product selection, we proposed a framework for Web Service selection using multidimensional recommender system.

The Web service selection framework we proposed consists of five components, they are Web service directory facilitator agent, User registry agent, Quality of service ontology agent, ranking registry agent and Web service selection agent. The Web service directory facilitator agent enable service provider to register

their services, user registry agent handles user registration to enable them to rate services, quality of service ontology agent provide services attributes that user can rate on. While ranking registry store rating of Web services, and Web service selection agent perform service selection using multidimensional recommender system on the web service ratings.

## 7.2 Future Work

The Web service selection framework discussed in this dissertation, only considered as centralized system. On the other hand, the current infrastructure of Web service works on a distributed environment. What we mean by distributed is, for example there are more than one UDDI registry where service provider can publish their service to. Therefore there are some issue need to be addressed for the distributed Web service selection framework.

One issue need to be addressed in our framework, is when a service consumer register their service to one of the directory facilitator agent, other directory facilitator agent must be informed of this operation. The other directory agent need to be informed because the other agent need to add the service provider and update their directory as well. Therefore there are a lot of communication need to be done between directory facilitator agent.

## APPENDIX A

# Original Honours Proposal

**Title:** Personalized Collaborative Filtering for Web Service Selection

**Author:** Kenneth Hein Karta

**Supervisor:** Dr. Wei Liu

## Background

Web services are a new type of Web application. It can be published, located and invoked across the Web. Web services can perform functions anything from a simple request to a complicated process.

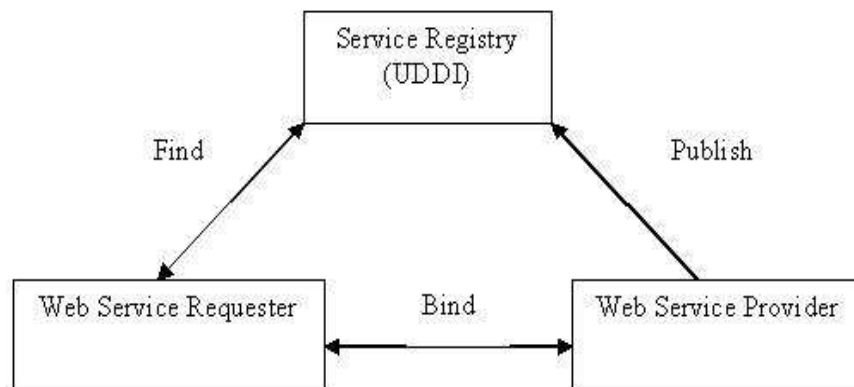


Figure A.1: Web Service Architecture

Figure 1 describes the current structure of Web services. Web services providers use Web Service Description Language (WSDL) to describe the service that are provided. The service providers publish their services to a public service

registry using Universal, Description, Discovery and Integration (UDDI).Service consumer discover service in the registry and obtain a URL for the WSDL file that describe the service. The Service then can be invoked using the XML based Simple Object Access Protocol (SOAP) [2].

The current Web service architecture addresses the problem of service discovery. Service discovery deals with finding a set of services that match the specific description. On the other hand, Service Selection involves in selecting a service from the set of services return by service discovery. This can be thought of selecting the best service from the set. In more general term, service selection maps a set of services to a ranking of the services in that set [10]. This project will focus on techniques that enable services selection based on Quality of Service comparison.

## Aim

The aim of this project is to adapt and extend Collaborative Filtering to be used in Web Service selection. There are several task need to be addressed to achieve this aim:

- Designing Web Services Quality of Service Ontology
- Reviewing existing Collaborative Filtering Algorithm.
- Applying and extending the algorithm to service selection based on the QoS Ontology.

To measure the effectiveness of the developed algorithm, this project will also compare the Collaborative Filtering service selection with other approaches

## Method

To begin the project, a thorough understanding is needed for the OWL Ontology Language to represent Quality of Service Ontology and Matlab for implementing the collaborative filtering algorithms.

The next step is to design a Web service Quality of Service Ontology using the OWL language. This QoS ontology is basically providing the non-functional attributes of the services. The QoS values of the service will be used by Collaborative filtering for service selection criteria.

After Designing the QoS ontology, the project will develop the Collaborative Filtering algorithm using Matlab. The goal of Collaborative filtering is to suggest new service or to predict the utility of a certain service for a particular user based on previous likings and opinions of similar minded users. In Collaborative Filtering scenario, there is a list of user and list of services available. Each user maintains a list of rankings indicating their opinions about the services. The opinions can be in the form of rating score within a certain numerical range. The Collaborative Filtering algorithm can return service likeliness that can be of two forms [12]:

- Prediction: is a numerical value expressing the predicted likeliness of a service for the active user.
- Recommendation: is a list of services that the active user may like most.

Collaborative filtering algorithms are available for single dimensional data. The project needs to modify and extend these algorithm so that they will work on the QoS ontology.

The last step is to conduct experiment to test how the Collaborative Filtering Selection performs compare with other approaches. For this project, the Ontology will be based on the MovieLens [7] movie database, where it would be extended to a multi-dimensional database to simulate the Web service Quality of Service.

## Software and Hardware Requirements

- Computer With Microsoft Windows XP or Linux
- Programming Language: OWL and Matlab

## Project Time Estimate

<b>Task</b>	<b>Duration</b>
Project Proposal	3 Weeks (February 28 2005 - March 17 2005)
Project Proposal Due	Thursday 17 March
Literature Review	8 Weeks (March 18 2005 – May 15 2005)
Revised Project Proposal Due	Thursday 26 May
Application Design	2 Weeks (May 16 2005 - May 29 2005)
Implementation	9 Weeks (May 30 2005 – 31 Jul 2005)
Testing	2 Weeks (August 1 2005 - 14 August 2005)
Result Analysis	2 Weeks (August 15 2005 - 28 August 2005)
Report and Presentation Preparation	8 Weeks (29 August 2005 - 20 October 2005)
Draft Dissertation Due	Thursday 15 September
Seminar Title and Abstract Due	Thursday 6 October
Final Dissertation Due	Thursday 20 October
Poster Due	Thursday 27 October
Demo and Presentation	1 Weeks (25 October 2005 - 28 October 2005)
Seminar Presented to Seminar Marking Panel	25 October – 28 October

## APPENDIX B

# Matlab Code for Two dimensional Recommender System

# Bibliography

- [1] ADOMAVICIUS, G., SANKARANARAYANAN, R., SEN, S., AND TUZHILIN, A. Incorporating contextual information in recommender system using a multidimensional approach. *ACM Transactions on Information System* 23, 1 (January 2005), 103–145.
- [2] A.MENASCE, D. Quality issues in web services. *IEEE Internet Computing* (November 2002).
- [3] BALABANOVIC, M., AND SHOHAM, Y. Combining content-based and collaborative recommendation. *Communications of the ACM* 40, 3 (March 1997).
- [4] BOOTH, D., HAAS, H., MCCABE, F., NEWCOMER, E., CHAMPION, M., FERRIS, C., AND ORCHARD, D. Web services architecture. [Online] <http://www.w3.org/TR/ws-arch/>, February 2004.
- [5] BREESE, J. S., HECKERMAN, D., AND KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. *In Proceeding of the 14th Conference on Uncertainty in Artificial Intelligence* (May 1998), 43–52.
- [6] CLAYPOOL, M., GOKHALE, A., MIRANDA, T., MURNIKOV, P., NETES, D., AND SARTIN, M. Combining content-based and collaborative filters in an online newspaper. *In Proceedings of ACM SIGIR Workshop on Recommender Systems, August 19 1999*. (August 1999).
- [7] GROUPLENS. MovieLens. [Online] <http://www.cs.umn.edu/Research/GroupLens/>, April 2005.
- [8] MAXIMILIEN, E. M., AND P. SINGH, M. A framework and ontology for dynamic web services selection. *IEEE Internet Computing* 8, 5 (September 2004), 84–93.
- [9] MAXIMILIEN, E. M., AND P. SINGH, M. Toward autonomic web services trust and selection. *Proceedings of the 2nd International Conference on Service Oriented Computing (ICSOC)* (November 2004).
- [10] P. SINGH, M., AND M. SREENATH, R. Agent-based service selection. *Journal on Web Semantics (JWS)* 1, 3 (April 2004), 261–279.

- [11] RESNICK, P., LACOBOW, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. GroupLens: An open architecture for collaborative filtering of netnews. *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work* (1994), 175–186.
- [12] SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. Item-based collaborative filtering recommendation algorithms. *WWW10* (2001).
- [13] W3C. Qos for web services: Requirements and possible approaches. [Online] <http://www.w3c.or.kr/kr-office/TR/2003/ws-qos/>, November 2003.
- [14] ZAN, H., HSINCHUN, C., AND ZENG, D. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information System* 22, 1 (January 2004), 116–142.