Location Aware Multi-Criteria Recommender System for Intelligent Data Mining

by

Salvador Valencia Rodríguez

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MCS degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

One of the most important challenges facing us today is to personalize services based on user preferences. In order to achieve this objective, the design of Recommender Systems (RSs), which are systems designed to aid the users through different decision-making processes by providing recommendations to them, have been an active area of research. RSs may produce personalized and non-personalized recommendations. Non-personalized RSs provide general suggestions to a user, based on the number of times an item has been selected in the past. Personalized RSs, on the other hand, aim to predict the most suitable items for a specific user, based on the user's preferences and constraints. The latter are the focus of this thesis.

While Recommender Systems have been successful in many domains, a number of challenges remain. For example, most implementations consider only single criteria ratings, and consequently are unable to identify why a user prefers an item over others. Many systems classify the user into one single group or cluster which is an unrealistic approach, since in real world users share commonalities in different degrees with diverse types of users. Others require a large amount of previously gathered data about users' interactions and preferences, in order to be successfully applied.

In this study, we introduce a methodology for the creation of Personalized Multi Criteria Context Aware Recommender Systems that aims to overcome these shortcomings. Our methodology incorporates the user's current context information, and techniques from the Multiple Criteria Decision Analysis (MCDA) field of study to analyze and model the user preferences. To this end, we create a multi criteria user preference model to assess the utility of each item for a specific user, to then recommend the items with the highest utility. The criteria considered when creating the user preference model are the user's location, mobility level and user profile. The latter is obtained by considering the user specific needs, and generalizing the user data from a large scale demographic database.

We present a case study where we applied our methodology into PeRS, a personal Recommender System to recommend events that will take place within the Ottawa/Gatineau Region. Furthermore, we conduct an offline experiment performed to evaluate our methodology, as implemented in our case study. From the experimental results we conclude that our RS is capable to accurately narrow down, and identify, the groups from a demographic database where a user may belong, and subsequently generate highly accurate recommendation lists of items that match with his/her preferences. This means that the system has the ability to understand and typify the user.

Moreover, the results show that the obtained system accuracy doesn't depend on the user profile. Therefore, the system is potentially capable to produce equally accurate recommendations for a wide range of the population.

Acknowledgements

Above all, I would like to thank my supervisor, Dr. Herna L. Viktor, for her constant and valuable help, guidance and support, not only in my thesis project but during my entire graduate studies. I am particularly thankful with her for believe and encourage me since the beginning of this study. Her advices, recommendations, and excellent supervision made every chapter of this thesis possible. Additionally I want to thank Craig Statchuk from IBM Canada for his valuable feedback and the fruitful discussions we had.

I especially want to thank my wife Katia Preciado Caballero, who is also my partner and best friend in every project of my life. Without her constant and unconditional love, support, help, and understanding, this thesis would have never been possible. Thank you for being part of my life and always being there for me.

I would also like to thank my parents, Salvador Valencia Garcia and Mireya Rodríguez Marínez, for their teachings and support during all my life. Their life attitude of effort and perseverance has always been a source of inspiration and strength. I am very proud of being their son. Thank you for being my life teachers and role models.

Last but not least, I want to thank my sister, Mireya Valencia Rodríguez. Thank you for your contagious happiness and for being the best sister anyone could have.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1
Introduction

Currently, businesses and enterprises own massive databases that contain vast amounts of data. This data has been increasing with the use of the World Wide Web, that makes it easier to automatically collect new data and record it directly into databases. Increasingly companies are facing the problem that these vast collections of data have become "meaningless" to their owners. The data have exceeded our human ability for comprehension without powerful data analysis tools. This is due to the fact that these databases often contain Terabytes of information, stored in hundreds of interconnected tables. This implies that it's impossible for a person to infer patterns, find correlations between the data, or consider all the options included in the database in order to make good decisions when an opportunity is presented. Consequently a number of data mining and statistical analysis techniques have been proposed to discover unknown, useful, novel and non-trivial information from our large repositories. We can say that our data repositories have become "data tombs", but with the help of data mining and statistical tools we can convert them into "golden nuggets" of knowledge [1].

On the other hand, during the previous decades, we have witnessed the increased use of the Internet and development of new technologies, which have lead to the creation of web-based systems that are accessible for most of the population anytime and anywhere. Consequently, nowadays the use of Internet and more specifically these web-based systems, play a critical role in most of our daily activities and aids us to make decisions. For instance, if we want to buy a book we no longer go to a bookstore, instead we visit Amazon's web site. Similarly we see movies using Netflix, and share our feelings, thoughts and ideas with our friends using Facebook, Google+ or Twitter.

As a result from the large amount of information currently available in the databases, and the critical role that web-based systems play in our lives, Recommender Systems (RSs) are proposed as

software tools and techniques designed to provide suggestions for items to be of use to a user. These suggestions aim to aid the users in different decision-making processes, such as what items to buy, what music to listen, and so forth.

In this work we propose a methodology to create RSs that create multi criteria user preference models considering both, who the users are (i.e. personal information and preferences) and their current context (i.e. current position and mobility level), in order to recommend items for a user.

In this Chapter we present the motivations and goals of our study, and present how the subsequent chapters are organized.

1.1 Motivation and Goals

In our lives we usually find ourselves in situations where we have to decide between items. For instance, select which movie to see, which book to read, to which gym register in, and so forth. However, in today's world, due the advance in technology, the number of options available to choose from has dramatically increased, reaching a point where we have hundreds and even thousands of options at our finger tips through the use of web-based systems provided by companies. Therefore, the problem we are currently facing is that this massive amount of information and items offered, that is now available to everybody through the use of internet, gives us a large number of possibilities and choices, making it very difficult to decide the best one for us. We face this problem everyday when we have to decide which movie to see out of thousands offered by Netflix, or which book to buy when Amazon offers us an unlimited amount of choices. Consequently, in order to address these problems, the new trend of system development is the creation of Recommender Systems, that intend to provide personalized services to each user, showing them only the information that they are interested into. Companies are currently using Recommender Systems to interact with their customers to point them towards new, not-yet experienced items, which may be relevant to the user, based on the person's likes, preferences and behaviors. According to Ricci, et al. [2], Recommender Systems have proven to be valuable means for online users to cope with the information overload and have become one of the most powerful and popular tools in electronic commerce.

Recommender systems, in order to recommend an item to a user, have to first analyze and model the user, and subsequently select out from thousands of options and features the best items for him/her. Therefore in this sense RSs may be seen as data mining user front ends, where the user can browse the recommended items resulting from applying one or several data analysis tools and techniques. To do so, the system mines the data looking for patterns and correlations between the user's learned profile and the available items' information, in order to create intelligent models to

predict the most suitable items for each user. Correspondingly, the development of Recommender Systems involves a multi-disciplinary effort from various fields of study such as Artificial Intelligence, Human Computer Interaction, Information Technology, Data Mining, Statistics, Adaptive User Interfaces, and Decision Support Systems [2].

Consequently, a number of methods, algorithms and techniques to predict the best items for a user have been proposed (i.e. RS's implementations).While RSs have been successful in many domains, a number of challenges remain. For example, most implementations consider only single criteria ratings, and consequently are unable to identify why a user prefers an item over others; some classify the user into one single group or cluster which is an unrealistic approach, since in real world users share commonalities in different degrees with diverse types of users; and others require a large amount of previously gathered data about users interactions and preferences, in order to be successfully applied.

This thesis focuses on the creation of a methodology and algorithms to create RSs that overcome the above-mentioned limitations. Our methodology to create RSs, as presented in this study, builds user preference models considering multiple criteria, and therefore includes as part of the recommendation process the user preferences, special needs, behavior, context, and so forth. This provides the opportunity to the system to clearly identify what is more important for the user when selecting an item over the others, and correspondingly come with accurate recommendations that match his/her preference's value scale. Moreover, by including the user context as part of the user preference model and recommendation process, the system is able to produce different types of recommendations to the same user, depending on the context he/she is currently in. Additionally, our methodology exploits the information contained in a large-scale demographic database to generalize the user given information, by means of classifying the user into one or more demographic groups. This allows the system to leverage commonalities between similar user types, and create richer user profiles to generate recommendations without the need of previous stored data.

1.2 Thesis Outline

The following chapters of the present thesis are organized as follows: Chapter 2 provides a literature review on Recommender Systems. We present a general framework identified in every RS, along with its components and process followed to recommend items to the users. Subsequently we introduce a classification for RSs based on their recommender technique. We conclude this Chapter presenting a comparison between the presented types of RSs, stating their advantages and limitations. Moreover in Chapter 3, we introduce three different techniques to extend the capabilities for

traditional RSs. We present the UTA methods and context aware RSs, as solutions to consider multi-criteria and the user context during the recommendation process, respectively. We also introduce the demographic generalization technique, a method to generalize the user profile from a large-scale demographic database. We conclude this chapter, presenting fourteen different properties and metrics that can be measured in a RS in order to evaluate or compare it, along with three types of experiments that can be performed to measure the values for these properties. Subsequently in Chapter 4, we provide a detailed description for some implementations of RSs. We present a comparison between them, by means of their proposed methodology (i.e. recommender algorithms). We show their advantages and disadvantages in terms of their recommender capabilities, user required information and computational complexity. In Chapter 5, we present our proposed methodology and algorithms to create RSs that consider the user profile and context as criteria to model the user, in order to recommend items that best match with the created user preference models. In Chapter 6, we describe our case study, a Personal Recommender System (PeRS) that implements our proposed methodology to recommend events to a user. We provide a detail description of the datasets used and the software and hardware environment set up. We conclude the chapter presenting the experimental design and results obtained from the evaluation of our methodology, as implemented in our case study. Finally in Chapter 7, we conclude and summarize this thesis, presenting a discussion of the obtained results, the thesis contributions and some possible directions in which this thesis may be furthered.

Chapter 2
Introduction to Recommender Systems

Recall from Chapter 1 that Recommender Systems (RSs) are software tools and techniques that suggest items which could be helpful to a user [3] [4] [5]. In this sense, we can define RSs as software systems designed to assist users through various decision-making processes. In general, we will use the term "item" to denote what the system recommends to users (e.g. movies, books, etc.).

The development of RSs began with the observation that, in the real world, people rely heavily on the recommendations of others when making daily decisions [3] [6]. Consequently, the goal of RSs is to mimic this real world behavior by providing users with recommendations. RSs can produce both non-personalized and personalized recommendations. Non-personalized recommendations are based on the number of times a specific item has been selected in the past, thereby determining general recommendations for any user. Personalized recommendations, on the other hand, aim to predict the most suitable items for a specific user, based on that user's preferences and constraints [2].

The recommendation problem is formulated as follows: Let *C* represent the set of all users, and *S* represent the set of all possible items that can be recommended. Both spaces *S* of possible items and *C* of possible users may be very large, ranging to hundreds of thousands or even millions. Let *u* denote a utility function that measures the usefulness of item *s* to user *c*, defined as $u : C \times S \to R$, where *R* is a totally ordered set of possible utility values. Thus for each user $c \in C$, we aim to recommend the item $s' \in S$ that maximizes the user's utility [7]. More formally:

$$\forall c \in C, s'_c = \max_{s \in S} u(c, s) \qquad (2.1)$$

The utility of an item to a user is typically represented by a rating indicating how much a particular user prefers a particular item. However, the utility is not usually defined on the whole $C \times S$ space, but either on a subset of it (e.g. items previously rated by the users), or not defined at all. Therefore, the RS must extrapolate (if some items are rated), or calculate this utility for every instance within the space $C \times S$. RS engines should be able to estimate (predict) the ratings of the non-rated item/user combinations, and issue appropriate recommendations based on these predictions [7]. Each user $u \in C$ is defined by a user profile, which includes personal information (e.g. age, gender, income, marital status, etc.), depending on the items to be recommended. In the same way, each item $s \in S$ is defined by a set of characteristics or attributes.

The new ratings of the not-yet-rated items can be calculated using a number of different methods from machine learning, approximation theory and heuristics. These methods consider only the user profile and item characteristics to estimate the missing ratings. Extrapolations from known to unknown ratings are usually done either by using heuristics that define the utility function and empirically validate it's performance, or by estimating the utility function that optimizes certain performance criteria (e.g. the mean square error). Once the unknown ratings are estimated, the items recommended to a particular user are those with the highest rating among all ratings for that user, according to equation (2.1). Alternatively, the system can recommend *N* best items to a user, or a set of users to an item [7].

Ratings can be expressed in a variety of forms, including numerical (e.g. 1 to 5 stars), ordinal (e.g. strongly agree, agree, neutral, disagree, strongly disagree), binary (e.g. the user is asked to decide if a certain item is good or bad), or unary (e.g. indicates only that a user has observed or purchased an item) [8].

Interest in RSs has dramatically increased in recent years, to where they now play an important role in some of the most important websites, such as Amazon, YouTube, Yahoo, Google, Netflix and others [2]. RSs are usually classified according to the recommendation algorithm they use to estimate the utility of an item. In the next Section, we present a general framework for RSs, as well as a classification based on the algorithms they apply.

2.1 General Framework for Recommender Systems

The RS's design, graphical user interface and recommendation technique to be used should be decided based on the specific type of items the system will recommend, and the available sources of information. The information sources a RS requires are mainly related to the users and the items. However, some recommendation techniques may require other types of information, such as context

data, interests, behaviors, individual characteristics of the users and so forth. In some cases, selecting the recommendation technique to be used may also require consideration of other factors, including the type of target users, the devices they would use, and the role of the recommendation within the application [2].

Regardless, we can identify a general model or framework for all types of RSs using four key dimensions: system users, information about the users, characteristics of the items to be recommended, and the recommender engine (algorithms and techniques) used.

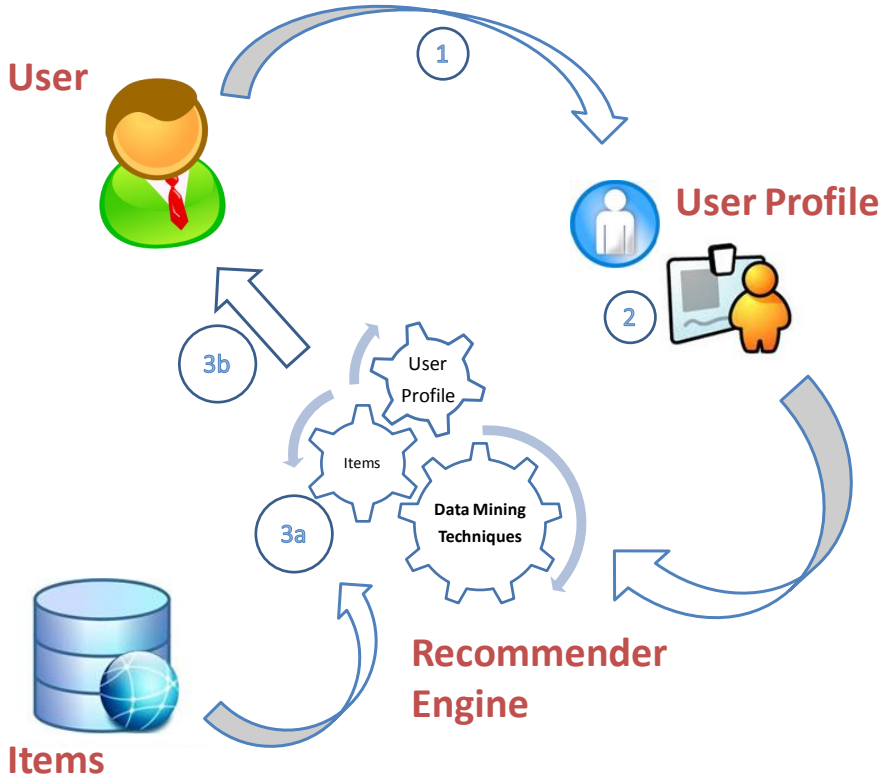


Figure 2.1: General framework for RS

Figure 2.1 shows the general model for all types of RSs. From this model we can identify the following four general tasks performed by any RS: first, obtain user information; second, create the user profile; third, assess the items' utility by applying a recommendation technique or algorithm, considering the user profile and the characteristics of the items to be recommended; and fourth, present the items with either the highest utility, or a utility higher than a selected threshold. These phases are explained in detail in the following Sections.

2.1.1 Gathering User Information

The first phase in RSs involves obtaining information from the user and storing it as transactions, in order to build the user profile. In this sense we define transactions as recorded interactions between a user and the system. The easiest way to get the required information would be to ask the user for it directly. However, this approach has several drawbacks, as explained below. To overcome these disadvantages a second methodology, which implicitly gathers the user information from the observation of his/her actions, is proposed. Additionally, the information gathered by the system may also include feedback from the user, thus allowing the RS to improve the created user profile, and consequently produce more accurate recommendations.

Explicit information. Directly asking the user what we need to know is the easiest way to obtain the required information. Types of information gathered in this way include individual characteristics (e.g. users' age, gender, job, birth date, marital status, hobbies), or personal interests. This information can be collected through interviews or questionnaires, or by using systemized rating systems, which allow the users to indicate if they like, or dislike, what they are looking at. The main disadvantages of this approach are that users generally don't want to complete long questionnaires or forms, they may not tell the truth, and sometimes they are unsure how to express their interests, or what they really want [9].

Observation of user actions (implicit information). This technique is intended to overcome the previously mentioned disadvantages presented when gathering explicit information. This method obtains the required data by observing user actions to discover patterns, using Machine Learning or Data Mining techniques. The results of applying these techniques (e.g. patterns, categories, clusters) must be different for different users to be considered suitable for building a user profile. The main advantage of using this method to gather the required information is that it automatically updates the user profile; as more actions are recorded, more accurate and better recommendations are produced [9].

User feedback. Feedback from the user is the only way a system can self-evaluate its predictions. Depending on this evaluation, in some cases it may be necessary to obtain additional information about a user, in order to produce more accurate recommendations in the future. The feedback can also be gathered directly from the user, or by evaluating the user actions after the system has assisted him/her [9].

2.1.2 Create the User Profile

In general, we will use the term "user" to refer to the person who is seeking a suggestion or recommendation. When creating the user profile it is important to keep in mind that everyone is unique, with different goals and preferences; the items that one person considers interesting or important, others might not. Due to this user diversity, a RS must gather information about each specific user. Depending on the recommendation technique being used, this information could be as simple as a list of ratings provided by the user, or very complex, with socio-demographic attributes such as genre, age, profession, education and lifestyle information.

Thus, the concept of User Profiling is present, as a way to successfully understand the user [10], in order to provide him/her with personalized services and recommended items. We define personalization as *"...the ability to provide content and services that are tailored to individuals based on knowledge about their preferences and behavior."* [11], and we define a User Profile as a description of a user that contains the most important or interesting facts about him/her. The user profiles allow us to analyze and discover how users differ from one other, which is vital for providing them with personalized services.

User data is said to constitute the user model [12] [13] that profiles the user. Hence, a RS can be viewed as a tool that generates recommendations by building and exploiting user models [14] [15] or profiles. The user profile plays a critical role in any RS, since it will be used by the system to identify similar users, and to ultimately recommend items to them that match their likes, goals, preferences and constraints. The kind of information in a user profile varies, depending on the goal of the application and the items being recommended. However, the most common content in user profiles for RSs is interests, behaviors, individual characteristics, and context.

Interests. This is one of the most important aspects of the user profile for information-driven systems such as RSs. Interests can represent news, web pages, documents, or work-related or hobbies-related topics. According to Schiaffino et al. [9], the most common representation for user interests are Keyword-based models, in which interests are denoted by weighted vectors of keywords, and the weights represent the relevance of the word for that specific user within a specific topic. An example of these models is *Term Frequency/Inverse Document Frequency* (TF-IDF), which uses vectors of weights for each word to predict the interesting documents for a user. The weights are obtained by comparing the word frequency in a specific document against the word frequency in all the documents (see Section 2.1.3 for more details on this method). Another approach to represent user interests is through topic tree hierarchies, in which each tree node represents an interesting topic for a user, defined by representative words. This approach is particularly useful

when it is important to model not only general user interests, but sub-topics of these interests as well [9].

Behavior. Including user behavior in the user profile allows RSs to look for repetitive conduct and discover patterns, which can then be used to assist the user according to the learnt behavior. Xu et al. [16] proposes storing behavior information as a set of $< t, e >$ pairs, where e is a behaviour of the user, and t a point in time (or an interval of time) when the behaviour occurred. Once a user's behaviour information is stored, data mining techniques can be applied to discover the user's behaviour patterns. These patterns will later be used by the recommendation technique to determine the final recommended items. The type of behavior modeled (behaviors that are stored) and the time intervals depend on the application domain. Discovered patterns may be regular, or seasonal (i.e. observed only during specific time periods) [9].

Individual characteristics. This refers to personal information, another important aspect of a user profile, and could include demographic and user personality data [9]. Demographic data might be information such as gender, age, marital status, city, country, number of children, number of cars, income, and more. This demographic information can also be obtained from large scale demographic databases, as discussed in Section 3.3. Personality information is generally obtained through observation and analysis of user responses to specific situations. The responses are gathered by monitoring a user's facial expressions or gestures when the system presents him/her with a particular situation. For more information on methods to determine user personality, we recommend [17].

Contextual information. The user profile could also include information about the user's situation, such as: **Environmental context**, which is information about the entities that surround the user; **Personal context**, which refers to data about the physiological and mental health of the user (e.g. pulse, blood pressure, weight, glucose level, hair color, mood, expertise, anger, stress); S**ocial context**, describing the social aspects of the user, including information about friends, neighbors, co-workers and relatives; and, **Spatio-temporal context**, which describes aspects such as time, location or direction. This type of information is used mainly by context-aware systems that provide services and information based on a user's situation, condition or context [9]. More information on context-aware systems is presented in Section 3.2.

2.1.3 Recommender Engines

In order for a RS to predict (or suggest) an item, it initially needs to calculate the utility or worth of all the items; or at least be able to compare the items to decide which to recommend to a user. However, depending on the number of items to be assessed, this process could involve high

computational complexity. Consequently, some RS first apply heuristic methods to determine the set of items that a user might like, then calculate the utility of only this reduced set. These heuristic predictions use the information about users, items and transactions. It is important to note that a user's utility of an item sometimes depends not only on the item characteristics and the user profile, but also on the context or other information. For example, the user may be more interested in items closer to his/her current location [2].

RSs can be used in a wide range of applications and domains, addressing many different tasks. Consequently, there are a large number of different techniques that can be applied. Balabanovic et al. [18] presents the following three categories to classify RSs, based on the recommendation algorithm they use: **Content-based recommendations** recommend items similar to those the user preferred in the past; **Collaborative recommendations** recommend items that similar people had preferred in the past; and **Hybrid approaches** combine both the collaborative and content-based methods to recommend items to a user.

Content-Based Methods

Content-based RSs recommend items that are similar to those the user preferred in the past. The similarity of the items is calculated according to the characteristics associated with the compared items [2]. More formally, the utility $u(c,s)$ of item s for user c is estimated based on the utilities $u(c,s_i)$ assigned by user c to items $s_i \in S$ that are 'similar' to item s [7].

Content-based systems are mainly focused and designed to recommend text-based items, such as documents and websites (URLs). Thus, in this type of RSs, the characteristics of items (item profiles) described as *Content(s)* for item $s \in S$ are normally represented by vectors of weighted keywords. Within these vectors, the importance of word k_i in document d_j is determined by weight $w_{i,j}$, which can be calculated using different measures [7]. One of the best-known measures to obtain these keyword weights is *Term Frequency/Inverse Document Frequency* (TF-IDF), as presented in [19] and defined as follows: Let N be the number of documents that may be recommended where keyword k_i appears in n_i of them, and let $f_{i,j}$ be the number of times that k_i appears in document d_j. Then the term frequency of keyword k_i in document d_j is defined as:

$$TF_{i,j} = \frac{f_{i,j}}{\max_z f_{z,j}} \qquad (2.2)$$

Here, the maximum is calculated for all keywords k_z that appear in document d_j. However, since the keywords that appear in many documents cannot help determine if a specific document is more relevant than others, the measure of *Inverse Document Frequency* (IDF_i) is used in combination with $TF_{i,j}$. The IDF_i for keyword k_i is defined as follows:

$$IDF_i = \log \frac{N}{n_i} \qquad (2.3)$$

Consequently, the final TF-IDF weight ($w_{i,j}$) for keyword k_i in document d_j is defined as:

$$w_{i,j} = TF_{i,j} \times IDF_i \qquad (2.4)$$

and the content of document d_j is defined as $\boldsymbol{Content(d_j) = (w_{1,j}, \ldots, w_{k,j})}$.

Once the item profile (*Content*(d_j)) has been obtained, a user profile that can be used to select the best documents, based on keyword weights, needs to be calculated by analyzing the items previously rated by the user. This keyword weights user profile is obtained using keyword analysis techniques, as follows. Let *ContentBasedProfile(c)* be the profile of user $c \in C$, then *ContentBasedProfile(c)* is defined as a weight vector of keywords $\boldsymbol{ContentBasedProfile(c) = (w_{c,1}, \ldots, w_{c,k})}$, where each weight $w_{c,i}$ represents the importance of keyword k_i to user c [7]. Rocchio et al. [20] presented the Rocchio algorithm, which computes the *ContentBasedProfile(c)* as an average vector from individual user content vectors, while Pazzani et al. [21] proposed using the Naïve Bayesian classifier (presented below) to estimate the probability that a specific user may prefer a specific document.

Once we have obtained both user (*ContentBasedProfile(c)*) and item (*Content(s)*) profiles, represented as TF-IDF vectors $\overrightarrow{w_c}$ and $\overrightarrow{w_s}$ of keywords weights, the RS can determine the utility of each item for each user ($\boldsymbol{u(c,s)}$). This utility is represented by a heuristic distance function defined in terms of the vectors [7]. Several distance functions have been proposed to measure the distance between two vectors. According to Muhivuwomunda et al. [22], three of the most representative and widely used distance metrics are the Euclidean, Manhattan, and Cosine Similarity measures. Euclidean distance, defined in equation (2.5), represents the shortest distance between two points, and when used as a distance measure it describes the distance between two string vectors [23]. Manhattan distance (also known as city block distance) is defined in equation (2.6), and graphically represents the shortest path, in a city designed in square blocks, to go from one point to another [24].

$$Euclidean(s,t) = \sqrt{\sum_x (s(x) - t(x))^2} \quad (2.5)$$

$$Manhattan(s,t) = \sum_x |s(x) - t(x)| \quad (2.6)$$

Cosine similarity measure, on the other hand, is defined as follows, where K is the total number of keywords [25] [19]:

$$u(c,s) = score(ContentBasedProfile(c), Content(s)) \quad (2.7)$$

$$u(c,s) = \cos(\overrightarrow{w_c}, \overrightarrow{w_s}) = \frac{\overrightarrow{w_c} \bullet \overrightarrow{w_s}}{\|\overrightarrow{w_c}\|_2 \times \|\overrightarrow{w_s}\|_2} = \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2} \sqrt{\sum_{i=1}^{K} w_{i,s}^2}} \quad (2.8)$$

In addition to the methods that predict utility based only on heuristics measures, other types of RSs calculate utilities based on models learned from the data, using statistical and data mining techniques (e.g. clustering and Naïve Bayesian data mining techniques). Naïve Bayesian classifiers, presented by Pazzani et al. [21], estimate the probability that a document or web page p_j belongs to a certain class C_i (relevant or irrelevant), given the set of keywords $k_{1,j}, \ldots, k_{n,j}$ on that page.

$$P(C_i \mid k_{1,j} \,\&\, \ldots \,\&\, k_{n,j}) \quad (2.9)$$

Since a Bayesian classifier considers all keywords independent, this probability is proportional to the following equation:

$$P(C_i) \prod_x P(k_{x,j} \mid C_i) \quad (2.10)$$

For each page p_j, the probability $P(C_i \mid k_{1,j} \& \ldots \& k_{n,j})$ is computed for each class C_i, assigning page p_j to the class C_i with the highest probability.

Content-based RSs have the following limitations:

Limited content analysis. RSs using this approach are limited to considering only the characteristics associated with the items to be recommended. Therefore, the system requires a large number of features per item to obtain accurate and relevant recommendations. However, only some

features can be automatically parsed from the item information by the system, while the rest need to be assigned manually. Due to a limitation of resources, this is not a practical approach [26]. Another problem is that if two different items have the same set of characteristics (same weighted vector of keywords), it is impossible to distinguish one from the other.

Over-specialization. RSs only recommend items to a user with a high utility, according to the user profile. Therefore, the recommendations that a user receives are limited to items similar to the ones the user has already rated in the past. This problem is often addressed by introducing some randomness into the user profile. On the other hand, it is also important to consider that in some cases, the system shouldn't recommend items that are *too similar* (highest utility) to others that the user has already seen (rated). This is in order to provide diversity to the recommendations, which is a desired feature in any RS. The user will expect to receive a range of options and not only homogeneous set of items [18] [26].

New user problem. Since RSs of this type, based their recommendations on the user profile; it follows that in order for a user to receive reliable recommendations, the user has to have already rated a large number of items [18] [26].

Collaborative Methods

RSs that use collaborative methods recommend items that other people with similar tastes preferred in the past. The similarity in taste of two users is calculated based on the similarity of their rating history [2]. More formally, the utility of an item for a user $u(c,s)$ is estimated based on the utilities $u(c_j,s)$ given to item s by similar users $c_j \in C$ to user c [7].

According to Breese et al. [27], collaborative methods can be further classified into memory-based (heuristic-based) and model-based.

Memory-Based Methods

Memory-based methods are heuristic techniques that estimate the utility of an item based on the entire collection of items previously rated by the users. Thus, the values of unknown ratings $r_{c,s}$ for user c and item s are obtained by applying an aggregate function over the ratings of the N most similar users for the same item s [7], as follows:

$$r_{c,s} = aggr_{c' \in \hat{C}} r_{c',s} \qquad (2.11)$$

Where $\hat{C}$ represents the N most similar users to user c who have rated item s. There are several aggregation functions (*aggr*) that can be used to compute the unknown ratings $r_{c,s}$. For example, Breese et al. [27] and Delgado et al. [28] proposed the following:

$$\text{(a)} \quad r_{c,s} = \frac{1}{N}\sum_{c' \in \hat{C}} r_{c',s} \tag{2.12}$$

$$\text{(b)} \quad r_{c,s} = k\sum_{c' \in \hat{C}} sim(c,c') \times r_{c',s}$$

$$\text{(c)} \quad r_{c,s} = \overline{r_c} + k\sum_{c' \in \hat{C}} sim(c,c') \times (r_{c',s} - \overline{r_{c'}})$$

where k represents a normalizing factor, usually selected as $k = 1/\sum_{c' \in \hat{C}} |sim(c,c')|$, and $\overline{r_c}$ represents the average rating of user c, defined as follows:

$$\overline{r_c} = \left(\frac{1}{|S_c|}\right)\sum_{s \in S_c} r_{c,s} \text{ , where } S_c = \{s \in S \mid r_{c,s} \neq \varnothing\} \tag{2.13}$$

Here, $r_{c,s} = \emptyset$ indicates that item s has not been rated by user c. The aggregation function can be a simple average as shown in equation (2.12.a) [27], or a weighted sum as in equation (2.12.b). However, equation (2.12.b) does not take into account that different users could use different rating scales; some might use a 1 to 5 rating scale, while others use a 1 to 10 scale. Equation (2.12.c) [27] [28] overcomes this problem by presenting an adjusted weighted sum, which uses derivations from the average rating of the corresponding user, rather than absolute values of ratings [7].

The similarity measure between user c and c' (i.e. $sim(c,c')$), is a distance measure used as a weight. Since the user is compared with every other user, the more similar the users are the more weight ratings $r_{c',s}$ will be considered in the prediction of $r_{c,s}$. The two most widely used similarity measures for these types of methods are correlation-based and cosine-based (see equation (2.8)). For both of these measures, if S_{xy} is the set of all items rated by both users x and y, then $S_{xy} = \{s \in S | r_{x,s} \neq \emptyset \ \& \ r_{y,s} \neq \emptyset\}$. The Pearson correlation coefficient is used to measure the similarity in the correlation-based approach, and is defined as follows [29] [26]:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} \left(r_{x,s} - \overline{r_x}\right)(r_{y,s} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{x,s} - \overline{r_x})^2 \sum_{s \in S_{xy}} (r_{y,s} - \overline{r_y})^2}} \qquad (2.14)$$

On the other hand, the cosine-based approach [28] [30], considers the two users *x* and *y* as two vectors in m-dimensional space, where $m=|S_{xy}|$. Therefore, the similarity between these users is measured by the cosine of the angle between them:

$$sim(x,y) = \cos(\vec{x},\vec{y}) = \frac{\vec{x} \bullet \vec{y}}{\|\vec{x}\|_2 \times \|\vec{y}\|_2} = \frac{\sum_{s \in S_{xy}} r_{x,s} r_{y,s}}{\sqrt{\sum_{s \in S_{xy}} r_{x,s}^2} \sqrt{\sum_{s \in S_{xy}} r_{y,s}^2}} \qquad (2.15)$$

As seen in equations (2.8) and (2.15), both content-based and collaborative memory-based approaches use the same cosine measure. However, in content-based it is used to measure the similarity between vectors of TF-IDF weights, while in collaborative memory-based it measures the similarity between vectors of user-specified ratings.

Model-Based Methods

Model-based methods, as opposed to memory-based methods, learn the model that will be used to make predictions from the user's ratings. Many models have been proposed. For example, Breese et al. [27] recommended an approach in which the unknown ratings are estimated as follows:

$$r_{c,s} = E\left(r_{c,s}\right) = \sum_{i=0}^{n} i \times \Pr(r_{c,s} = i | r_{c,s'}, s' \in S_c) \qquad (2.16)$$

The rating values are integers between *0* and *n*, and the probability function represents the probability that user *c* will rate item *s* with a rating $r_{c,s}$, considering the ratings of items previously rated by user *c*. This probability can be obtained by using data mining clustering techniques or Bayesian networks. In cluster models, "similar" users are clustered into classes where the user ratings are considered independent within the same class, and the number of classes and parameters are learned from the data. Conversely, Bayesian networks represent each item as a node, where the states of each node match the possible rating values for each item. In Bayesian networks, both structure and probabilities are learned from the data. However, the main limitation of both approaches is that each user is classified into a single cluster, limiting the domain of possible recommended items for each user.

Alternatively, Sarwar et al. [30] proposed a method to predict the rating for an item *i* for a user *u* by computing the sum of the approximated ratings for the user ($R'_{u,N}$) to items similar to *i*. The weighted sum is scaled by the sum of the similarity terms, to ensure that the predicted ratings will be within the same range. The weighted sum is calculated as follows:

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N}(S_{i,N} * R'_{u,N})}{\sum_{all\ similar\ items,N}(|S_{i,N}|)} \qquad (2.17)$$

Approximated ratings are obtained with a linear regression model, which is calculated using the vectors of ratings of the target item *i* (R_i), and ratings for the similar item *N* (R_N). Thus the linear regression model, where α and β are obtained from both rating vectors and ϵ is the error of the regression model, is expressed as follows:

$$R'_N = \alpha R_i + \beta + \epsilon \qquad (2.18)$$

Another methodology is presented by Billsus et al. [31], who proposed a collaborative filtering within the machine learning framework. This technique models the collaborative approach of finding similar users as a classification task, to create a classification model for each user. The classification model can be used later to classify unrated items in "like" or "dislike" categories. In order to reduce the sparcity problem, which occurs when a user has only a few previously rated items, a feature extraction technique is first applied over the set of user ratings.

Since collaborative-based RSs use other users' ratings, these systems can be used for other types of items, including different items than those already rated by the user [7]. However, these kinds of systems have the following limitations [18] [32].

New user problem. Collaborative-based methods have the same problem as previously described for content-based methods: in order to provide accurate recommendations, the system must first learn about user's preferences.

New item problem. Since collaborative RSs estimate the unknown ratings based only on users' previous ratings, in order to recommend a new item be added, the system requires that a large number of users rate it first.

Sparsity. In general, for all RSs the number of already rated items is very small compared to the total number of items in the system, and this leads to inaccurate recommendations. Therefore, the

success of these systems depends on the availability of a large number of users rating items. However, users with unusual preferences compared to other users will always receive poor recommendations. One option to overcome this problem is to use demographic information when calculating the user similarities. This approach is presented in more detail in Section 3.3.

Hybrid Methods

Hybrid methods use a combination of the techniques already mentioned, in order to overcome the weaknesses of one with the strengths of another. For example, though collaborative filtering methods cannot recommend items that have no ratings, this can be accomplished using a content-based approach, since predicting for new items is based only on the item descriptions [2]. In general, the following four methods are used to combine these approaches: First, implementing collaborative and content-based techniques separately and then combine their predictions; second, incorporating some content-based characteristics into a collaborative approach; third, incorporating some collaborative characteristics into a content-based approach; and fourth, constructing a general unifying model that incorporates both content-based and collaborative characteristics. These different techniques are presented and defined by Adomavicius et al. [7] as follows:

Combining separate recommenders. This technique requires implementing both approaches separately, then either combining the estimated ratings into one final recommendation using a linear combination or a voting scheme, or using only the estimated rating with the highest selected quality criterion. One quality criterion that can be used is selecting ratings that are more consistent with the user's past ratings.

Adding content-based characteristics to collaborative models. This technique is based on collaborative models, but it uses content-based users' profiles, rather than commonly rated items, to obtain the similarity between users. The approach is intended to overcome the previously described problem caused by a lack of people who have previously rated the same items as the user.

Adding collaborative characteristics to content-based models. This approach applies a dimensionality reduction technique over a group of content-based users' profiles, thereby creating a single unified collaborative view of a user profile that can be applied in place of the individual ones.

Developing a single unifying recommendation model. Ansari et al. [33], proposed a technique to estimate unknown ratings r_{ij} for user i and item j that uses items and users' profile information as input to the following statistical model:

$$r_{ij} = x_{ij}\mu + z_i\gamma_j + w_j\lambda_i + e_{ij} \ , \quad e_{ij} \sim N(0,\sigma^2) \ , \ \lambda_i \sim N(0,\Lambda) \ , \ \gamma_j \sim N(0,\Gamma) \qquad (2.19)$$

where e_{ij}, λ_i and γ_j are random variables that consider noise, unobserved sources of heterogeneity, and item heterogeneity respectively. x_{ij} is a matrix containing users and characteristics, z_i is a vector of user characteristics, and w_j is a vector of item characteristics. Finally, μ, σ^2, Λ and Γ are estimated from already known ratings, using *Markov Chain Monte Carlo* (MCMC) methods. To summarize, this model uses user attributes $\{z_i\}$ as part of user profiles, item attributes $\{w_j\}$ as part of item profiles, and their interaction $\{x_{ij}\}$ to estimate the rating of an item for a user. Monte Carlo methods are algorithms designed to sample Markov chain-based probability distributions. The state of the chain after a large number of steps is used as a sample of the desired distribution. A Markov chain is a mathematical system developed to create statistical models of real world processes. It models random transitions from one state to another within a finite number of states. The model is characterized as memory less, since the next state depends only on the current state, not on the sequence of states previously visited. For more information on Monte Carlo methods and Markov Chain models, we recommend [34] [35].

Once the items' utilities are estimated using one of the previously outlined techniques, the final step for a RS is to present the highest utility items to the user. Depending on the domain of the application and the number of items to be recommended, in some cases the system might present those items with a utility higher than a previously selected threshold.

Comparison between Types of Recommender Systems

RSs based on their recommendation technique can be classified into the three categories mentioned above: content-based RSs, collaborative-based RSs and hybrid RSs. Each of these recommends items to the user by applying different algorithms and techniques over different types of input data, and each has different limitations. Consequently, the RS type to be used for a specific application would depend on the domain of the application and the information available. Table 2.1 shows a comparative summary of these three types of RSs.

Content-based RSs recommend items similar to those the user preferred in the past, and thus reduce the recommendation problem to find similar items based on their information and attributes. Conversely, collaborative-based RSs recommend to the user items that other people with similar tastes preferred in the past, reducing the recommendation problem to find similar users based on their user profile.

Type	Summary	Suitable for	Limitations
Content-based	Recommend items similar to those the user preferred in the past. Recommendation problem is reduced to find similar items.	Applications with a large number of information for each item (e.g. text-based items)	**Limited content analysis.** Require a large number of features per item. **Over-specialization.** Can only recommend items similar to those the user has already given a high rating. **New user problem.** Requires the user to rate a large number of items
Collaborative-based	Recommend to the user the items that other people with similar tastes preferred in the past. Recommendation problem is reduced to find similar users.	**Memory-based** Applications with rich user profiles and either a small number of items, or a high rate of newly added users. **Model-based** Applications with rich user profiles and either a large number of items, or a low rate of newly added users.	**New user problem.** Requires the user to rate a large number of items. **New item problem.** To recommend an item, requires that a large number of users have already rated it. **Sparsity.** Requires a large number of users providing ratings to items.
Hybrid	Seeks to combine content and collaborative based methods using any of the following techniques: **Combining separate recommenders**, **adding content-based characteristics to collaborative models**, **adding collaborative characteristics to content-based methods**, or **developing a single unifying recommendation model**.		

Table 2.1: Comparative summary of RS types

Due the data they require to predict item's utilities, content-based RSs have the following limitations: they require a large number of features per item (i.e. limited content analysis); they can only recommend items similar to those the user has already given a high rating (i.e. over-specialization); and, they require that the user has already rated a large number of items (i.e. new user problem). Meanwhile, collaborative-based RSs have limitations as well: they also require that the user has already rated a large number of items (i.e. new user problem); they require that a large number of users have already rated an item in order to recommend it (i.e. new item problem); and, their success depends on the availability of a large number of users rating items (i.e. sparsity). Consequently, while content-based RSs are more suitable for applications with a large amount of information for each item (e.g. text-based items), collaborative-based RSs are more appropriate for applications in which user profiles contain a large amount of data.

Collaborative-based RSs can be sub-classified as memory-based or model based, depending on how they predict the item utility for a user once similar users have been identified. Memory-based

methods estimate the utility of an item using the entire collection of previous ratings from all similar users for each item. Alternatively, model-based methods learn a model from the similar users' ratings, which is then used to estimate the item utility. Therefore, selecting the appropriate collaborative type of RSs depends on the number of items and similar users, and the degree to which new users and items are being added. Since, to be able to predict the utility of an item, model-based methods need to initially create a model, these methods are preferred for applications with a large number of item utilities to be assessed, and in which the rate of newly added users is low. Conversely, memory-based methods can predict an item utility without the previous creation of a model; however, a higher computational complexity is required to estimate each item utility. Thus, these methods are more suitable for applications where the number of items requiring estimates of their utility is small, or the rate of newly added users is high.

Finally, hybrid RSs seek to combine content and collaborative-based methods to overcome the weaknesses of one with the strengths of another. From the literature, the four possible techniques to combine content and collaborative-based methods are the following: First, implement collaborative and content-based methods separately, then combine their predictions (i.e. combining separate recommenders); second, use collaborative methods to obtain the similar users, not based on their user profile but on commonly rated items (i.e. adding content-based characteristics to collaborative models); third, apply a dimensionality reduction technique over content-based users' profiles (i.e. adding collaborative characteristics to content-based methods); and fourth, construct a general unifying model that incorporates both content-based and collaborative characteristics (i.e. developing a single unifying recommendation model).

2.2 Summary

In this Chapter we provided an overview of Recommender Systems (RSs). We started by discussing a general framework present in all types of RSs, with system users, user profile, item profile, and recommender engine as the main components. We also presented a detailed description of the steps involved in this framework to recommend items to users: obtain user information, create the user profile, apply a recommendation technique or algorithm to assess the utility for each item, considering the user profile and the characteristics of the items to be recommended, and finally, recommend the items with highest utility.

Additionally, we presented a classification for RSs based on their recommender technique: content-based, collaborative-based or hybrid. Content-based RSs recommend items similar to those the user preferred in the past, while collaborative-based RSs recommend items to the user that other

people with similar tastes preferred. Therefore, while the content-based techniques reduce the recommendation problem to find similar items based on their information and attributes, collaborative-based approached reduce it to find similar users based on their user profiles. Hybrid RSs, on the other hand, aim to combine content and collaborative-based methodologies by one of the following techniques: combine the recommendations produced by content and collaborative techniques, create a new unified recommendation model that considers both user and item similarities, add content-based characteristics to collaborative techniques, or add collaborative characteristics to content based techniques.

We concluded Chapter 2 with a comparison between the types of RSs, based on their limitations and their recommender engines. From the comparisons we concluded that, while content-based techniques are more suitable for applications with a large amount of information for each item, collaborative-based methodologies are more appropriate for applications where user profiles contain a large amount of data.

The types of RSs presented in this Chapter, consider only the user's preferences and items' information to produce recommendations. However, in many application domains, this information may not be enough to generate appropriate (or accurate) recommendations. The following Chapter presents techniques that extend the RSs capabilities to address this issue.

Chapter 3
Advanced Recommender Systems and Evaluation

The recommendation methods described in Chapter 2 have been proved to perform well in several simple applications, where user and item information are sufficient to recommend an item. However, these methods need to be extended in order to be applied to more complex applications, such as application domains without enough initial data to produce an accurate recommendation, or where the recommendations should also consider the user context information [7]. There are several ways that RSs can be extended to provide better recommendation capabilities.

Moreover, RSs are designed and built based on the specific application domain in which they are going to be applied. Consequently, the recommendation technique used, how the data is modeled, how the results are presented to the user, and the types of item and user information required to provide a recommendation, are all different from one RS to another. Thus, deciding which RS is the best to be applied in a system, or evaluating a new proposed RS, can be difficult tasks. For this reason, a number of experiments, techniques and system properties have been developed to measure RSs' performance.

This Chapter presents some proposed extensions for RSs, as well as fourteen different properties that can be measured to evaluate them, and three types of experiments that can be used to measure these properties.

In Section 3.1, we describe some recommender algorithms that consider multiple criteria in order to recommend items to the user. This is followed in Section 3.2 by different techniques to model and consider the user context when recommending an item. Section 3.3 presents a method to

generalize the user provided information from a large-scale demographic database. In Section 3.4, we briefly introduce different types of experiments that can be performed to compare or evaluate RSs, followed by a description of some properties under which RSs may be evaluated. Section 3.5 concludes the Chapter with a summary of the presented techniques, properties and experiments.

3.1 Multi-Criteria Rating

Most current RSs consider only single-criterion ratings, where users rate each item in general. Consequently, these RSs cannot identify the characteristics of the items the user considers most important when making a decision. Therefore, other recommender engines that consider multi-criteria ratings have been developed. In multi-criteria RSs in addition to the general ratings, the user provides rates for each different criterion (or characteristic) for each item. Thus, these types of RSs can better interpret user interests and preferences, leading to more accurate recommendations.

According to Adomavicius et al. [7], the solutions most widely used to consider multi-criteria for RSs are the following: One, find the Pareto optimal solution; two, applying a linear combination of multiple criteria, and reducing the problem to a single-criterion optimization problem; and three, consecutively optimizing one criterion at a time, converting an optimal solution to constraints and repeating the process for other criteria. The third approach requires multiple iterations over all the items. The Pareto optimal solution provides a minimal notion of efficiency by assigning weights to each criterion, to be consistent with the user general ratings, but it doesn't guarantee the best possible distribution of weights. However, the second technique, implemented using *UTilités Additives* (UTA) methods, always guarantees the best possible solution. UTA methods allow the creation of a user preference model that can later be used to assess the utility of each item for that specific user. Hence, this technique will be our focus, presenting the UTA methods in the following Section.

3.1.1 UTA Methods

When considering multiple-criteria in decision making processes, the basic problem is determining how the final decision should be made. *Multi-Criteria Decision Analysis* (MCDA) is a discipline that explicitly considers multiple-criteria in decision-making environments [36]. The goal of MCDA is to help decision-makers achieve more informed and better decisions, by modeling complex problems and considering multiple criteria. However, when considering more than one criterion, there is usually not a unique optimal solution to these types of problems. Consequently, the goal of attaining one optimal solution is replaced by achieving a set of non-dominated solutions. A property of a non-

dominated solution is that one criterion in it cannot be improved without sacrificing at least one other criterion. Therefore, for these problems it is necessary that the user provides his/her global preferences over a set of reference items (i.e. a weak preference order) in order to differentiate between the possible non-dominated solutions, and select the one that best meets the user preferences [37]. UTA methods are regression-based techniques from the MCDA field of study that adopt the aggregation disaggregation paradigm, as shown in Figure 3.1. In the traditional aggregation paradigm, the user preference model is known 'a priori', and is used to obtain his/her global preferences. This is, obtain the global items' preferences from known user criteria weights. On the other hand, the philosophy of preference disaggregation aims to infer preference models (criteria weights) from given global preferences [38].

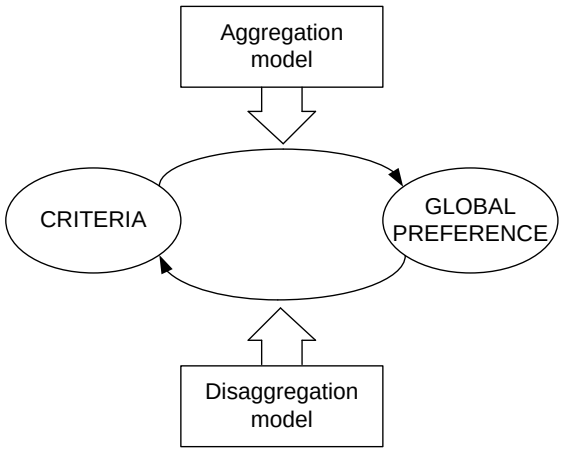


Figure 3.1: The aggregation and disaggregation paradigm in MCDA

The disaggregation-aggregation methodology proposed by Jacquet-Lagrèze et al. [39], is intended to analyze the behavior and cognitive style of the user. The goal is to assist the user by improving his/her understanding of his/her decision making process, leading to more informed and better decisions. This approach, depicted in Figure 3.2, consists of five levels or steps, identified and defined by Roy et al. [40] as follows:

Level 1 – Problem. The first level identifies the set of potential items or actions *A*, and determines the problem to be solved (i.e. the type of results we are trying to achieve), from one of four identified reference problem statements: One, choose one action from *A* (choice problem); two, sort the actions into pre-defined and preference-ordered categories (sorting problem); three, rank the actions from best to worst (ranking problem); and four, describe the actions in terms of their performance on the criteria (description problem) [40].

Level 2 – Criteria modeling. Model a family of criteria; that is, select the criteria under which each item will be evaluated, and establish the scales to evaluate each of them. The selected criteria should represent features or values that can be assessed for all the possible items to be recommended, and which are important for the user when making a decision. These criteria must be non-decreasing, exhaustive and non-redundant value functions defined on *A* as follows [40]:

$$g_i : A \rightarrow \left[g_{i*}, g_i^* \right] \subset \square \; / a \rightarrow g(a) \in \square \qquad (3.1)$$

$[g_{i*}, g_i^*]$ is the criterion evaluation scale, and g_{i*} and g_i^* are the lowest and highest value for the i-th criterion respectively. $g_i(a)$ is the evaluation or performance of action *a* on the i-th criterion, and $g(a)$ is the vector of performances of action *a* over all *n* criteria. From this definition, the following preferential situations are defined [40]:

$$\begin{cases} g_i(a) > g_i(b) \Leftrightarrow a \succ b \; (a \text{ is preferred to } b) \\ g_i(a) = g_i(b) \Leftrightarrow a \; \square \; b \; (a \text{ is indifferent to } b) \end{cases} \qquad (3.2)$$

Considering that the user will also provide a weak preference order on a reference set of actions (a general rating over all the actions) in the next step, the problem is reduced to adjusting additive value or utility functions based on multiple criteria. This adjustment is done seeking that the utility functions are as consistent as possible with the user provided weak preference structure [38].

Level 3 – Decision data. The user is required to provide his/her global preferences over a set of reference actions A_R, taking into account the performance of all the criteria for each action in the set. The set could be a set of past decision alternatives, a subset of decision actions when *A* is large ($A_R \subset A$), or a set of fictitious actions [38].

Levels 4 and 5 – Preference model construction and consistency of preference model. In Level 4, the UTA algorithm (described in the next Section) is implemented to obtain a user preference model. And in Level 5, this model is validated against the user global preference structure.

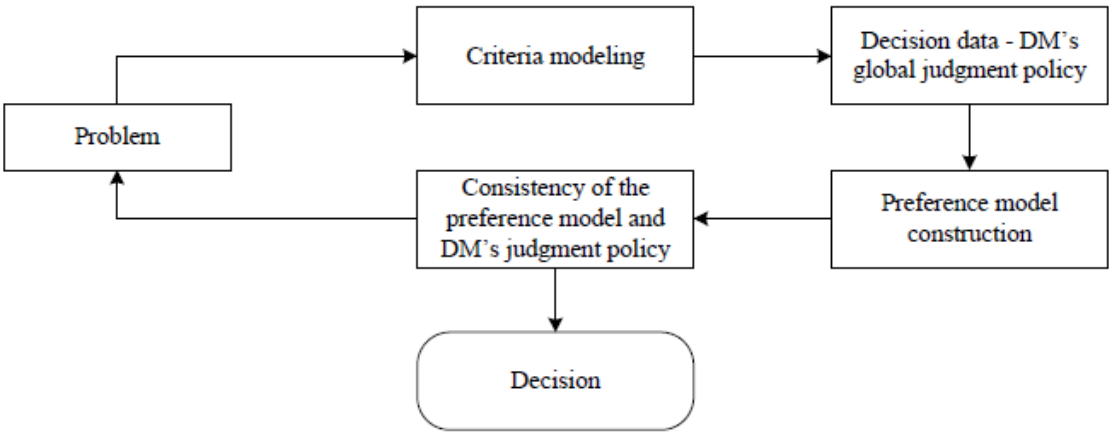


Figure 3.2: The aggregation disaggregation approach [39]

Development of the UTA method

The UTA (UTilités Additives) method, proposed by Jacquet-Lagrèze et al. [41] and explained in detail by Siskos et al. [38] as shown in this Section, aims to infer a criteria aggregation model from a given ranking on a reference set A_R, using linear programming techniques. The criteria aggregation model is assumed to be one or more additive value functions of the following form:

$$u(g) = \sum_{i=1}^{n} p_i u_i(g_i) \qquad (3.3)$$

where u_i, $i=1,2,\ldots,n$ are non-decreasing real value functions, termed marginal value or utility functions, normalized between 0 and 1 so they can be compared to each another. p_i is the weight for the utility function u_i for the user, presented as a percentage. Thus, the sum of all the weights of the different value functions for that user must equal *1*. Finally, g_i represents the i-th criterion under which the actions or items are being evaluated. The utility value functions must be chosen so the utility for the lowest value in the scale of the criterion *g* equals *0*, and the utility for the highest value equals *1* (as depicted in Figure 3.3). These constraints are expressed by the following equation:

$$\begin{cases} \sum_{i=1}^{n} p_i = 1 \\ u_i(g_{i*}) = 0, u_i(g_i^*) = 1 \forall i = 1,2,\ldots,n \end{cases} \qquad (3.4)$$

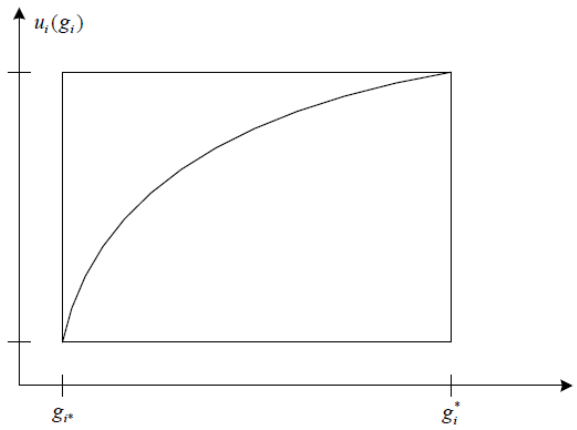


Figure 3.3: The normalized value function [38]

The global value function has the monotonicity property of the true criterion, and therefore the following properties hold for it. If the utility of the performance (i.e. the values for each criterion) of action *a* over all the criteria is greater than the utility of the performance of action *b* over the same criteria, then the user prefers action *a* over action *b*. However, if both utilities are the same it makes no difference for the user which action to select. These properties are expressed by the following equation:

$$\begin{cases} u\left[g(a)\right] > u\left[g(b)\right] \Leftrightarrow a \succ b \text{ (preference)} \\ u\left[g(a)\right] = u\left[g(b)\right] \Leftrightarrow a \ \square \ b \text{ (indifference)} \end{cases} \tag{3.5}$$

Since the UTA methods assume preferential independence of the criteria for the user, they infer an un-weighted form of the additive value function, as defined in equation (3.3). This un-weighted value function is shown by the following equation:

$$u(g) = \sum_{i=1}^{n} u_i(g_i) \tag{3.6}$$

Subject to the following un-weighted forms of its constraints, previously defined in equation (3.4):

$$\begin{cases} \sum_{i=1}^{n} u_i(g_i^*) = 1 \\ u_i\left(g_{i*}\right) = 0 \ \ \forall i = 1,2,\ldots,n \end{cases} \tag{3.7}$$

According to the model shown in equations (3.6) and (3.7), and considering the preference conditions expressed in equation (3.5), the value of each alternative $a \in A_R$ is given by the sum of all the utilities obtained from each of its criteria, as expressed by the following equation:

$$u'\left[g\left(a\right)\right] = \sum_{i=1}^{n} u_i\left[g_i\left(a\right)\right] + \sigma\left(a\right) \ \forall a \in A_R \qquad (3.8)$$

where $\sigma(a)$ is a potential error relative to $u'[g(a)]$. In order to estimate the corresponding marginal value functions, Jacquet-Lagrèze et al. [41] proposed the following technique: First, for each criterion, the interval $[\, g_{i*}, g_i^* \,]$ is divided into $(\alpha_i - 1)$ equal intervals. The end points for each interval g_i^j are obtained by the following equation:

$$g_i^j = g_{i*} + \frac{j-1}{\alpha_i - 1}\left(g_i^* - g_{i*}\right) \forall j = 1, 2, \ldots, \alpha_i \qquad (3.9)$$

Then the marginal value of an action *a* is approximated by a linear interpolation. Thus, the utility of an action performance over the criterion $g_i(a) \in [g_i^j, g_i^{j+1}]$ is given by the following equation:

$$u_i\left[g_i\left(a\right)\right] = u_i\left(g_i^j\right) + \frac{g_i\left(a\right) - g_i^j}{g_i^{j+1} - g_i^j}\left[u_i\left(g_i^{j+1}\right) - u_i\left(g_i^j\right)\right] \qquad (3.10)$$

Subsequently, the set of reference actions $A_R=\{a_1,a_2,\ldots,a_m\}$ is sorted in non-descendent order, such that a_1 is the head of the ranking (i.e. the highest ranked action by the user), and a_m is the tail (i.e. the lowest ranked action by the user). Consequently, each pair of consecutive actions (a_k, a_{k+1}) holds either $a_k \succ a_{k+1}$ (preference) or $a_k \square\, a_{k+1}$ (indifference). The difference between two consecutive actions is defined by the following equation:

$$\Delta\left(a_k, a_{k+1}\right) = u'\left[g\left(a_k\right)\right] - u'\left[g\left(a_{k+1}\right)\right] \qquad (3.11)$$

Thus, the preference conditions previously defined in equation (3.7) can now be defined in terms of these differences between consecutive actions, as expressed in the following equation:

$$\begin{cases} \Delta(a_k, a_{k+1}) \geq \delta \ \text{ iff } a_k \succ a_{k+1} \\ \Delta(a_k, a_{k+1}) = 0 \ \text{ iff } a_k \ \square \ a_{k+1} \end{cases} \tag{3.12}$$

where δ is a small positive number, to significantly discriminate the utility difference of two consecutive actions. Considering the hypothesis on monotonicity of preferences, the marginal value (utility) $u_i(g_i)$ obtained for a rate on one criterion should be higher than the utility obtained from the same utility function for a smaller rate on the same criterion. These constraints are expressed by the following equation:

$$u_i\left(g_i^{j+1}\right) - u_i\left(g_i^{j}\right) \geq s_i \quad \forall j = 1, 2, \ldots, \alpha_i - 1, i = 1, 2, \ldots, n \tag{3.13}$$

Here, $s_i \geq 0$ represents indifference thresholds defined on each criterion g_i. The thresholds are introduced to ensure, when needed, that the utility difference between two consecutive rates on the same criterion is higher than the threshold. This leads to better utility differentiations between user consecutively assigned rate values for each criterion. However, according to Jacquet-Lagrèze et al. [41], for most applications of the UTA methods the thresholds are unnecessary ($s_i = 0$), except when the following condition holds: $u_i(g_i^{j+1}) = u_i(g_i^j)$ *and* $g_i^{j+1} \succ g_i^j$. Finally, the marginal value functions are estimated by means of the linear program defined in equation (3.14). The objective of this program is to minimize the sum of potential errors $\sigma(a)$, with equations (3.6), (3.7), (3.12) and (3.13) as constraints.

$$\begin{cases} [min] F = \sum_{a \in A_R} \sigma(a) \\ \text{subject to:} \\ \Delta(a_k, a_{k+1}) \geq \delta \ \text{ iff } \ a_k \succ a_{k+1} \\ \Delta(a_k, a_{k+1}) = 0 \ \text{ iff } \ a_k \ \square \ a_{k+1} \\ u_i\left(g_i^{j+1}\right) - u_i\left(g_i^{j}\right) \geq 0 \ \forall i \, and \ j \\ \sum_{i=1}^{n} u_i\left(g_i^{*}\right) = 1 \\ u_i\left(g_{i*}\right) = 0, \ u_i\left(g_i^{j}\right) \geq 0, \ \ \sigma(a) \geq 0 \quad \forall a \in A_{R,}, \ \forall i \text{ and } j \end{cases} \tag{3.14}$$

If the optimum $F^* = 0$, then the polyhedron of admissible solutions (i.e. optimal solutions) for $u_i(g_i)$ is not empty, and many value functions lead to a perfect representation of the weak order provided by

the user. However, even when F^* is positive, other solutions (i.e. non-dominant solutions) which improve some criterions without sacrificing the others are found. Therefore, the UTA method ensures that the best possible optimal value function is always found. The post-optimal solutions space, shown in Figure 3.4, is defined by the following polyhedron:

$$\begin{cases} F \le F^* + k(F^*) \\ \text{all constraints of Linear Program (3.14)} \end{cases} \tag{3.15}$$

where $k(F^*)$ is a positive threshold, which is a small portion of F^*. Jacquet-Lagrèze et al. [41] proposed the partial exploration of the polyhedron from equation (3.15), finding its minimum and maximum optimal solutions by solving the following linear programs:

$$\begin{cases} [min]\, u_i(g_i^*) \\ \text{in} \\ \text{polyhedron (3.15)} \end{cases} \text{and} \begin{cases} [max]\, u_i(g_i^*) \\ \text{in} \\ \text{polyhedron (3.15)} \end{cases} \quad \forall i = 1, 2, \ldots, n \tag{3.16}$$

The solutions of these linear programs give the internal variation of the weight of all criteria g_i, and thus represent the importance of all criteria in the user preference model. Finally, instead of selecting one of these possible optimal or non-dominant solutions as the solution to the problem, Jacquet-Lagrèze et al. [41] proposed using the average of them as the final solution of the problem.

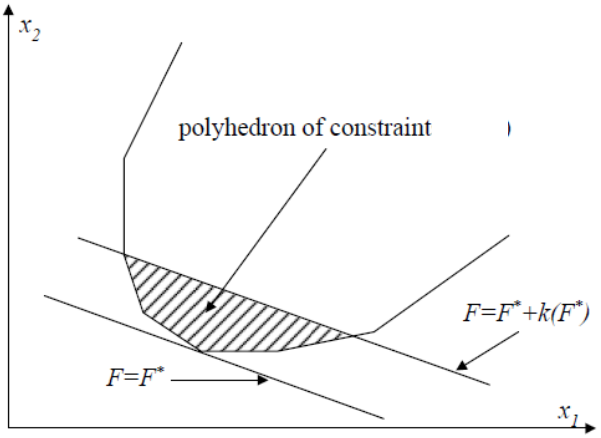


Figure 3.4: Post-optimality analysis [41]

The UTASTAR Algorithm

The UTASTAR (UTA*) method, proposed by Siskos et al. [42], is an extended version of the UTA method. Differences and improvements between both methods are briefly presented in this Section, in addition to a summary of the steps to implement the UTA* algorithm. In the original version of UTA, for each action $a \in A_R$, a single error $\sigma(a)$ is introduced to be minimized. However, this error function is not enough to completely minimize the dispersion between rankings and global values assigned by the user, as depicted in Figure 3.5. Therefore, UTA* introduces a double positive error function so that equation (3.8), to obtain the utility of an item, is reformulated as follows [38]:

$$u'\left[g(a)\right] = \sum_{i=1}^{n} u_i\left[g_i(a)\right] - \sigma^{+}(a) + \sigma^{-}(a) \;\; \forall a \in A_R \qquad (3.17)$$

where σ^{+} and σ^{-} are the overestimation and the underestimation errors, as shown in Figure 3.5.

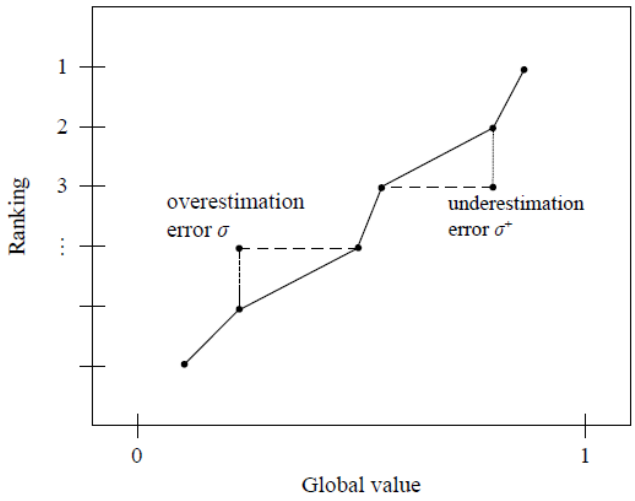


Figure 3.5: Rankings versus global values assigned by the user [38]

Another important modification is related to the monotonicity constraints of the criteria, defined by equation (3.18). These constraints, previously presented as indifference thresholds defined on each criterion, are now considered when solving the linear program [38].

$$w_{ij} = u_i\left(g_i^{j+1}\right) - u_i\left(g_i^{j}\right) \geq 0 \;\; \forall i = 1,2,\ldots,n \text{ and } \; j = 1,2,\ldots,\alpha_i - 1 \qquad (3.18)$$

Now the monotonicity conditions from equation (3.13) can be replaced by the non-negative constraints for the variables w_{ij} (for $s_i = 0$) [38]. Taking these changes and improvements into consideration, applying the UTA* algorithm can be summarized by the following four steps.

Step 1. First express the global value of reference actions $u[g(a_k)], k = 1,2,\dots,m$ in terms of marginal values $u_i(g_i)$, and then in terms of variables w_{ij} according to equation (3.18). Thus, the transformation of the global value of reference actions into a weight values expression is achieved by the following equation [42] [38] [11]:

$$\begin{cases} u_i\left(g_i^1\right) = 0 \quad \forall i = 1,2,\dots,n \\ u_i\left(g_i^j\right) = \sum_{t=1}^{j-1} w_{it} \quad \forall i = 1,2,\dots,n \text{ and } j = 2,3,\dots,\alpha_i - 1 \end{cases} \tag{3.19}$$

Step 2. Introduce two error functions, σ^+ and σ^-, on A_R by applying the following equation for each pair of successive actions in the given ranking [42] [38] [11]:

$$\Delta\left(a_k, a_{k+1}\right) = u\left[g\left(a_k\right)\right] - \sigma^+\left(\alpha_k\right) + \sigma^-\left(\alpha_k\right) - u\left[g\left(a_{k+1}\right)\right] + \sigma^+\left(\alpha_{k+1}\right) - \sigma^-\left(\alpha_{k+1}\right) \tag{3.20}$$

Step 3. Solve the following linear program [42] [38] [11]:

$$\begin{cases} [min]\, z = \sum_{k=1}^{\mu} [\sigma^+\left(\alpha_k\right) + \sigma^-\left(\alpha_k\right)] \\ \text{subject to:} \\ \Delta\left(a_k, a_{k+1}\right) \geq \delta \text{ if } a_k \succ a_{k+1} \ \forall k \\ \Delta\left(a_k, a_{k+1}\right) = 0 \text{ if } a_k \ \square \ a_{k+1} \ \forall k \\ \sum_{i=1}^{n} \sum_{j=1}^{a_i - 1} w_{ij} \\ w_{ij} \geq 0, \ \sigma^+\left(\alpha_k\right) \geq 0, \ \sigma^-\left(\alpha_k\right) \geq 0 \quad \forall i, j \text{ and } k \end{cases} \tag{3.21}$$

Step 4 (stability analysis). Obtain the final solution from multiple optimal or near optimal solutions of the linear program from equation (3.21). In the case of multiple solutions, find the mean additive value function of the optimal solutions that maximize the objective functions of equation (3.22). The polyhedron is bounded by the constraints expressed by equation (3.23), where z^* is the optimal value of the linear program in Step 3, and ε is a very small positive number [42] [38] [11].

$$u_i\left(g_i^*\right)=\sum_{j=1}^{a_i-1}w_{ij}\quad \forall i=1,2,\ldots,n \tag{3.22}$$

$$\sum_{k=1}^{m}\left[\sigma^+\left(a_k\right)+\sigma^-\left(a_k\right)\right]\le z^*+\varepsilon \tag{3.23}$$

In this Section we presented the UTA methods, as well as an extended version of the traditional UTA methods, the UTA* algorithm, as a solution to considering multi-criteria ratings in the decision making process. These methods are intended to create a user preference model from users' explicit preferences. The model can later be used by the recommender engine to assess the utility for each item for a specific user. The possible criteria to evaluate each item may include information about the item characteristics or, in some cases, about how the item relates to the user. The latter type of information can contain data such as the distance from the item to the user's current position, or the time the user would need to reach the item based on his/her mobility level. In these cases, the user context (position and mobility level) is considered in the user preference model. A number of methodologies and techniques have been proposed to include and model the user context in the recommendation process. These techniques are presented in detail in the following Section.

3.2 Context Awareness

Most existing RSs recommend items after considering only the user and item information in the recommendation process, using one of the methods described in Chapter 2,. However, another type of RSs, known as context aware Recommender Systems (CARSs), provide recommendations to the user considering also the user's context information. Context information includes data such as location (referred to as location aware), orientation, temperature, velocity, biometrics, the people near the user, the objects around him/her as well as any changes in these elements. Context-awareness is defined as the ability of a system to be aware of the user's surrounding physical environment and state [43] [44]. This Section examines CARSs, presenting different approaches and techniques that have been implemented to include context information in the recommendation process.

3.2.1 Context Aware Recommender Systems

CARSs deal with modeling and predicting user tastes and preferences by considering contextual information as additional categories of data. In CARSs, similarly to traditional two-dimensional (2D)

Recommender Systems, these preferences are expressed as ratings. However, these ratings are modeled not only as a function of items and users, but also of context, as follows [2]:

$$R : User \times Item \times Context \rightarrow Rating \qquad (3.24)$$

Context information is comprised of different types of data, with each type defining a particular aspect of context (e.g. time, location, companion, purpose). However, not all aspects of context are always relevant or useful, for recommendation purposes. Depending on the domain of the recommendations and the information available, some types might be important and others not. Thus, in order to determine the relevance of a given type of contextual information, several techniques from machine learning, data mining and statistics have been applied; these are known as feature selection and feature extraction techniques. Feature selection techniques are intended to identify the most representative subset from the original set of context aspects, while feature extraction techniques aim to produce a new set by transforming the original set of context aspects [45] [46]. Both techniques are meant to be applied during the data pre-processing phase, to measure the relevance or importance of the attributes for predicting the rating of an item. For more information on methods and techniques to determine the relevance of specific context aspects within specific domains, we suggest [47] [48] [49].

In order to model contextual information, Ricci et al. [2], defined context as a set of contextual dimensions K that are defined by a set of q attributes $K = (K^1, \ldots, K^q)$, with a hierarchical structure and capturing a particular type of context. The values of attribute K^q define finer (more granular) levels of contextual information, while attribute K^1 values define coarser (less granular) levels. Another way to model context information for RSs, introduced by Adomavicius et al. [50], uses OLAP-based multi-dimensional (MD) data models. In OLAP models, $D_1, D_2, \ldots, D_n$ are dimensions, two of which represent the User and the Item, and the rest the different types of context information. Each of the dimensions D_i is a subset of a Cartesian product of some attributes A_{ij}, $(j=1, \ldots, k_i)$, which define a domain or a set of values.

$$D_i \subseteq A_{i1} \times A_{i2} \times \ldots \times A_{ik_i} \qquad (3.25)$$

Consider the recommendation space defined by the User, Item and Time dimensions. The User dimension represents a set of users having specific names, addresses, incomes, and ages. The Item dimension represents a set of items defined by their names, types and price. And the Time dimension consists of a list of days [2].

$$User \times Item \times Time \begin{cases} User \subseteq UName \times Address \times Income \times Age \\ Item \subseteq IName \times Type \times Price \\ Time \subseteq Year \times Month \times Day \end{cases} \quad (3.26)$$

Thus, the recommendation space is defined as the Cartesian product $S = D_1 \times D_2 \times D_3$, and the rating domain as an ordered set of all possible rating values. The rating function is defined over the space $D_1 \times \ldots \times D_n$, as follows:

$$R : D_1 \times \ldots \times D_n \rightarrow Rating \quad (3.27)$$

Therefore, we can define a rating in the space $R(u,i,t)$ as how much user $u \in User$ preferred item $i \in Item$ at time $t \in Time$, and visually store the ratings in a multidimensional cube, as shown in Figure 3.6. Function R is a partial function, as is common in RSs, where the initial set of ratings is known, and the goal is to estimate the unknown ratings [2]. The main advantage of this modeling technique is that it represents the dimensions and estimated ratings in a structured and known form. Therefore, this information can be accessed, displayed and analyzed by the systems and OLAP tools that the user already has and uses. The fact that the results produced by the RS match an already known data structure, provides the potential benefit of being coupled as part of a larger, integrated intelligent system.

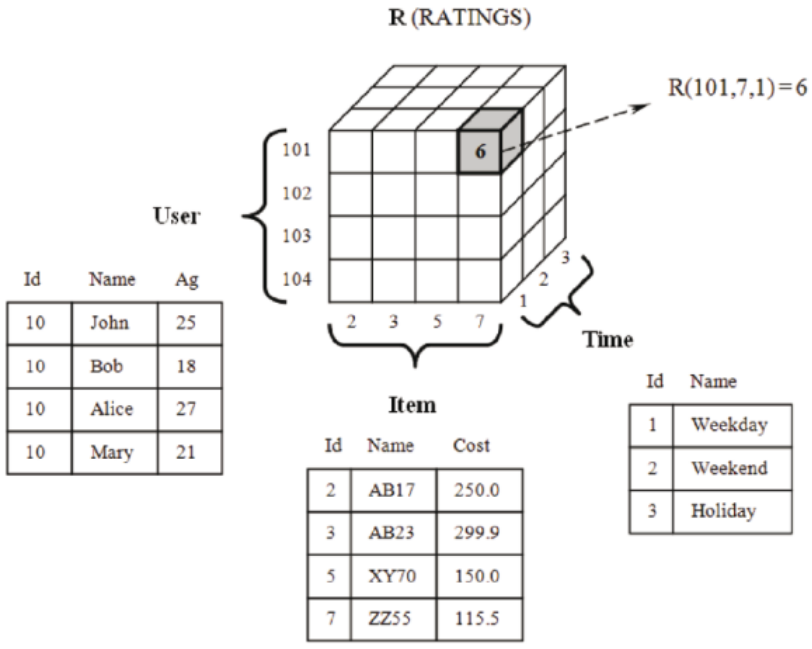


Figure 3.6: Multidimensional model for the User x Item x Time recommendation space [2]

In this Section we presented two techniques that have been applied to model the context information within the recommendation space. This is done so the recommendation engines can consider the context data when assessing the utility for each item. The following Section presents three different approaches that use this information as part of the recommendation process.

3.2.2 Incorporating Context in the Recommendation Process

A traditional 2D recommendation process is performed using the following three components: data (including users, items and ratings), a 2D recommender function, and a recommendation list. The recommendation list for a specific user *u* is obtained by applying the recommendation function on user *u* and all possible items, in order to predict a rating for each of them and be ranked accordingly. This process is presented in the following Figure [2].

Figure 3.7: General components of the traditional recommendation process [2]

However, in CARSs the initial data includes information about users, items, ratings and context (*U x I x C x R*) and, unlike 2D RSs, contextual information can be applied at different stages of the recommendation process. This leads to three different approaches, identified and described by Ricci et al. [2] as follows: **Contextual pre-filtering**. Information about the context is used as a filter over the initial data, in order to select only the information that matches a specific context. Subsequently, any 2D Recommender System can be applied over the filtered data. **Contextual post-filtering**. Context information is initially ignored and ratings are produced using a 2D Recommender System, then the estimated ratings are adjusted for each user using the contextual information. **Contextual modeling**. Contextual information is directly used in the recommendation process as part of the rating estimation. The three approaches are shown in Figure 3.8.

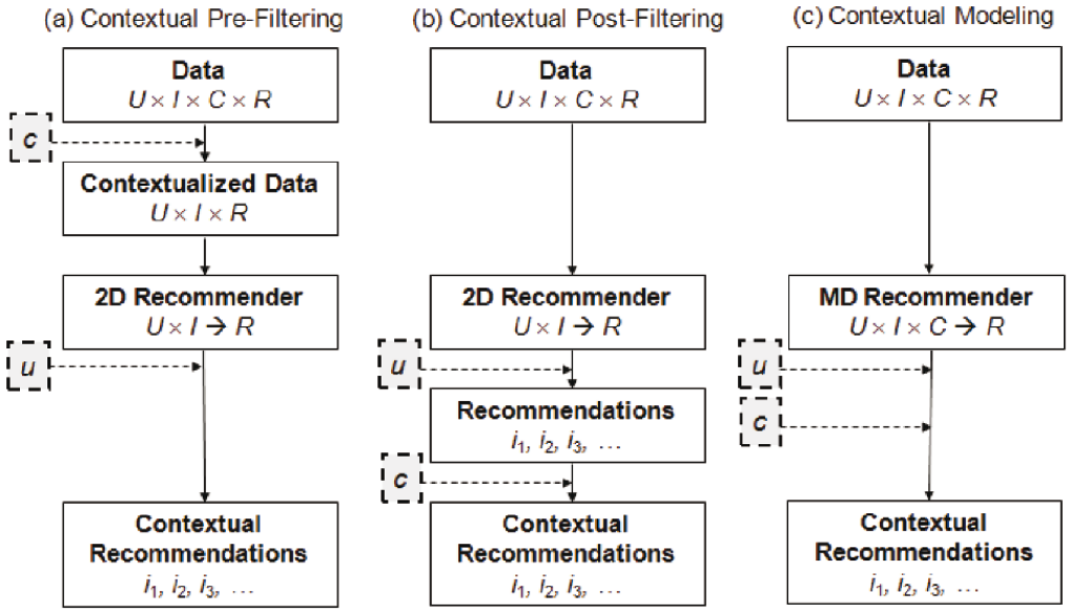


Figure 3.8: Techniques for incorporating context in RSs [2]

Contextual Pre-Filtering

Contextual pre-filtering, as shown in Figure 3.8.a, uses context information only as a filter to select the most relevant 2D data to generate recommendations. Hence, the main advantage of this method is that it allows us to apply a recommendation technique from the previously proposed 2D RSs. This approach, proposed by Adomavicius et al. [50], and also known as reduction-based technique, decreases the problem of multidimensional (MD) contextual recommendations to the standard 2D (*User x Item*) recommendation space. Thus, for a recommendation space with *User x Item x Time* dimensions, the rating prediction function would be expressed as a 2D function, as follows:

$$\forall (u,i,t) \in U \times I \times T, R^{D}_{User \times Item \times Time}(u,i,t) = R^{D[Time=t](User,Item,Rating)}_{User \times Item}(u,i) \qquad (3.28)$$

where *[Time = t]* represents the contextual filter, and *D[Time = t](User, Item, Rating)* represents a rating dataset obtained from *D* by selecting only the instances where the Time dimension has value *t*. The main drawback of this approach is that sometimes the context can be so narrow that some of its aspects may not be significant, or not have enough data after the filtering to estimate accurate recommendations (i.e. a sparsity problem). In some cases, the remaining filtered data instances are not enough to produce accurate recommendations. To overcome this problem, the following context generalization method is proposed [50].

Context Generalization

Context generalization is a method that allows generalization of the data filtering query based on a specific context. Formally, let $c' = (c_1', \dots, c_k')$ be a generalization of context $c = (c_1, \dots, c_k)$, if and only if $c_i \rightarrow c_i' \; \forall \; i = 1, \dots, k$ in the corresponding context hierarchy. Then *c'* can be used instead of c to filter the input data. This idea generally proposes not using a simple filter *[Time = t]*, which represents the exact content *t* of the rating *R(u,i,t)*, but rather using a generalized filter $[Time \in S_t]$, where S_t expresses a higher hierarchy level of context t, known as a contextual segment [50]. However, this makes choosing the 'right' contextual level (hierarchy level) more difficult, and presents a trade-off between having more general data, or having less but more relevant data, to estimate unknown ratings. One way to address this problem is to empirically evaluate the performance of the RS using all possible context levels, then choosing the one with the best performance. Regardless, in cases with highly granularity contexts, the number of possible hierarchy levels may be very large, rendering exhaustive search techniques impractical. For these cases, heuristic greedy approaches are required [2].

Contextual Post-Filtering

Contextual post-filtering, as shown in Figure 3.8.b, disregards the context information from the input data during the recommendation process, to later adjust the obtained recommendations for each user considering the context of the user. Similar to the contextual pre-filtering approach, this technique also has the advantage of being able to apply any recommendation technique from the previously proposed 2D RSs presented in Chapter 2. Estimated recommendations can be adjusted by either filtering out the recommendations that are irrelevant for the user's current context, or adjusting the ranking of the produced recommendations based on the user's current context. The basic idea is to analyze the contextual preference data for a given user in a given context to find specific item usage patterns, then use these patterns to filter or adjust the recommended item list, leading to contextual recommendations. This process is shown in the following Figure [2].

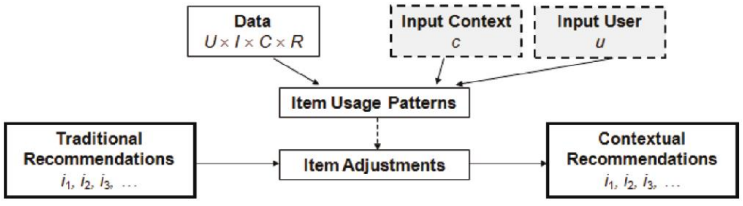


Figure 3.9: Final phase of contextual post-filtering approach [2]

Contextual post-filtering can be further classified into heuristic and model-based techniques. Heuristic techniques focus on finding common item characteristics for a given user in a given context, then using these common attributes to adjust the recommendations. The adjustment can be done by either filtering out the items that do not have a significant number of common attributes, or ranking the items based on how many of these attributes each holds. Model-based post-filtering techniques build models that calculate the probability that a user would prefer a specific item within a specific context, then use this probability to adjust the recommendations. In this case, the adjustment can be done by either filtering out the items that have a probability smaller than a selected threshold, or ranking the predicted ratings by their probability of relevance [2].

Contextual Modeling

Contextual modeling, as depicted in Figure 3.8.c, uses the contextual information during the recommendation process directly. Therefore, it requires real multidimensional recommendation functions. These multidimensional functions essentially represent predictive models [2], and are further classified, based on the algorithm that they use, as heuristic-based or model-based approaches. Both approaches are discussed briefly in the following Sections.

Heuristic-Based Approaches

The traditional 2D neighborhood-based approach, presented by Breese et al. [27] and Sarwar et al. [30], was extended by Adomavicius et al. [51] to be used in multidimensional scenarios by applying an n-dimensional distance metric, instead of user-user or item-item similarity metrics. Considering the recommendation space of *User x Item x Time*, the predictions of a specific rating $r_{u,i,t}$ based on the weighted sum of relevant ratings, is expressed as follows:

$$r_{u,i,t} = k \sum_{(u',i',t') \neq (u,i,t)} W\left((u,i,t),(u',i',t')\right) \times r_{u',i',t'} \qquad (3.29)$$

where $W((u,i,t),(u',i',t'))$ represents the weight of rating $r_{u',i',t'}$, and k is a normalizing factor. These weights W are obtained by the distance between points (u,i,t) and (u',i',t') in multidimensional space, expressed as $dist[(u,i,t),(u',i',t')]$. There are several metrics to calculate the distance between points, recall from Section 2.1.3, the Euclidian and Manhattan distances. Both distances can be modified to measure the weighted distance between two points in a multidimensional space, as presented in equations (3.30) for the Manhattan distance, and (3.31) for the Euclidean.

$$dist\left[(u,i,t),(u',i',t')\right] = w_1 d_1(u,u') + w_2 d_2(i,i') + w_3 d_3(t,t') \quad (3.30)$$

$$dist\left[(u,i,t),(u',i',t')\right] = \sqrt{w_1 d_1^2(u,u') + w_2 d_2^2(i,i') + w_3 d_3^2(t,t')} \quad (3.31)$$

Here, d_1,d_2, and d_3 are distance functions defined for dimensions User, Item and Time respectively, and w_1,w_2, and w_3 are the weights assigned for each of them.

Model-Based Approaches

Some existing 2D model-based methods have also been extended for multidimensional cases. For instance, the method proposed by Ansari et al. [33], presented in Section 2.1.3. In this method, the unknown rating estimation problem for *User x Item* space was defined as follows.

$$r_{ui} = x_{ui}^{'}\mu + z_u^{'}\gamma_i + w_i^{'}\lambda_u + e_{ui},\ e_{ui} \sim N\left(0,\sigma^2\right),\ \lambda_u \sim N\left(0,\Lambda\right),\ \gamma_i \sim N(0,\Gamma) \quad (3.32)$$

where z_u is a vector that represents the attributes of user u, w_i is a vector representing the attributes of item i, x_{ui} is a vector containing all possible cross-product combinations between individual elements z_u and w_i, μ is a vector representing unknown coefficients in the regression model that need to be estimated, γ_i represents the heterogeneity of item i that is not explicitly recorded in the item information, and λ_u represents the heterogeneity of user u. e_{ui} is an error normally distributed with mean zero and standard deviation σ. Regression parameters γ_i and λ_u are normally distributed as $\lambda_u \sim N(0,\Lambda)$ and $\gamma_i \sim N(0,\Gamma)$. The parameters of the model μ, σ^2, Λ and Γ are estimated from the data containing already known user ratings, using the *Markov Chain Monte Carlo* (MCMC) technique, as presented in Section 2.1.3.

Adomavicius et al. [50], presents an extension of this method to consider multiple dimensions, which states that for a recommendation space formed by the three dimensions *User x Item x Time*, the ratings for each item are calculated by applying the following equation:

$$r_{uit} = x_{uit}^{'}\mu + p_{ui}^{'}\theta_t + q_{it}^{'}\lambda_u + r_{tu}^{'}\gamma_i + z_u^{'}\delta_{it} + w_i^{'}\pi_{tu} + y_t^{'}\sigma_{ui} + e_{uit} \quad (3.33)$$

where $e_{uit} \sim N(0,\sigma^2)$, $\gamma_i \sim N(0,\Gamma)$, $\lambda_u \sim N(0,\Lambda)$, $\theta_t \sim N(0,\Theta)$, $\delta_{it} \sim N(0,\Delta)$, $\pi_{tu} \sim N(0,\Pi)$ and $\sigma_{ui} \sim N(0,\Sigma)$. This extended model considers the effect of observed and unobserved user-item-temporal variables and their interactions to predict r_{uit}. The vector p_{ui} represents the interaction of the

observed user and item variables. Similarly, q_{it} and r_{tu} represent the interactions between the item/time and time/user variables respectively. The vector σ_{ui} represents the interaction of the unobserved user and item attributes, and vectors π_{tu} and δ_{it} denote similar interactions. Finally, the parameters of the model (i.e. μ, σ^2, Λ, Γ, Θ, Δ, Π and Σ) are estimated from the data of the known ratings using a MCMC technique, as in the 2D approach. The main disadvantage of this method is that the number of parameters to be estimated increases with the number of considered dimensions, which leads to a sparsity problem.

In this Section, we presented two techniques to model user context information within the recommendation space, and three methods to consider it during the recommendation process. However, in some cases there is not enough user information to identify his/her context, or to produce accurate recommendations for him/her. Therefore, in the following Section we present a methodology to extend the user profile, by generalizing user information from a large-scale demographic database.

3.3 Generalizing User Profiles

As explained in Chapter 2, user profiles are normally obtained by analyzing the information provided by the user, to reveal user patterns or models. After user profiles have been built, RSs can use them to assess the utility of each item for a specific user. This process of assessing the utility from user profiles is usually done using either content-based or collaborative-based methods. Recall that content-based methods extract features from each item and compare them against the attributes identified as relevant within each user profile, while collaborative-based methods compare the profiles of all the users in order to find overlaps in interests, then recommend the items that similar users preferred in the past. However, both approaches have the following limitations: Content-based methods can only encompass direct generalizations of the input features, so users are typically required to specify a large number of samples to extract a user profile, and the systems can only reason about each user rather than the commonalities between them. On the other hand, the main limitation of collaborative-based methods is they require data from a large number of users before they can be effectively applied. Hence Krulwich et al. [52] presented another technique, known as the demographic generalization method that combines the benefits of these previous two approaches, which is described in the following Section.

3.3.1 Demographic Generalization Method

This method exploits the information contained in large-scale demographic databases that encompasses interests and preferences of people living in a certain region. When applying this method, the information provided by users is drawn on to classify them in terms of these demographic data, and the classifications are then used as general characterizations of them and their interests. The resulting user profiles span the range of information contained in the demographic database [52].

The generalization is applied as follows: First, given some user information, the set of demographic clusters to which the user is most likely to belong is computed. If only one cluster matches, all the available data for the cluster is used as the user profile, and the process ends. However, if more than one cluster matches the user information a partial user profile is built, which contains the information of similar demographic variables in the matching clusters. Subsequently the demographic variables that best differentiate the matching clusters are used to obtain more information from the user, thereby obtaining a new, narrower set of matching clusters. The process can reiterate until the user profile converges on a single matching cluster, or it can stop at any point and use the partial user profile. This process is presented in Figure 3.10 [52].

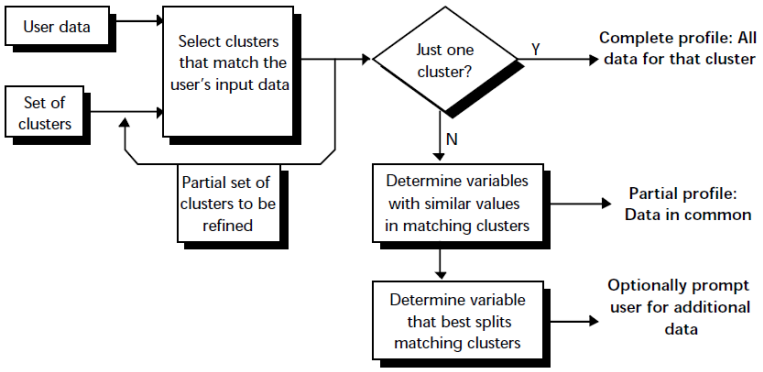


Figure 3.10: Process of Demographic Generalization method [52]

The following two techniques can be applied to select matching clusters based on user information. Use the information as constraints to filter only the matching demographic clusters, or apply a data-mining clustering algorithm over the user information and the demographic database.

The demographic generalization method presented in this Section has three noteworthy characteristics. First, it is designed to incrementally and iteratively construct the user profile, which means a partial user profile is always available, and the user has the option of continuing to answer

selected questions to refine and improve his/her profile, leading to more accurate and better recommendations. Second, it can operate with a minimal amount of information from the user, and does not require information from other users to be effective. Third, the profile created by this method can later be used to recommend items from areas where the user has never input information [52].

This demographic generalization technique to extend user provided information from a large-scale demographic database is the final technique to extend the RS capabilities addressed in this thesis. The following section presents different experiments and properties that can be used to evaluate or compare RSs.

3.4 Evaluating Recommender Systems

Traditionally, RSs have been evaluated based only on their prediction accuracy; their ability to accurately predict users' selections. However, in many application domains users use RSs for more than accurate recommendations based on their preferences and tastes. They may also be interested in discovering new items, system response, the diversity of recommended items, intuitive user interfaces and so forth. Hence, Ricci et al. [2] identified and presented fourteen different properties under which RSs can be evaluated and compared. Moreover, three different experiments have been proposed to measure the values of each of these system properties. These experiments and properties are presented in this Section.

3.4.1 Types of Experiments to Evaluate a RS

In this Section we briefly present three types of experiments: Offline experiments, which are the easiest to conduct since they don't require interaction with users; user studies, in which a small group of users interacts with the system in a controlled environment, and are then asked about their experiences; and online evaluation, in which the system is used by real users who are unaware of the experiment. This latter approach, while closest to reality and potentially considered the most trustworthy, can collect only limited types of data [2].

Offline Experiments

The goal of offline experiments is to simulate the behavior of users who interact with a RS, by using a previously collected set of user selected or rated items. The user data is used to create the user

profile, and to compare the predicted results of the system. Offline experiments are attractive by their low implementation cost, due the fact that they don't require any real user interaction with the system. For this reason these experiments are the most appropriate to compare several techniques, or to evaluate a system at an earlier development phase when it is not yet fully integrated. However, the main disadvantage of offline experiments is they are limited to evaluating only the system properties that can be determined from the available user information. Factors such as the system's influence on the user, the user's overall appreciation of the system, the system's user interface and others cannot be measured.

The data required for these experiments includes information about the user (to create the user profile), and the final user preferences over a set of reference items (to compare the results obtained by the system). Therefore, when collecting information from the users, it is important to ask them the same type of information they would be required to input to the system, and to provide their item preferences in the same way the system would present the recommended items to them. Additionally, it is important to ensure that there are no biases in the distribution of the users being queried, the information presented to the users, and the preferred items selected by the users (i.e. the user doesn't rate only items of the same kind).

When conducting offline experiments, it is necessary to simulate how the system would interact with users as closely as possible. First it will store some information about their preferences and subsequently would recommend new items to them. Ideally, the users would find the recommended items interesting and select them. This process can be simulated in offline experiments, by hiding some of the data given by the user about his/her preferred items when constructing the user profile and recommending the items. Subsequently, the hidden information is used to assess the recommendations provided by the system. Ideally, the system should recommend all these hidden items, and with the same user preference order [2].

User Studies

User studies, unlike offline experiments, involve actual user interaction with the system, and provide additional quantitative and qualitative information about system performance. This type of system evaluation is performed by selecting a group of users and asking them to use the system, during which time we observe their behavior and record a number of quantitative measures, such as the time taken to obtain a recommendation, which part took them the most time to complete, and the accuracy of the results, measured by how many times a recommended item was 'clicked'. Qualitative measures can also be recorded before, during and after the user interacts with the system, and could

include how intuitive the user interface is, the overall user experience, the ease of using the system, and whether the user thought the recommendations were relevant or not.

Of the three types of experiments, user studies can measure the widest set of quantitative system properties, and are also the only type that enables the collection of qualitative system properties. However, these studies are the most expensive to conduct, measured by the user time and monetary costs required to collect a large set of users and have them perform numerous tasks. For this reason, these studies are normally restricted to a small set of users and a limited number of tasks, which leads to the possibility that some scenarios might not be tested. Moreover, in order to obtain highly confident conclusions that can be generalized, each scenario must be tested several times, by different users. Due the high cost of these studies, as much user data as possible should be collected when being performed, and in the lowest granularity. To avoid bias, users selected for the study should, as closely as possible, represent the users that will interact with the real system. However, even when the selected users represent the exact distribution of the final system users, the results may still be biased due the fact that the study users are aware that they are participating in an experiment.

Finally, to avoid failed experiments due technical problems or system malfunctions, pilot user studies are recommended. These are not designed to collect user information, but to test the overall system functionality. The results of the pilot studies are used to improve the system, but should not be considered when obtaining the measurements in the final user study [2].

Online Evaluation

Online evaluations are the only experiment of the three that allow us to determine how much a system influences the behavior of the users, and how it improves overall system goals such as long-term profit or user retention. This is achieved by measuring the change in users' behavior when interacting with different RSs. Online evaluations are conducted by releasing the system and monitoring users' interactions with it. Since the users are not aware that they are in an experiment, the obtained results are less likely to be biased. Moreover, as the system is used by real users to perform real tasks in a real environment, these evaluations provide the strongest measure of the true value of a system.

Due to how they are applied, online evaluations are more suitable for comparing multiple algorithms, or evaluating the performance of an update in the actual recommender engine before it is released. In these scenarios, the system randomly redirects a percentage of the users to the alternative algorithms or updated engine, and then records the users' interactions for further analysis. However, it is important to note that these evaluations have some risk, since the users redirected to the test

systems may receive irrelevant or poor recommendations, which could discourage them from ever using the real system again [2].

3.4.2 Properties and Metrics to Evaluate a RS

In this Section we introduce a range of properties by which a RS can be measured or evaluated in order to determine which recommender engine is best for a particular application. Since different applications could have different requirements and goals, the properties used to evaluate a system should be selected according to each application domain. When selecting the properties to evaluate, it is also important to consider that some of them will have to be trade-off and cannot be equally improved in parallel. For example, the accuracy of the recommendations usually decreases when the diversity of recommended items is increased.

The following are brief descriptions of the RS properties that can be measured, as identified and presented by Ricci et al. [2], as well as the metrics that can be used to measure their values for a Recommender System.

User Preference

User preference refers to the overall user rating of a system, hence it can only be measured through user studies or online evaluation techniques. This property is particularly useful when comparing two or more RSs in order to choose which to use. It is measured by asking the participants to interact with all the subject systems, and then select the one they liked the most. However, this metric assumes that all the users are similar, which will not be the case in some scenarios; for example, the preferences of users with less experience using technological systems may be more desirable in some scenarios. Additionally, this metric doesn't consider the degree to which the user preferred one system over the others. For more information on this metric, we recommend [53].

Prediction Accuracy

Prediction accuracy refers to the precision of the recommended items, based on the constructed user profile and provided user preferences and rates. It is measured by calculating the matching percentage between the recommended and the user's previously selected items. One of the main advantages of this property is that it is independent from the user interface, and as such it can be measured using offline experiments. There are three different types of prediction accuracy that can be measured in a RS, depending on the type of results given by the system. Type one, the **accuracy of**

the rating predictions, is for systems intended to predict the rating a user would give an item (e.g. 1-5 stars). Type two, the **accuracy of usage predictions**, is for systems that aim to recommend to users items that they may use. In this type we are interested in whether the predicted items will be used by the user, independent of the estimated rating for them. Type three, the **accuracy of ranking of items**, is for systems that present the recommended items as an ordered ranked list to the user, imposing a specific browsing order on them. In this type we are no longer interested in predicting a specific rating for an item, or in selecting a set of recommended items that a user may use. Instead, we now consider the ordering of the recommended items according to the user's preferences.

To compute the accuracy of ranked items, we determine how closely a system's ordered list of items resembles the users' correct order. In the following Sub-Section we present a detailed technique to assess the accuracy for ranked items, since this is the type of results that our proposed RS produces (presented in Chapter 5). For more information on measuring the accuracy of rating predictions and usage predictions, we recommend [2] [54].

Using a Reference Ranking

Evaluating the ranking accuracy of items against a reference ranking (i.e. a correct order) requires items previously rated by the user; if these are unavailable, we need to ask the user to rate a set of items. The reference ranking list is obtained by ordering the reference set of items, based on the user provided rates. The ordering (i.e. non-ascendant or non-descendant) must be consistent with the one used by the RS under evaluation. Subsequently, the system produced ranked list of items can be compared against the reference ranking list. This is done by measuring the percentage of items from the reference set ordered by the system that are in the same relative position to the rest of the items in the reference ranking list [2].

It is important to note that the user may have assigned the same rank to two or more items, which means we have no information about the user's relative ordering between them. In these cases we consider the two items to be tied, and for tied items the system under evaluation should not be penalized for ranking one over the other, even if this ordering turns out to be different than the one in the reference ranking list. Yao et al. [55], presents the *Normalized Distance-based Performance Measure* (NDPM) to evaluate the system's ranking accuracy in these cases.

Obtaining the NDPM of the system requires the following four values to be assessed over all the possible pairs of different items from the reference set.

$$c^{+} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \begin{cases} 1: \text{ If } \operatorname{sign}(r_{u,i} - r_{u,j}) \operatorname{sign}(\hat{r}_{u,i} - \hat{r}_{u,j}) > 0 \\ 0: \text{ Otherwise} \end{cases} \tag{3.34}$$

$$C^{-} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \begin{cases} 1: \text{ If } sign\left(r_{u,i} - r_{u,j}\right) sign\left(\hat{r}_{u,i} - \hat{r}_{u,j}\right) < 0 \\ 0: \text{Otherwise} \end{cases} \tag{3.35}$$

$$C^{u} = \sum_{i=1}^{N} \sum_{j=i+1}^{N} sign^{2}\left(r_{u,i} - r_{u,j}\right) \tag{3.36}$$

$$C^{u0} = c^{u} - (c^{+} + c^{-}) \tag{3.37}$$

Here, N stands for the number of items in the reference set, $r_{u,i}$ represents the user given rate for item I, and $\hat{r}_{u,i}$ is the rate assigned by the system for item i. Consequently, C^{+} represents the number of pairs of items for which the system agrees with the user, C^{-} is the number of pairs of items for which the system disagrees with the user, C^{u} is the number of pairs of items for which the user provided a different rate (i.e. they are not tied), and $C^{u0}$is the number of pairs of items for which the user provided a different rate (not tied), but the system assigned them the same rate (tied). Since we are comparing each possible pair of different items from the reference set, equations (3.34), (3.35) and (3.36) will range over $\frac{1}{2}N(N-1)$ pairs of items.

After these values have been assessed, the NDPM of the system is obtained by the following equation:

$$NDPM = \frac{C^{-} + 0.5C^{u0}}{C^{u}} \tag{3.38}$$

The NDPM gives a perfect score of 0 to systems that correctly estimate every user preference relation from the reference ranking set, and it gives the worst score of 1 to systems that contradict every user preference relation from the reference ranking set. This measure does penalize a system if it doesn't estimate a relation between two items, but only half as much as if it contradicts it. Predicting a preference between tied items in the reference set doesn't penalize the system. Finally, the accuracy percentage of a system is given by the following equation:

$$Accuracy\ \% = \left(1 - NDPM\right) * 100 \tag{3.39}$$

Coverage

Coverage refers to the measurement of how much data the system considers during the recommendation process. This metric is particularly important for systems with a large number of available data about both items and users, since in these cases the system might present lists of recommendations with high prediction accuracy or utility, but only consider, and thus be able to recommend, a small subset of the available items in the dataset. Moreover, the coverage of a system can refer to the following three aspects of it: the items covered, the users covered, and the coverage for newly added users and items (known as 'cold start') [2]. These types of coverage are presented in the following Sub-Sections.

Item Space Coverage

Item space coverage, also known as catalog coverage, measures the percentage of items that can be recommended to users from all the items in the database. Though this can be measured by analyzing the recommendation algorithm and the item dataset, a more meaningful and realistic measurement is the percentage of items that were recommended to the users during the conducted experiments [2]. For systems that present their recommendations as an ordered ranked list, it is important to define an item utility threshold; the items from the list with a utility higher than the threshold are considered recommended items to the user.

Another catalog coverage metric, known as sales diversity, measures how disproportionately different items are chosen by users when different RSs are used. The Gini Index and the Shannon Entropy, are two metrics that can be used to measure sales diversity. For more information on the sales diversity measure and its metrics, we recommend [56].

User Space Coverage

User space coverage measures the percentage of users to whom the system can recommend items. A RS may not be able to recommend items to users, due to factors such as low accuracy in the produced recommendations, lack of information in the user profile, or incongruent and contradictory user given data, which leads to system inability to create a user preference model [2].

User space coverage can be measured using one or both of the following two metrics: the required amount of data in the user profile in order to make a recommendation (i.e. the number of items the user must rate in order to have new items recommended to him/her); and/or the percentage

of users from the experiments for whom the system was able to create a model, and produce recommendations with a prediction accuracy higher than a selected threshold.

Cold Start

Cold start is considered a coverage problem, since it measures the system coverage over newly added items [2]. It determines the amount of information each item requires for the system to be able to recommend it; for example, the number of required ratings of an item, the number of days it must be in the database, or the type of information it needs to contain.

Confidence

Confidence is a metric of how much the system trusts its recommendations or predictions [57]. It is expressed as the probability that the predicted value, or an interval around the predicted value, is indeed true; that is, the percentage of the population for whom the obtained results from the evaluation can be considered true. Consequently, the required number of users considered for the experiment (i.e. sample size) depends on the confidence level sought. The size of the sample required to run the experiments is given by the following equation [58]:

$$n = \frac{Z^2 * \sigma^2}{e^2} \qquad (3.40)$$

Here *Z*, the standard normal variable, represents the matching value in the normal distribution table (also known as Gauss table or z-table) to the desired confidence for the experiment. The matching value expresses to how many standard deviations, with respect to the media, the desired area under the curve of the normal distribution corresponds (i.e. the confidence). Recall, in a normal distribution there are three standard deviations on either side of the media. Figure 3.11 depicts the normal distribution curve, along with the matching z-values for some area percentages under it.

In equation (3.40), σ^2 is the population variance obtained by squaring the population standard deviation (σ). This standard deviation of the population can be obtained either from previous studies, or through a pilot study. When conducting a pilot study, according to the central limit theorem, the obtained standard deviation from the study (s) can be considered as a punctual estimation of the standard deviation of the population (σ) if the study considered 30 users or more. The central limit theorem states that, for a population distribution, regardless of the distribution shape of the sample, if

it considered at least 30 observations the media will have an approximated normal distribution. This theorem is of decisive importance when using inferential statistics to draw conclusions about a population. It enables inferences about the media of the population, without needing to know the specific shape of the population distribution in advance.

In equation (3.40), *e* represents the allowed sample error. This is a parameter of the equation that controls how much error is acceptable when inferring the value for the population from the value obtained in our experiment (i.e. sample). Hence, it represents the difference between the sample media and the population media.

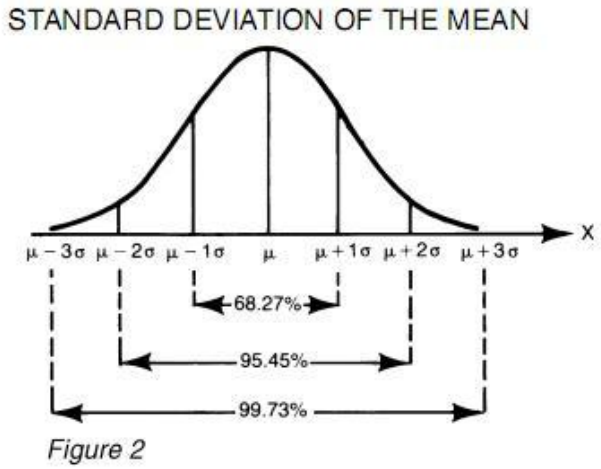


Figure 2

Percent	99.73%	99%	95.45%	95%	90%	80%	68.27%
No. of ± σ's	3.00	2.58	2.00	1.96	1.645	1.28	1.00

Figure 3.11: Normal distribution and corresponding z-values [59]

Once we have computed the resulting media value from our experiment, using a sample size obtained from equation (3.40), we infer the result for the population in general. A result for a population inferred from a sample media value, is presented as an interval around the sample media where the desired percentage of the population (i.e. confidence) will lie. This confidence interval is obtained by the following equation [58]:

$$\mu = \bar{x} \pm Z * \sigma_{\bar{x}} \qquad (3.41)$$

Here, μ represents the media value for the population, $\bar{x}$ the sample media value obtained from our experiment and, $\sigma_{\bar{x}}$ the standard error obtained using the following equation:

$$\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}} \qquad (3.42)$$

In equation (3.42), σ represents the standard deviation of the population, which was punctually estimated from the standard deviation of the sample (s). n is the size of the sample, obtained from equation (3.40).

Trust

Trust is a system metric that measures how much the users rely on the recommendations produced by the system; that is, how much they trust that the items being recommended are indeed the best items for them. A number of techniques have been developed and proposed to improve trust in systems. For example, the system can recommend some items that the user has previously rated as good; even though the user doesn't gain anything, his/her trust in the recommended items increases. Another technique is to explain to the user why each item is being recommended, letting him/her know how closely it matches his/her preferences [2].

Due the nature of this property, it cannot be evaluated using offline experiments, and must be measured through user studies and online evaluations. In the evaluations, trust can be measured as either how many recommendations were selected/used by the users, or how many users made use of the system more than once. The latter can only be measured using online evaluations [57] [60] [61].

Novelty

Novelty, as the name suggests, is a metric that measures how new the recommended items are to the user. Novel recommendations are items that the user didn't know about, which in many applications would represent a desired user feature [2]. However, it is important to consider that improving the novelty of recommendations could cause a decline in prediction accuracy, leading to a system that recommends new, but worthless, items to its users [62].

In user studies and online evaluations, novelty can be measured by directly asking the user how novel the recommended items are for him/her. However, in order to measure this property with offline experiments we need some items previously rated by the user, and the time they were rated. This way we can hide items that were rated by the user after a selected point in time, and measure the percentage of them the system recommends [63] [64]. Another technique to measure this property relies on the assumption that popular items are less likely to be novel. Here, novelty is considered by

using an accuracy metric that doesn't give the system the same utility for recommending a popular item as it does when it recommends non-popular items [65].

Serendipity

Serendipity measures how surprising the recommended items are for a user; that is, how much information about a recommended item is new to the user. The more different the recommended items are from those the user rated, the higher the serendipity of the system. Serendipity can also be seen as a deviation from 'natural' system prediction. Thus, as with the novelty measure, improving the serendipity of a system reduces its prediction accuracy [2].

Serendipity can be measured in a system by grouping all the items that are very similar or redundant, then measuring how many of the items recommended to the user are from different groups. The grouping of items can be done either manually, or by applying a data mining clustering technique that groups similar items together. Hence, the system is rewarded for recommending items far from the user profile [66].

Diversity

Diversity refers to how different the recommended items are from each other [2]. The difference between diversity, novelty and serendipity measures is as follows: novelty measures how many recommended items the user was unaware of; serendipity measures how many items that don't match the user profile were selected by the user as 'good ones'; and diversity measures how different the recommended items are from one another, even if they all match the user profile. Thus, as with novelty and serendipity, diversity can also come at the expense of prediction accuracy. Diversity can be measured as the sum, average, min or max distance between pairs of recommended items, and the distance between items can be computed using a distance measure, as shown in Section 2.1.3.

Utility

Utility is defined as the value that either the system or the user gains from the recommendations [2]. The following Sub-Section discusses some techniques and metrics to measure the utility of a system that presents its recommendations as an ordered, ranked list of items, such as our proposed RS (as presented in Chapter 5). For utility metrics in systems that produce other types of recommendations, we recommend [27].

Utility for Ranking-Based Recommendations

The utility of an ordered, ranked list of items is given by the sum of the utilities of each recommendation in it. The utility of each recommendation is given by the utility of the recommended item, discounted by a factor determined by its position in the list. This discounted factor can be considered as the probability that the user will observe each item, based on its position in the list. Therefore, this factor should depend only on the item's position, and not its utility [2]. However, we must also consider how many items are being presented to the user in the recommended lists. In lists where the number of recommended items is small, the utility of the items should decrease faster than in those that recommend a large number of items. Consequently, for scenarios where the system presents a reduced number of items, Breese et al. [27] presented the R-Score metric to assess the recommended list utility. This metric assumes that the value of each item in the list declines exponentially due to its position in the list. The R-Score metric for an ordered ranked list for a user *u* is given by the following equation:

$$R-Score_u = \sum_{i=1}^{N} \frac{\max(u_i - d, 0)}{2^{\frac{position(i)-1}{\alpha-1}}} \qquad (3.43)$$

Here, *N* represents the number of items in the ranked list, u_i is the utility of item *i*, *position(i) is* the absolute position of item *i* in the list, and *d* is the threshold under which items from the sum of the final list utility are excluded. Finally, α represents half of the maximum ranking of an item, to control the exponential decline of the value of positions in the ranked list. However, the final list utility alone doesn't express anything. Therefore, the R-Score per user is given as the percentage it represents compared to the maximum R-Score obtained from all the users in the experiment, as expressed in the following equation:

$$R-ScoreUtility(u) = \frac{R-Score_u * 100}{\max(R-Score)} \qquad (3.44)$$

Alternatively, for scenarios where the system recommends a large number of items, Järvelin et al. [67] presented the *Normalized Cumulative Discounted Gain* (NDCG) measure, by which items' utilities are discounted logarithmically based on their position in the list, rather than exponentially as

in the R-Score measure. The NDCG measure for an ordered, ranked list for a user *u* is given by the following equation:

$$NDCG = \sum_{i=1}^{N} \frac{u_i}{\max(1, log_b position(i))} \qquad (3.45)$$

Here, similar to equation (3.43), *N* represents the number of items in the ranked list, u_i is the utility of item *i*, and *position(i)* is the absolute position of item *i* in the list. The base *b* of the logarithm is a parameter chosen by the user to control how quickly items' utilities should be discounted. An algorithm base 2 ensures that all positions in the list are discounted by a different factor. In the same way as the R-Score, the utility given by the NDCG measure per user is given as the percentage it represents compared to the maximum NDCG obtained from all the users in the experiment, as expressed in the following equation:

$$NDCGUtility(u) = \frac{NDCG_u * 100}{\max(NDCG)} \qquad (3.46)$$

However, unlike other properties such as prediction accuracy, the utility of a system cannot evaluate a system by itself. This is because its value depends on both the recommendation technique being used, and the items included in the database. Therefore, if the database contains only items that are useless to the users the recommended items produced will present a low utility, even if the RS creates an accurate user model. Consequently, this metric is only useful when comparing different RSs implemented over the same dataset of items with the same users.

Risk

This metric measures the risk associated with a recommended item, for domains where it applies. In these cases, a system that produces less risky recommendations may be preferable, even though its utility may not be as high as that produced by other systems [2]. Risk can be evaluated by discounting a 'risk factor' from each recommended item when assessing the system utility [68]. The risk factor should be defined in advance, depending on the application domain and the type of items being recommended.

Robustness

Robustness refers to a system's stability when it is under attack. 'Attack' means fake information inserted intentionally, to either influence the recommendations produced for users, or to learn private information from the stored user profiles. Robustness may also refer to the stability of the system when there are a large number of simultaneous requests [2]. System robustness can be measured by inserting some fake information into the system, then analyzing how much the recommendations are affected. For more information about robustness measures, we recommend [69] [70].

Privacy

Privacy is a measure of how sensitive the recommended algorithm being used is to breaches of privacy [2]. As presented in Chapter 2, in order for a RS to provide accurate recommendations it requires sensitive private information from the users, and it creates a user profile intended to describe and model the user as accurately as possible. Therefore, a RS must ensure that both the user provided information and the created user models won't be disclosed, or used for purposes other than recommending usable items to the user. For information on metrics to measure the privacy in RSs, we recommend [71].

Adaptivity

Some RSs operate in application domains where the collection of items to be recommended to users is constantly changing, or where trends in the interest in items shift rapidly. Thus, adaptivity is a system measure that, when applicable, expresses how well the system adapts to these types of environments. Adaptivity can be measured as the amount of information required by the system, about both users and items, before an item is recommended [2]. Additionally, adaptivity can be seen as how fast the system adapts the user recommendations when user preferences change. This rate can be measured by evaluating the differences between the recommendation lists, before and after new user information is added [72].

Scalability

Scalability is a measure of how well a system would perform in application domains with large collections of items. The scalability of a system can be measured as the computational complexity of its recommender algorithm, in terms of time and space requirements [73]. Scalability also refers to

how fast a system produces recommendations, thus another measure of scalability is the throughput of the system and its latency; that is, the number of recommendations the system generates per second, and its response time, respectively [74] [30].

3.5 Summary

In this Chapter, we presented three different ways to extend the capabilities of traditional Recommender Systems to construct systems that can be implemented in more complex and real-world applications. In Section 3.1, we described how to include multiple criteria in the recommendation process, and presented the UTASTAR algorithm, an extended and improved version of traditional UTA methods. This algorithm adopts the aggregation-disaggregation principle, allowing us to infer decision preference models for specific users by means of linear programming techniques. The preference models are expressed as additive value functions that assign different weights to each criterion, while seeking to be as consistent as possible with user global preferences. We concluded the Section by presenting a summary of the steps to implement the UTASTAR algorithm.

In Section 3.2, we defined context-awareness as the ability to be aware of users' surrounding physical environment and state, and then introduced Context Aware Recommender Systems (CARSs). Unlike traditional two-dimensional (2D) RSs that only consider User and Item information when recommending items to users, CARS systems also consider user context data; that is, information such as location, orientation, temperature, velocity, biometrics, and so forth. We also presented two different approaches to model context information in Recommender Systems: hierarchical structures and OLAP-based multidimensional data models (MD), and three different techniques to incorporate contextual information into the recommendation process: contextual pre-filtering, contextual post-filtering and contextual modeling. These techniques differ from one another at the stage of the process where they consider contextual information. Contextual modeling can be further classified, depending on how it represents the predictive models as heuristic or model-based approaches.

In Section 3.3, we presented a method to generalize user profiles by exploiting the information contained in large-scale demographic databases. This method encompasses the interests and preferences of the people living in a certain region, and has the following advantages: it incrementally and iteratively constructs the user profile; it builds RSs that can operate with a minimal amount of information from the user, and don't require information from other users to be effectively

applied; and, it uses the profile to recommend items from areas where the user has never input information.

Finally, in Section 3.4 we presented the problem of evaluating and comparing RSs, which are difficult tasks because RSs are designed and built based on the specific application domain in which they are going to be applied. Therefore, each of these systems can have different requirements and goals, and exhibit different strengths. We briefly described fourteen different metrics (i.e. RS properties) that can be assessed in RSs in order to evaluate and compare them. These are user preference, prediction accuracy, coverage, utility, confidence, trust, risk, robustness, adaptivity, serendipity, diversity, novelty, privacy and scalability. In addition, we introduced offline experiments, user studies and online evaluations, as three types of experiments that can be conducted to evaluate each of the properties. These experiments differ in the required interaction with the user, the implementation cost, and the types of system properties that can be measured. Offline experiments due they don't require user interaction, are the easiest to be apply, and have the lowest implementation cost. User studies, on the other hand, can be used to measure the widest quantitative and qualitative set of system properties, though they are also the most expensive to implement. Online evaluations are the only experiments that allow us to determine how much the system influences users' behaviour, or improves the long-term system goals. Regardless, this type of experiment can only measure a few system properties.

The techniques presented within this Chapter allow RSs to address specific problems or challenges (i.e. consider multi-criteria, include context information, or generalize user profiles). However, RSs may need to combine these techniques in different ways to achieve real world application goals. Chapter 4 presents a number of RS's implementations that use and combine some of the techniques introduced here.

Chapter 4
Recommender System's Implementations

A number of different approaches have been proposed to create RSs. They combine different methodologies and techniques in an effort to create workable real world applications and provide more accurate recommendations. This Chapter presents previous works from the literature that differ from one another in the techniques they implement, and the type of data used to recommend items to users. Some of these techniques presented, include application of the UTA* algorithm over multi-criteria ratings, the use of large scale demographic databases to generalize the user profiles, the creation of CARSs, and the application of multiple data mining algorithms (e.g. clustering and classification techniques).

4.1 Multi-Criteria User Modeling in Recommender Systems

Lakiotaki et al. [11] proposed a framework that combines techniques from multi-criteria decision analysis and collaborative filtering approaches. The proposed method constructs user profiles from user preferences information, and integrates them as a user preference model (i.e. user value system). It then uses the preference model to identify common users and produce recommendations. Creating a user preference model enables the system to identify how users think about items, by considering the attributes that influence a user to choose a particular item. Consequently, the system has a more sophisticated understanding about the user, as it is able to recognize preferences and not just patterns.

4.1.1 Methodological Framework

The proposed framework shown in Figure 4.1 depicts the phases or steps identified and presented by Lakiotaki et al. [11] as follows.

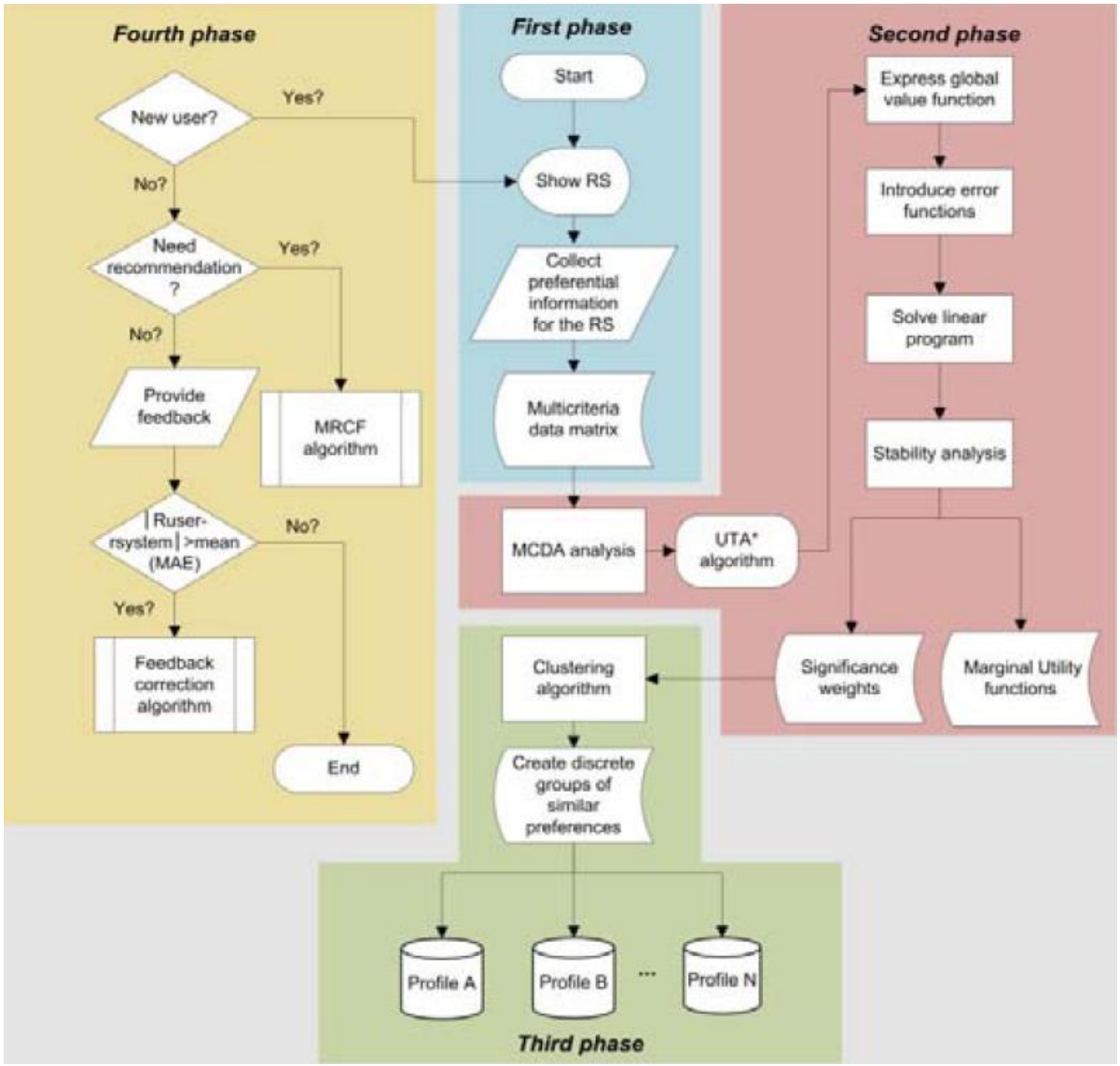


Figure 4.1: Framework for multi-criteria user modeling [11]

1) Data acquisition. The system first collects two types of data: a numerical rating representing the preference data over each criterion, and a ranking order representing the user global preference order (also called the weak preference order). Every user $u \in U$ is asked to evaluate a set of items $A_i \in A_R$, called reference set A_R. Using a predefined scale (i.e. 1 to 5), the user provides a rating r_{ui} for every

criterion $c_j,\ j = 1,2,..,k$, where k is the total number of criteria. In addition, the user ranks all the items that belong to the reference set in descending order, to provide the weak preference order [11].

2) Multi-criteria user modeling. In this phase, the UTA* algorithm (shown in Section 3.1.1) is applied over the input data acquired from phase 1 for each user, to obtain preference models from the users' preferential structures. Since the final output of the system will present the *N* top items for the user, the application of the UTA* algorithm corresponds to the ranking problem statement (one of the application problem statements identified for UTA methods). As previously explained, the UTA* algorithm returns a set of value functions associated to each criterion, approximated by linear segments, as well as the criteria significance weights for each user (known as the user weight vector). The criteria significance weights for each user serve as the users' preference models [11].

3) Clustering. In this phase, the Global K-means clustering algorithm is applied over the set of user preference models obtained from the second phase. The Global K-means algorithm is a variation of the well-known k-means clustering algorithm, but it does not depend on initial parameters (e.g. *k* number of clusters, initial centroids). The algorithm acts incrementally to select the initial values for all cluster centers, optimally adding one new cluster centre at a time until a specific clustering criterion is reached. That clustering criterion could be the number of clusters, or the minimum *Sum of Squared Error* (SSE) between each data point and the centroid of the cluster that the point belongs to. Applying this algorithm over the set of user preference models gives a classification of each user (represented by his/her preference model), within one of the identified groups or clusters [11].

4) Recommendation phase. Once the users are clustered based on their preference models, the system applies the multidimensional *Multi-criteria Collaborative Filtering* (MRCF-dim) method inside each cluster. The MRCF-dim technique consists of the following steps: First, calculate the distance between two users u and u' for the same item, using their rating vectors r_u and $r_{u'}$ respectively. 'Rating vector' refers to the vector of ratings of dimension *k* that user *u* provided for an item *i* including the global rating for the item and the rating of each criterion for that item. The distance between users is computed using the Euclidian distance, as follows:

$$d_{uu'} = \sqrt{\sum_{n=1}^{k} (r_{un} - r_{u'n})^2} \qquad (4.1)$$

Second, determine the overall distance between two users u and u'. The overall distance between two users is obtained from the average distance between their ratings for all their common items. This distance is computed using the following equation:

$$dist(u,u') = \frac{1}{|U(u,u')|} \sum_{i \in U(u,u')} d_{uu'} \quad (4.2)$$

where *U(u,u')* denotes the set of items that both users have rated. The third step is finding the similarity between the two users, which is inversely related to their distance, and obtained by the following equation:

$$sim(u,u') = \frac{1}{1 + dist(u,u')} \quad (4.3)$$

It is important to note that the similarity between pairs of users considers only the users within each cluster, and where both users have similar items rated in the past. Therefore, the computation required is minimal compared to non-clustering approaches that obtain this similarity for all possible pairs of users. Once the similarity between users of the same cluster is obtained, the predicted rating of an item for a user is given by the following equation [11]:

$$R(u,i) = \left(\frac{1}{\sum_{u' \in C(u)} sim(u,u')} \right) \bullet \sum_{u' \in C(u)} sim(u,u') \bullet R(u',i) \quad (4.4)$$

where C(u) represents the users within the same cluster.

Feedback mechanism. The user has the option of providing a different rating than the one predicted by the system for a specific item. When a user does this, the system compares the two ratings, and then compares the absolute difference between the ratings against the mean of differences between predicted ratings and stored user ratings for previously rated items. If this difference is larger than the mean the new item rating is stored, and both UTA* and Global K-means algorithms are run again so the user may be assigned to a different cluster [11].

According to the comparison experiments, and the results presented by Lakiotaki et al. [11], the proposed technique performs better than other single and multi-rating RSs, due to two main factors: the creation of groups of users prior to the application of collaborative filtering algorithms, and the fact that the clustering is done using user preference models, obtained by MCDA techniques.

4.2 Intelligent User Profiling Using Large-Scale Demographic Data

Krulwich et al. [52] presented an approach for generating user profiles using large-scale demographic databases to generalize the data specified by the user, in order to include areas not represented in the user's original input. The methodology uses the user input data to classify the user within one or more demographic groups identified in a demographic database. Thus, the final user profiles include all the information contained in the database.

The system computes the set of possible demographic clusters a user could belong to. In the case of only one matching cluster, all the data of that cluster is used as the user profile. If the user could belong to more than one cluster, the system produces a partial user profile containing all the similar information between all possible clusters. The difference in the data between possible clusters can later be used to obtain further information from the user, in order to narrow down the number of clusters. The system does not use a clustering algorithm to select the demographic groups the user could belong to. Instead, it treats each user input as a constraint on the values of the demographic cluster variables. Based on this, it computes the probability of that user to belong to each of the demographic clusters; the matching clusters are the ones with a probability higher than a selected threshold [52]. Finally, the recommended items are those identified by the demographic database for all possible matching clusters. This method, and its advantages, are presented in Figure 3.10 and Section 3.3.1, respectively.

This methodology is a technique to create and use user profiles that combines the benefits of both the content-based and collaborative-based approaches. Although comparative experiments against other recommender techniques conducted by Krulwich et al. [52] show that this method results in a slight decrease in recommendation accuracy, it is still considered effective [52]. Since it exploits the information in large-scaled demographic databases, the method is more suitable for applications where users are not willing to provide large amount of information or personal data.

Finally, it is important to note that this approach can only recommend those items identified by the demographic database for each demographic cluster. Therefore, in order to recommend other types of items not included in the database, a link between clusters and items is required. The link can either be created manually, based on the information in the database, or implicitly observed from the collected data of items selected by users of different clusters.

4.3 Client Side Mobile User Profile for Content Management using Data Mining Techniques

Paireekreng et al. [10] proposed a framework for building a user profile at the user's mobile device, using clustering and classification data mining algorithms, and considering user context information. This user profile can later be used by a mobile internet provider, to present the user with different web content.

4.3.1 Methodological Framework

Applying this technique requires an initial set of previously collected data containing the user's preferences for mobile internet content (e.g. multimedia, news, information). The information included about the users is both demographic data (e.g. gender, age, occupation, income) and user context data. The context data considered only includes the type of information the user is looking for at a specific moment (e.g. important or up-to-date information).

The framework consists of four steps. First, divide the collected data, based on the context information, between the users' content preferences when they are searching for important information, or when they are looking for up-to-date information. Second, apply a clustering algorithm over both sets of data, in order to create groups or clusters of users with similar content preferences in the same context. Third, name each cluster according to the information it contains. Finally, apply a classification data mining technique over both sets of data, using the cluster names as class labels [10].

Classification is a data mining concept that predicts the class to which data instances belong [1]. To do this, the classifier first learns the characteristics of the data and produces a representative model from a training set, where the classes for each data instance are given. Subsequently it uses the model to predict the class for the instances of another unclassified dataset [22]. For more information on classification algorithms, we recommend [1].

Once the classification models are built, they can be used to categorize a user in one of the previously identified clusters. The classification model to be used and clusters to be considered depend on the type of information the user is looking for (i.e. user context). The content presented to the user by the content provider is the content preferences of the users within the matching cluster. This process is summarized in Figure 4.2.

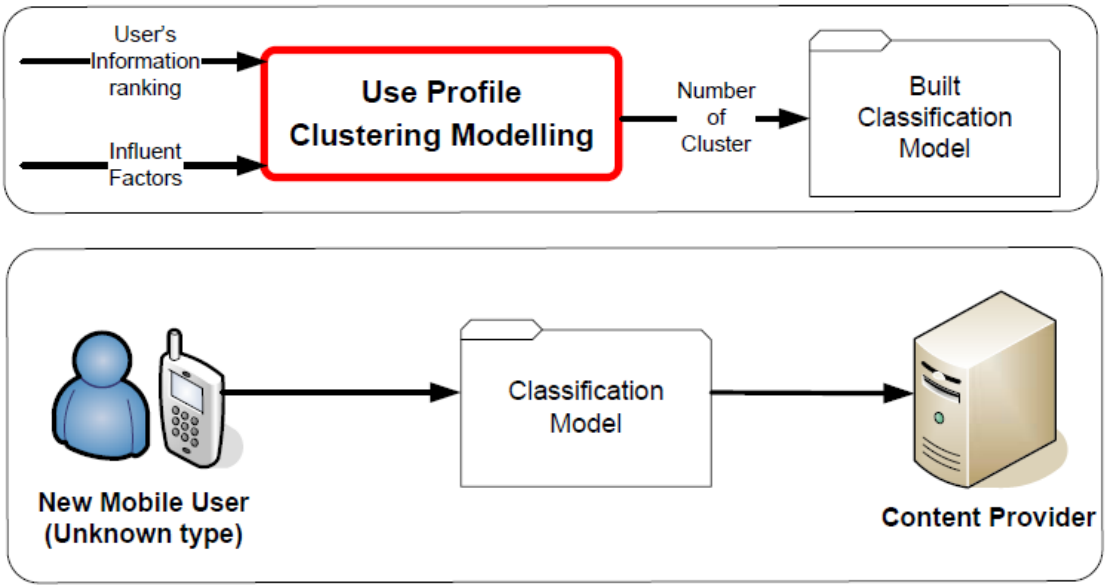


Figure 4.2: Process followed to provide content personalization [10]

It is important to remember that this framework was developed to be executed on the user's mobile device, using the k-means clustering and RepTree classification algorithms. Based on how the technique is implemented, the framework could be considered as a contextual pre-filtering CARS (as presented in Section 3.2.2), since it first considers the context information to filter out data that doesn't match the current user context.

4.4 Applying User Profile Ontology for Mining Web Site Adaptation Recommendations

Robal et al. [75] proposed a framework to create a recommender engine that suggests improvements to adaptive websites, based on users' locality models (i.e. user profiles) and website ontologies. This technique is meant to overcome the problem of finding the information we want in the vast amount of information on the Internet. The goal is to help users discover pages that they otherwise might not find during their web browsing. The proposed methodology collects and analyzes user interaction data during web browsing sessions, to assist users by either recommending related links, or continually adapting the website to their needs.

Adaptive websites are those that are modified according to their use, either strategically (i.e. adaptations have a serious impact on the site structure) or tactically (i.e. adaptations do not interfere with the site structure and can be done in real time). In order to represent the information on the websites, the use of website ontologies is proposed [75]. Website ontologies represent the thematic

categories covered by the pages of the website, where each page is an instance of one or more concepts. These, in turn, can be arranged in a hierarchal structure, using "is-a" relationships [76].

4.4.1 Methodological Framework

Applying this methodology first requires the collection of a large amount of data about users' interactions during their web browsing sessions. A log system, proposed by Robal et al. [77], was used to gather this data. The proposed log system has the ability to capture distinct and recurring user sessions, and store information about requested pages, session IDs and operations performed during the sessions. Once user interaction raw data is stored, a locality model is applied, based on the belief that if a large number of users frequently access a particular set of pages, these pages must be related. A locality is defined as users' most recent activity history on the site. Thus, during web sessions users are moving from one locality to another (i.e. performing different sequences of activities). The localities that are most frequently presented for a specific user represent the user profile, and each user profile must be manually mapped to the web ontology [75].

This Recommender System relies on the idea that a user would normally use his/her own domain or topic knowledge to perform a search. However, he/she might not know how to represent it [78]. Analyzing the actions of a current user makes it possible to classify him/her into one of the previously identified user profiles, by comparing his/her actions to the localities of each profile. Once a user profile is identified, the system recommends new pages that correlate with that profile (i.e. tactical adaptation), or proposes a new website topology (i.e. strategic adaptation). The pages are also ranked using an inverse time weight equation (4.5), where *Age(i)* represents the number of days in the past, *Interest value(i)* is the number of hits per page during *Age(i)*, and *p* is a predefined probability between 0 and 1. Thus, the most recent page hits have more weight in the page ranking [75].

$$Rank = p\sum_{i=1}^{n}\frac{Interest\,value(i)}{Age(i)} \qquad (4.5)$$

To summarize, the recommender engine classifies the user into a user profile based on his/her current actions, and provides recommendations. The recommendations are based on the web ontologies, mapped to the user profile and the page rankings. The proposed framework is shown in Figure 4.3.

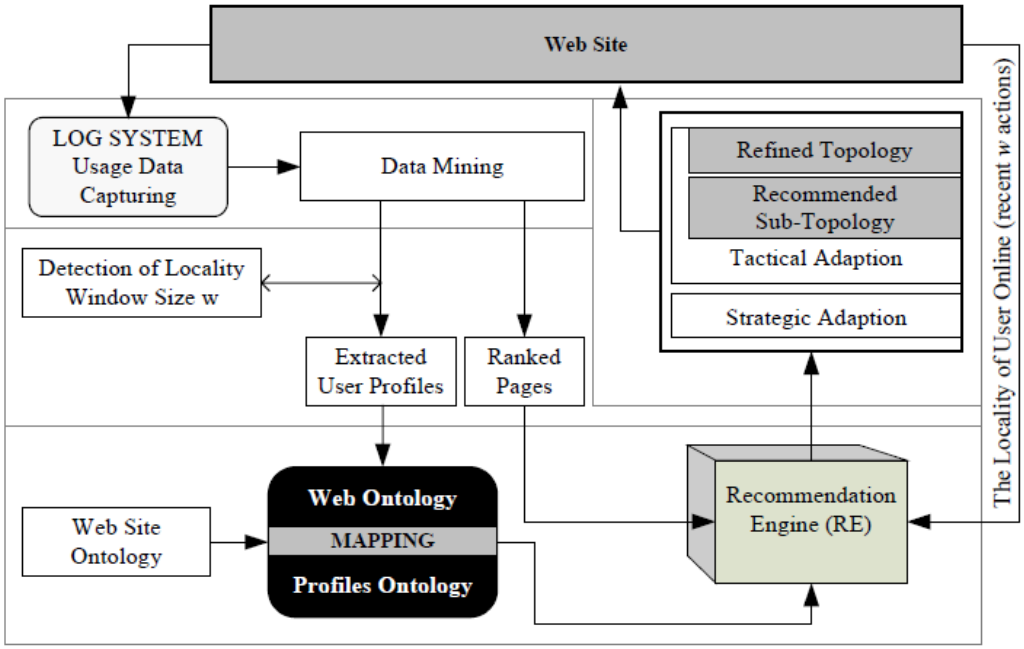


Figure 4.3: Adaptive website recommendations based on extracted user profiles [75]

4.5 Comparison and Discussion

The previously presented frameworks, methodologies and techniques proposed to create RSs, are intended to be applied under specific application domains using specific types of data. However, they all follow the general framework for RSs presented in Section 2.1: First, obtain some information from the user, then create or extract a user profile and, finally, using different recommendation algorithms and techniques, recommend items to the user. In this Section, we present a comparison of these different approaches, and in Chapter 5 we propose a new technique that combines the advantages of some of them. Table 4.1 compares the methodologies, advantages and disadvantages of the previously proposed approaches.

The Multi-Criteria technique (the first technique presented) has the unique advantage of considering multiple decision criteria to create a user preference model. Rather than simply finding patterns in user behaviour, this model is intended to recognize the reasons why a user selects a particular item over others. Furthermore, it also has the advantage of reducing computational complexity, since the collaborative algorithm is applied considering only the users in each cluster.

Approach	Proposed Methodology	Advantages	Disadvantages
1) Multi-Criteria user Modeling in Recommender Systems [11]	1) User rates some items under different criteria, and provides a global preference rate for each item. 2) Creates user preference models using the UTA* algorithm. 3) Clusters the users based on their preference models. 4) Applies the MRCF-dim algorithm to create a model to predict the rating for an item in each cluster.	➢ Combines MCDA and Collaborative filtering techniques. ➢ Considers multiple decision criteria. ➢ Creates a User Preference model to better understand the user. ➢ Reduces computational complexity.	➢ The user must rate a large number of items under several criterions. ➢ Requires information from a large number of users. ➢ Can only recommend items previously rated by users in the same cluster. ➢ The user may belong to only one cluster.
2) Intelligent User Profiling Using Large-Scale Demographic Data [52]	1) User answers some previously selected questions. 2) Clusters the user into one or more demographic groups, computing the probability for a user to belong to each of them. 3) Demographic clusters with a probability higher than a selected threshold are used as matching clusters. 4) The commonalities between all matching clusters are used as user profiles. 5) Recommendations are done using all the information of the matching clusters, and their probability.	➢ Considers the preferences of all similar users. ➢ Can recommend items of areas where the user has never input information. ➢ Does not require information from a large number of users, or that the user answers huge surveys. ➢ Can operate incrementally.	➢ The recommendations are not as accurate as they would be if only using the user given data. ➢ Can only recommend items included or covered in the demographic database.
3) Client side mobile user profile for content management using Data Mining Techniques [10]	1) Requires previously gathered information about user content preferences. 2) Divides the information into two sets, based on the user context. 3) Applies a k-means algorithm over both sets to group users with similar content preferences. 4) Applies the RepTree algorithm to obtain a classification model, depending on the context. 5) Classifies the user into the corresponding cluster, using the appropriate classification model. 6) Recommends the preferences of the users in the matching cluster.	➢ Considers the user current context. ➢ Produces different recommendations, depending on what the user is interested in at the moment	➢ Requires a large amount of data about users' content preferences. ➢ The user may belong to only one cluster. ➢ Can only recommend contents previously selected by users in the same matching cluster.
4) Applying User Profile Ontology for Mining web Site Adaptation Recommendations [75]	1) Requires previous data about user web browsing sessions. 2) The most frequent sequences of visited pages represent the user profiles. 3) Each user profile is manually mapped to the website ontology. 4) Analyzes the user actions to classify him/her into one user profile. 5) All pages are ranked based on the number of hits they have received. 6) Recommendations are based on the web ontologies mapped to the corresponding user profile and the page rankings.	➢ Doesn't require information from the user. ➢ The identified user profiles are constantly changing.	➢ Requires a large amount of information. ➢ Can only recommend pages previously visited by users in the same user profile. ➢ The user may belong to only one user profile.

Table 4.1: Comparison of previous works on RSs

However, the main disadvantage of the Multi-Criteria technique is that it requires a large number of users who have already rated several items under different criteria, in order to have enough samples of user preference models to create the clusters. Similarly, the mobile and website methodologies have the same disadvantage of requiring a large number of previously gathered data to be successfully applied.

The demographic methodology (the second technique presented), is the only one that can be directly applied using only partial user information, without the need of huge user surveys or analysis of gathered data from previous users interactions. The advantage of this approach is that it can recommend items to the users from areas where the user has never input information. However, since the user profiles are obtained from generalization of the user input, the accuracy of the recommendations could be decreased. Moreover, this technique also has a major disadvantage; if we want it to recommend items that are not covered or included in the demographic database we need to manually map all possible items or item types to each demographic cluster.

The mobile system (the third technique presented) is the only one that considers the current user context, by creating different models and applying the one that corresponds to what the user is looking for. Therefore, it can produce different recommendations for the same user, depending on the type of information the user is interested in at the moment. However, in addition to the already mentioned problem of requiring a large number of previously collected data to create the clusters, another disadvantage of this approach, as with the multi-criteria and web site techniques, is that recommending an item (i.e. objects, internet contents or pages to visit next), requires that the item has already been rated by other users of the same cluster to which the user belongs.

Finally, the web site algorithm (the fourth technique presented) is the only one that produces recommendations without asking the user information; it collects all the required data implicitly by examining user actions. Since it gathers data in this manner, this approach can continuously improve the user profile and produce more accurate recommendations, as the demographic technique does by asking for more information from the user. Regardless, in addition to the already mentioned disadvantages of the requirement for previously collected data and the limitation that an item must be already rated in order to be recommended, this methodology, like the multi-criteria and demographic approaches, has the disadvantage that every user can only be classified to a single cluster or user profile, rather than being able to leverage commonalities between different clusters or profiles.

4.6 Summary

In this Chapter we presented four RS's implementations that use and combine different techniques to extend their recommendation capabilities in order to address real world application requirements. We start describing in Section 4.1, a multi-criteria RS that, by means of the UTA* algorithm, creates user preferences models. The users are then clustered considering their preference models, instead of their personal information. Finally recommendations are generated applying the collaborative MRCF-dim algorithm within each cluster.

In Section 4.2 we introduced a RS that, using a demographic generalization method, generalizes the users given data to incrementally produce user profiles. Recommendations are produced considering the common information contained in the clusters of the demographic database, identified as possible groups where each user may belong, along with their probability of belonging to each of them.

In Section 4.3, we presented a RS that divides the users' stored information depending on what they were interested at that moment (i.e. users' context). The users in each group are then clustered based on their preferences, and a classification model is created to predict the cluster where a new user may belong. The recommended items are the ones preferred by the users in the same cluster. The classification model used to cluster a new user depends on the user current context. Therefore the same user may belong to different groups depending on his/her context and consequently receive different recommendations.

In Section 4.4, we described a RS that personalizes websites for each user, based on his/her current browsing actions. The system analyzes the user actions to classify him/her into a user profile. User profiles (i.e. user browsing patterns) are discovered from historical web browsing logs. Additionally all pages are ranked based on the number of times a user has visited them. Therefore, websites are updated considering the selected user profile and page rankings.

Finally, in Section 4.5 we presented a comparison between these RS's implementations, by means of their proposed methodology, advantages and disadvantages. This comparison may be used as a user guide to select the best RS implementation to be applied, depending on the application domain, and available sources of information.

Based on the recommendation techniques, advantages and limitations of the RS's implementations as presented within this Chapter, we introduce a methodology to create RSs that aims to combine their advantages and overcome their current limitations. This methodology is presented, in details, within the following Chapter.

Chapter 5
Methodology

We introduce a new methodology for RSs that follows the general structure for RSs presented in Section 2.1. It combines the advantages of previous approaches, and overcomes some of their disadvantages. The disadvantages presented in previous RS's implementations (as presented in Chapter 4), and overcome by our methodology are the following: most implementations consider only single criteria ratings, and consequently are unable to identify why a user prefers an item over others; or they classify the user into one single group or cluster which is an unrealistic approach, since in real world users share commonalities in different degrees with diverse types of users; or they require a large amount of previously gathered data about users interactions and preferences, in order to be successfully applied.

The RS obtained by following our methodology can be classified as a **Personalized Multi-Criteria Context-Aware Hybrid Recommender-System**, with the following noteworthy characteristics. It considers the user's current location (i.e. location aware) to recommend items (i.e. CARS, presented in Section 3.2). It creates a user preference model that considers multi-criteria to assess the utility of each item (i.e. multi-criteria, presented in Section 3.1), obtained by applying the UTA* algorithm (presented in Section 3.1.1). It considers the user context during the recommendation process when creating the user preference model (as presented in Section 3.2.2). It combines collaborative-based and content-based techniques (i.e. Hybrid RSs, presented in Section 2.1.3). Collaborative-based techniques (as presented in Section 2.1.3), are used to generalize the user profile from a large-scale demographic database, using the demographic generalization method (as presented in Section 3.3.1). Content-based techniques (as presented in Section 2.1.3) are used to obtain items with similar desired features.

5.1 Steps in the Methodology

Our methodology was created to achieve a number of objectives. We consider the user context so a user can receive different recommendations depending on his/her current situation. We identify the user preferences that lead him/her to select an item, and correspondingly potentially generate more accurate recommendations. Further, we aim to model and consider the user context during the recommendation process, instead of using it as a filter only (i.e. contextual modeling approach); and finally, our goal is to produce accurate recommendations without the need of gathered data from previous users interactions.

Our methodology for RSs is based on the creation of a multiple criteria user preference model considering the following four criteria: the probability that a user prefers an item, given the probabilities that the user belongs to each of the demographic clusters identified in a large-scale demographic database; the weight of each item, based on its attributes and the weights the user assigns for each attribute; the distance between the user's current location and the item's location; and, the time the user would require to reach each item, based on his/her mobility level (e.g. driving, walking, using the bus), geo-spatial constraints and average route times.

The user preference model is then used to assess the utility of each item for the user. Finally, the system will recommend items with an estimated utility higher than a selected threshold to the user. Figure 5.1 illustrates a high-level model of this methodology.

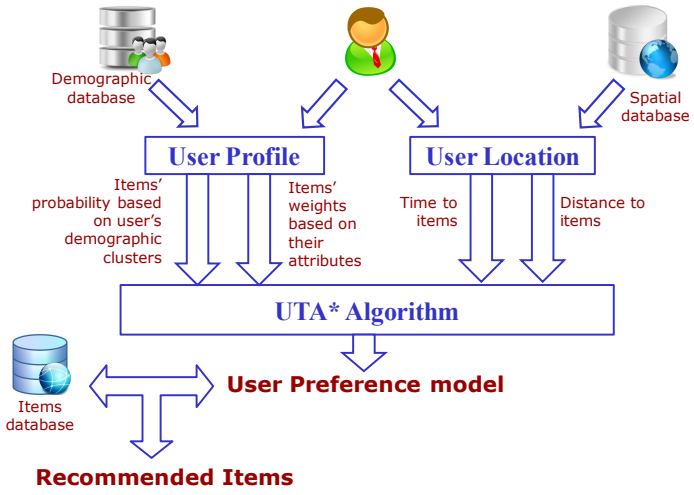


Figure 5.1: High level model of the proposed methodology

This methodology is comprised of the following phases, as presented in the Algorithm 5.1. First, create the user profile. Second, identify the user location, mobility level, geo-spatial layout and

special needs. Third, obtain the probability and weight for each item, using the previously created user profile (i.e. apply the user profile to assess the items). Fourth, obtain the time and distance from the user to each item, using the previously identified user location and mobility level (i.e. apply location aware to assess the items). Fifth, create the user preference model. Sixth, assess the utility for each item using the obtained model, and present the items with the highest predicted utility to the user.

Each one of these phases is presented in detail in the following sections.

Our Recommender System Algorithm

Requirements:

- User (*u*).
- Items database.
- Demographic database.
- Geo-Spatial database.

Output:

- Ordered ranked list of recommended items for user *u*.

Algorithm:

1. Create the User Profile (algorithm 5.2).
2. Obtain the User Location and Mobility Level (algorithm 5.3).
3. Obtain the Demographic and Weighted Utilities for all items (algorithm 5.4).
4. Obtain the Distance and Time Utilities for all items (algorithm 5.5).
5. Create the User Preference Model (algorithm 5.6).
6. Recommend the Items that Best Match the User Preferences (algorithm 5.7).

Algorithm 5.1: Our Recommender System

5.1.1 Create the User Profile

The first phase of the methodology is to create the user profile. Our user profile is made up of the following two dimensions: the set of all possible demographic clusters the user could belong to and the probability of belonging to each of them; and how much the user cares about each item attribute expressed as user given weights to each of them. The first dimension allows us to understand who the user is, and to generalize the information about his/her using the data covered by the demographic database. The second dimension deals with the user specific preferences about items, identifying which item characteristics are more important for the user when selecting an item over others. Therefore, the user profile *profile(u)* for a user *u*, is expressed as:

$$profile(u) = \begin{cases} DemogClusters(u) \\ AttributeWeights(u) \end{cases} \quad (5.1)$$

where *DemogClusters(u)* represents the set of all possible demographic clusters and the probability of belonging for user *u*, and *AttributeWeights(u)* represents the set of all different item attributes and the weights assigned by user *u*.

Obtaining the Possible Demographic Clusters

The set of possible demographic clusters to which a user may belong, and their probability of belonging, is expressed as a vector of pairs of the form $< C_x, P(u, C_x) >$. C_x represents the demographic cluster $\boldsymbol{x}$, and $P(u, C_x)$ the probability that the user *u* belongs to cluster *x*. Therefore, the complete set of possible demographic clusters *DemogClusters(u)* for user *u* is given by the following equation:

$$DemogClusters(u) = \{< C_1, P(u, C_1) >, < C_2, P(u, C_2) >, \ldots, < C_k, P(u, C_k) >\} \quad (5.2)$$

Here, *k* represents the number of possible clusters the user could belong to. Thus, the value of *k* must be within the range of values *{1,…,N}*, *N* being the total number of demographic clusters identified in the demographic database.

The first requirement to obtain this vector is to find all the possible demographic clusters to which the user could belong. This is achieved by applying the following proposed technique, which is based on the demographic generalization method presented in Section 3.3.1. The technique consists of the following stages: first, select the *m* demographic variables (e.g. age, language, gender) that best differentiate each cluster from the others using feature selection techniques; second, ask the user the relevant questions to obtain his/her information for the *m* selected demographic variables; third, using the user provided demographic information as constraints, select the *k* demographic clusters in which the number of matching demographic data is higher than a selected threshold. The *k* selected clusters form the set of possible demographic clusters *PossibleClusters(u)* that user *u* could belong to. This technique is expressed by equation (5.3).

$$PossibleClusters(u) = \left\{ C_x : \left| \left[DV_j : Val(DV_j, C_x) = Val(DV_j, u) \right] \forall 1 \le j \le m \right| \ge Threshold \right\} \forall 1 \le x \le N \quad (5.3)$$

Here, *DV* stands for *Demographic Variable*, $Val(DV_j, C_x)$ represents the value of the demographic variable *j* in cluster *x*, and, similarly, $Val(DV_j, u)$ represents the value of the demographic variable *j* given by user *u*. Hence, equation (5.3) represents the subset of clusters in which the number of variables with a value equal to that provided by the user is higher than a selected threshold. The clusters are selected from the *N* demographic groups included in the database, considering only the *m* previously selected demographic variables.

Another technique to select the possible demographic clusters, once the user has entered his/her selected demographic information, is based on the belonging probabilities for each cluster presented in some databases. The belonging probabilities represent the probability that a user could belong to each cluster based on his/her demographic information for each demographic variable. These probabilities are expressed as follows:

$$P_{u,C_x}\left(DV_j = Y\right) = Z\% \qquad (5.4)$$

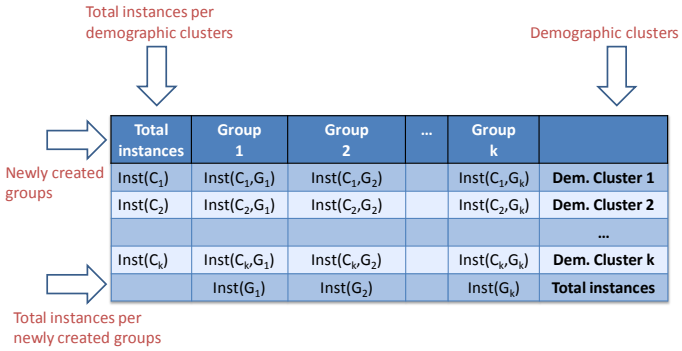
Equation (5.4) shows that there is a *Z%* probability that user *u* belongs to cluster *x*, given that the value of the demographic variable *j* for that user is *Y*. This means we can use these probabilities to select the possible demographic clusters the user could belong to. Thus, the possible demographic clusters (as expressed in equation 5.5) are those in which the product of the belonging probabilities for the user given demographic information is higher than a selected threshold.

$$PossibleClusters(u) = \left\{ C_x : \prod_{j=1}^{m} P_{u,C_x}(DV_j = Val(DV_j, u)) \geq Threshold \right\} \forall 1 \leq x \leq N \qquad (5.5)$$

After the possible demographic clusters a user could belong to have been selected, the probability already obtained from equation (5.5) can be considered the probability of belonging to each of them. However, this would mean classifying the user based only on his/her demographic information, and not considering other non-demographic data that might also be in the database (e.g. information about leisure, shopping, media preferences). Therefore, for databases where this type of non-demographic information is included, we propose the following technique to obtain the probabilities of belonging to each of the previously selected possible demographic clusters.

Obtaining the Probability of Belonging for each Demographic Cluster

This technique is implemented in the following stages. **First**, obtain the Cartesian product of the non-demographic information included in the database for each of the previously selected clusters (i.e. Leisure X Shopping X Media). **Second**, add a new attribute to keep track of which demographic cluster was obtained each instance of the Cartesian product. **Third**, apply the k-means data mining clustering algorithm over the Cartesian product dataset. The clustering algorithm is configured to learn the newly added attribute (i.e. demographic cluster), and to create the same number of groups as previously selected possible clusters (*k* groups). We propose to use the k-means clustering algorithm since we are working with noise-free datasets (i.e. PRIZM C2), and because the data to be clustered is uniformly distributed. Thus, we expect to have clusters of similar shape and size, and no missing or incorrect values in the data. For more information on the k-means clustering algorithm, we recommend [79] [80] [1]. Note that hereafter, we use the term "demographic cluster" to refer to the demographic clusters identified by the demographic database, and the term "group" to refer to the clusters obtained by the k-means algorithm. The results of the k-means algorithm are k groups, each labeled by the most representative non-demographic information it contains. The k-means algorithm also creates a table, with the distribution of instances from each demographic cluster classified into the newly created groups. Table 5.1 shows the general structure of the table resulting from applying the k-means clustering algorithm.



Total instances	Group 1	Group 2	...	Group k	
Inst(C_1)	Inst(C_1,G_1)	Inst(C_1,G_2)		Inst(C_1,G_k)	**Dem. Cluster 1**
Inst(C_2)	Inst(C_2,G_1)	Inst(C_2,G_2)		Inst(C_2,G_k)	**Dem. Cluster 2**
					...
Inst(C_k)	Inst(C_k,G_1)	Inst(C_k,G_2)		Inst(C_k,G_k)	**Dem. Cluster k**
	Inst(G_1)	Inst(G_2)		Inst(G_k)	**Total instances**

Table 5.1: Instances from demographic clusters classified in the newly created groups

In Table 5.1, $Inst(C_x)$ represents the total number of instances belonging to the demographic cluster *x*, $Inst(G_y)$ represents the total number of instances assigned to the newly created group *y*, and $Inst(C_x,G_y)$ represents the number of instances from the demographic cluster *x* that were assigned to the newly created group *y*.

Fourth, present the user with the obtained newly created groups, so he/she can select the one or more groups that best define him/her, based on the non-demographic information contained in each.

Since the user is asked to select one or more from the *k* obtained groups, the number of user selected groups *p* is within the range of *{1,...,k}*.

Fifth, obtain the belonging probabilities to each of the *k* demographic clusters, using the following equation:

$$P(u,C_x)\forall 1 \le x \le k = \begin{cases} \text{If } \sum_{j=1}^{p} Inst(C_x,G_j) = 0 : 0 \\ \text{Otherwise} : \frac{1}{p}\sum_{j=1}^{p}\left[\frac{Inst(C_x,G_j)*100}{Inst(G_j)}\right] \end{cases} \quad (5.6)$$

Since we are working with probabilities expressed as percentages, the sum of all the belonging probabilities for the *k* possible demographic clusters must equal 100%, as expressed in the following equation:

$$\sum_{i=1}^{k} P(u,C_i) = 100\% \quad (5.7)$$

With the belonging probabilities obtained, both the vector *DemogClusters(u)* for user *u* (as shown in equation (5.2)), and the first dimension of the user profile (as presented in equation (5.1)), are completed. Figure 5.2 presents the previously described technique to obtain the vector of demographic clusters where the user could belong, along with the belonging probabilities.

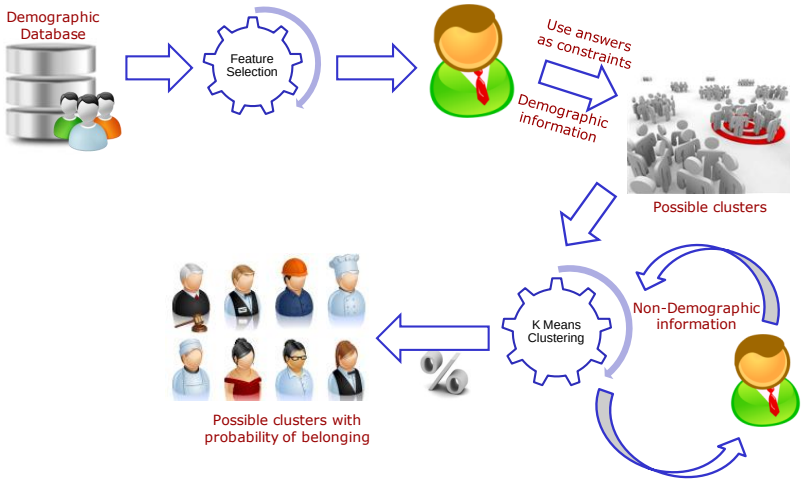


Figure 5.2: Obtaining the possible demographic clusters where the user may belong

The following Section presents the proposed methodology to obtain the second dimension for the user profile, which refers to obtaining the user weights for all possible items' attributes.

Obtaining the User Weights for all Items' Attributes

The second dimension of the user profile (as expressed by equation (5.1)) models how much the user cares about each item characteristic when selecting (or preferring) one item over the others. This set of user given weights is expressed as a vector of pairs of the form $< A_x, W(u, A_x) >$. A_x represents the item's attribute *x*, and $W(u, A_x)$ is the weight that user *u* assigns to attribute *x*. Therefore, the complete set of user assigned weights, *AttributeWeights (u)*, for user *u* is given by the following equation, where *r* represents the total number of items' attributes:

$$AttributeWeights(u) = \{< A_1, W(u, A_1) >, < A_2, W(u, A_2) >, \ldots, < A_r, W(u, A_r) >\} \quad (5.8)$$

We propose the following technique to obtain the vector of user weights for all items' attributes. First identify all the possible items' attributes or characteristics that define them, by analyzing every possible item to be recommended. Subsequently present the user with the list of different items' attributes, in order for he/she to assign weights to each of them. The weights must be assigned based on how important each attribute is for him/her when selecting one item over the others, using integer values within the range $\{-\infty, \ldots, 0, \ldots, \infty\}$. A weight of *0* means the user doesn't care about that attribute at all, a positive weight represents how much the user likes or prefers that attribute, and a negative weight represents how much the user dislikes that attribute.

Once the user assigns weights to each identified item's attribute, the vector *AttributeWeights(u)* for user *u* (as presented in equation (5.8)), and the user profile (as presented in equation (5.1)), are complete. Figures 5.3 and 5.4 depict this technique to obtain the user weights for each item's attribute, and how the user profile conforms, respectively. Algorithm 5.2 presents this methodology to create the user profile.

The following Section presents the phase after the user profile has been obtained, considering the location and mobility level of the user (i.e. location aware).

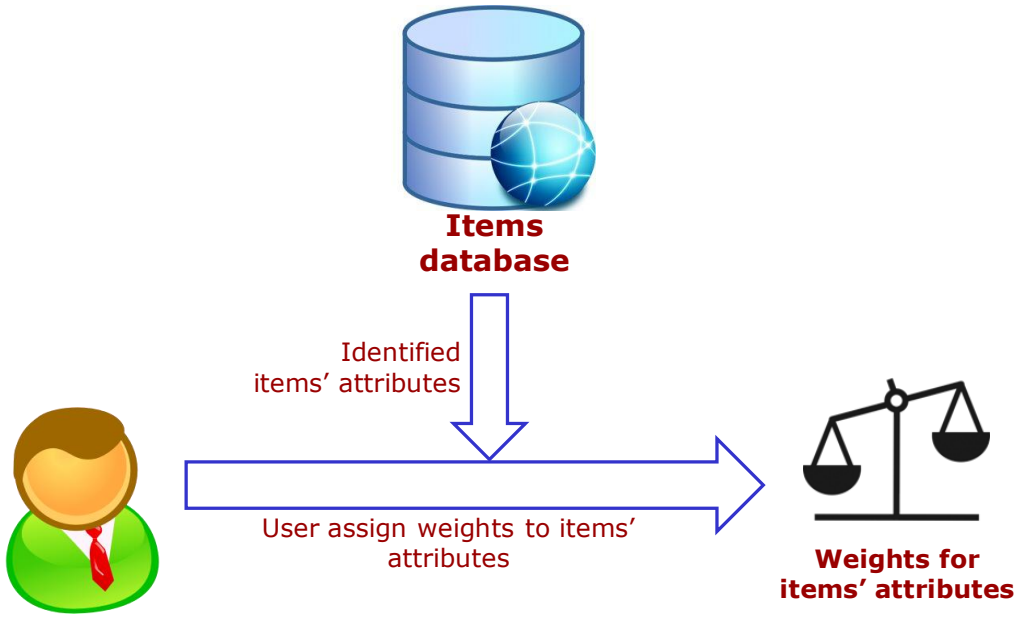


Figure 5.3: Technique to obtain the user weights for all items' attributes

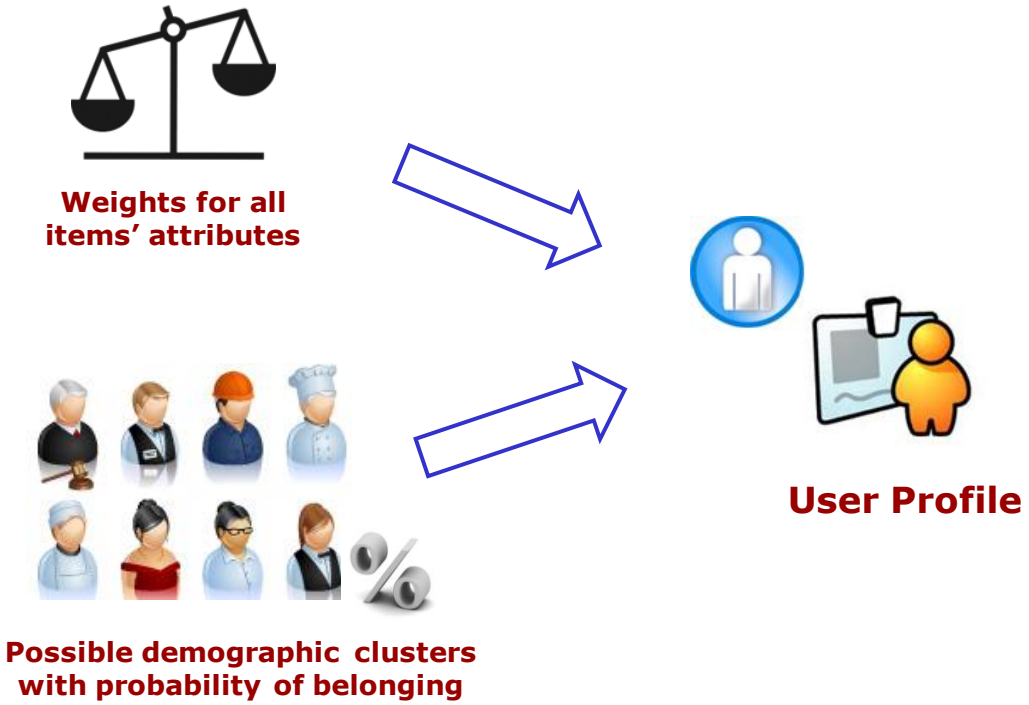


Figure 5.4: Dimensions of the proposed user profile

Algorithm to Create the User Profile

Requirements:

- User (u).
- Items database.
- Demographic database.

Output:

- The user profile, formed by the vector of possible demographic clusters with their probability of belonging, and the vector of user given weights for all items' attributes.

Algorithm:

1. Obtain the possible demographic clusters and their probability of belonging.
   - 1.1. Find the demographic clusters where the user might belong.
     - 1.1.1. By means of Feature Selection techniques, select the m demographic variables that best differentiate each cluster from the others.
     - 1.1.2. Ask the user for his/her information for the m selected demographic variables.
     - 1.1.3. Depending on the information in the database, use equations (5.3) or (5.5) to create the vector of possible demographic clusters *PossibleClusters(u)* that user u might belong to.
   - 1.2. Assess the belonging probability of each of the selected possible demographic clusters.
     - 1.2.1. Assess the belonging probability of each of the selected possible demographic clusters.
     - 1.2.2. Add a new attribute to keep track of which demographic cluster each instance in the Cartesian product was obtained.
     - 1.2.3. Apply the k-means clustering algorithm to learn the newly added attribute (demographic cluster), and create the same number of groups as selected clusters.
     - 1.2.4. Ask the user to select the groups that best define him/her, based on the information contained in each of them.
     - 1.2.5. Using equation (5.6), obtain the probability of belonging to each of the selected demographic clusters.
   - 1.3. Create the vector *DemogClusters(u)* for user u, as presented in equation (5.2).
2. Obtain the user weights for all items' attributes.
   - 2.1. Identify the item attributes that define all the possible items.
   - 2.2. Ask the user to assign weights to each of these attributes.
   - 2.3. Create the vector *AttributeWeights(u)* for user u, as presented in equation (5.8).
3. Create the User Profile of user u, by combining the *DemogClusters(u)* and *AttributeWeights(u)* vectors, as presented in equation (5.1).

Algorithm 5.2: Create the user profile

5.1.2 Location Aware

The second phase of the methodology consists of obtaining the user's mobility level and current location. This information will be used later to assess items' utility for the user, based on the distance and time from the user to each of the possible items to be recommended. The following two stage technique is proposed to obtain the required information. Initially the user is asked to choose the means of transport (i.e. mobility level) that best describes his/her current situation, from either driving a car, using the bus or walking. Based on the user selection, the routes to be considered when predicting the distance and time to each item, would be selected according to it. Secondly, obtain the user's current latitude and longitude spatial coordinates. These can be acquired either by the device the user is using to interact with the RS, if it has GPS capabilities, or by asking the user for his/her current address. We developed a Java software module to obtain a user's spatial coordinates from his/her current address. This software module connects to Google Maps®, through its application programming interface (API), sending the user address and receiving his/her latitude and longitude coordinates in response.

Finally, to understand and consider the user's surroundings, we propose using a spatial database to locate the user and obtain the spatial distributions and constraints of his/her current location. The spatial database may contain information about streets, street directions, average street times, rivers, buildings, points of interests, intersections, and so forth. The exact user position relative to the identified elements in the spatial database is obtained by the intersection of their coordinates. Figure 5.5 shows an example of the goal of this phase of the methodology, and Algorithm 5.3 presents the methodology to obtain the User Location and Mobility Level.

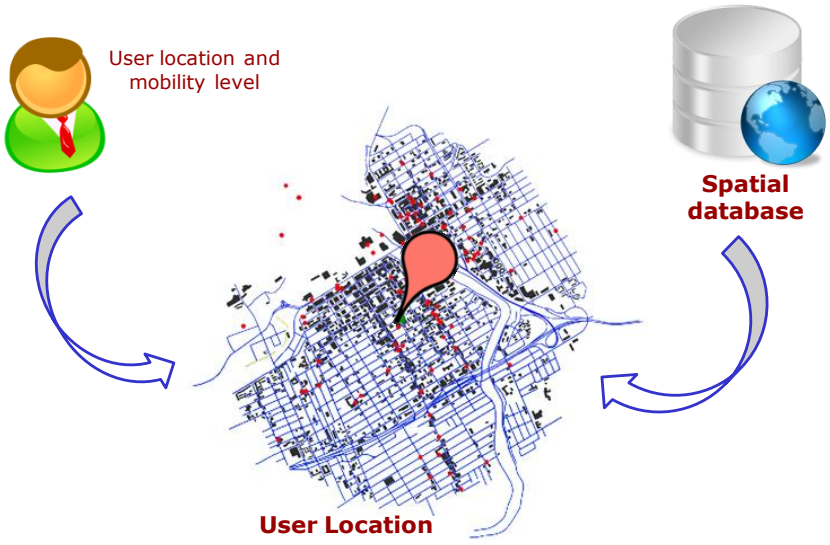


Figure 5.5: User location (Location Aware)

Algorithm to Obtain the User Location and Mobility Level

Requirements:

- User (u).
- Geo-Spatial database.

Output:

- User location by means of his/her latitude and longitude spatial coordinates.
- User current mobility level (driving a car, using the bus, walking).

Algorithm:

1. Ask the user to choose the means of transport that best describes his/her current situation; driving a car, using the bus or walking.
2. Obtain the user current location.
   2.1. Ask the user his/her current address.
   2.2. Using a developed software module that connects to Google Maps®, obtain the user spatial latitude and longitude coordinates from his/her current address.

Algorithm 5.3: Obtain the user location and mobility level

5.1.3 Assess the Items' Utilities Considering the User Profile

This phase of the methodology deals with assessing the utility of each item, based on the previously created user profile. Recall that the proposed user profile is formed of two dimensions: the possible demographic clusters the user could belong to, and the user weights for all items' attributes, as expressed by equation (5.1) and depicted in Figure 5.4. Since both dimensions are considered when obtaining the items' utilities we will obtain two utilities per item, one based on the possible demographic clusters and the other on the weights of its attributes. The two utilities per item are calculated during two different stages in this phase of the methodology. The following Sections describe the proposed techniques to obtain both of these item utilities in detail.

Obtaining the Items' Utilities Based on the Possible User Demographic Clusters

The first stage to obtain the items' utilities based on the user profile is achieved by considering the possible demographic clusters where the user could belong to. Thus, these items' utilities are calculated using the previously created vector of possible demographic clusters, and the probability of belonging to each of them (as presented in equation 5.2). From this point on, we will refer to this estimated item utility as *Demographic Utility* (*DU*), in order to differentiate one item utility from

another. However, as explained in Section 3.3.1, when using the demographic generalization method we may encounter one of two scenarios: In the first, the items to be recommended are included as information of each identified demographic cluster in the database. In this case, the utility for each item is obtained by the sum of the user belonging probabilities of all demographic clusters that include the item, as expressed by the following equation:

$$DU\left(I_x, u\right) = \sum_{i=1}^{k} \begin{cases} \text{If } I_x \in C_i : P(u, C_i) \\ \text{Otherwise} : 0 \end{cases} \qquad (5.9)$$

Here, $DU(I_x, u)$ represents the *Demographic Utility* of item *x* for user *u*, *k* represents the number of elements in the previously obtained vector *DemogClusters(u)* of possible demographic clusters the user *u* could belong to, and $P(u, C_i)$ represents the probability that user *u* belongs to the demographic cluster *i*.

The second scenario we may encounter when using the demographic generalization method is that the items to be recommended are not already linked to each demographic cluster. In these cases, we can apply one of two techniques. In the first, manually map each possible item to be recommended to all the demographic clusters it could belong to; this can be done considering the information included in the demographic database. The item's utilities can then be obtained using equation (5.9). However, this technique is impractical due the large number of both possible items to be recommended and demographic clusters. In addition, this approach also requires excessive maintenance, since every time a new item is added to the database we would need to manually link it to every possible demographic cluster it could belong to. Therefore, we suggest using the second technique to deal with scenarios where the items are not already linked to the demographic clusters.

In the second technique, arrange all the possible items to be recommended under several non-changing items' categories, if they are not already organized this way. An item may belong to one or more categories. Subsequently map each item category to one or more demographic clusters, based on the information contained in the demographic database. With this approach, we need to manually map only the items' categories, of which there are significantly fewer than the number of items, to each demographic cluster. In addition, we would only need to do this once, since every time an item is added to the database it would be enough to assign it to one or more of the previously identified categories. Figure 5.6 illustrates the required mapping between items and demographic clusters.

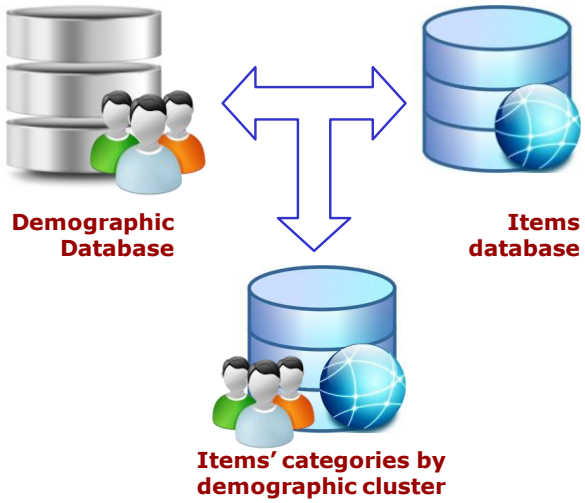


Figure 5.6: Mapping demographic clusters to items' categories

However, since the demographic clusters are now linked to the items' categories, we first need to obtain the probability that a user might like each of these categories. As shown in equation (5.10), these probabilities are calculated based on the probabilities of belonging to each demographic cluster the user could belong to.

$$P\left(u, ItemCat_x\right) = \sum_{i=1}^{k} \begin{cases} \text{If } ItemCat_x \in C_i : P(u, C_i) \\ \text{Otherwise} : 0 \end{cases} \tag{5.10}$$

Here, $P(u, ItemCat_x)$ represents the probability that user *u* likes item category *x*. Subsequently, these obtained probabilities are used to obtain the *Demographic Utility* for each item. The *Demographic Utility* for each item is represented by the average of the probabilities that the user likes each category that the item belongs to. Hence, the *Demographic Utilities* are obtained by the following equation, where *q* represents the number of identified items' categories.

$$DU\left(I_x, u\right) = \frac{1}{Num of\ Cat of\ I_x} \sum_{i=1}^{q} \begin{cases} \text{If } I_x \in ItemCat_i : P(u, ItemCat_i) \\ \text{Otherwise} : 0 \end{cases} \tag{5.11}$$

Obtaining the Items' Utilities Based on the User Weights for the Items' Attributes

The second stage deals with obtaining the items' utilities considering now the user given weights of each of the previously identified items' attributes. Therefore, these items' utilities are calculated

using the previously created vector of user given weights to each item attribute (as presented in equation (5.8)). Hereafter, we will refer to this item utility as *Weighted Utility* (*WU*), in order to differentiate one item utility from another. The weighted utility for each item is calculated as the sum of each user given weight for all the item's attributes or characteristics. The *Weighted Utilities* are obtained by applying the following equation:

$$WU(I_x,u)=\sum_{i=1}^{r}\begin{cases}\text{If } A_i \in I_x : W(u,A_i)\\ \text{Otherwise} : 0\end{cases} \qquad (5.12)$$

Here, $WU(I_x,u)$ represents the *Weighted Utility* of item *x* for user *u*, *r* represents the number of identified items' attributes in the previously obtained vector *AttributeWeights(u)*, and $W(u,A_i)$ represents the weight that user *u* gave to item attribute *i*.

After this third phase of the methodology is completed, both the *Demographic Utility* and the *Weighted Utility* for each item have been estimated. Figure 5.7 depicts the proposed methodology to obtain these two item utilities from the previously created user profile, and Algorithm 5.4 presents the methodology to obtain the demographic and weighted utilities for all the items.

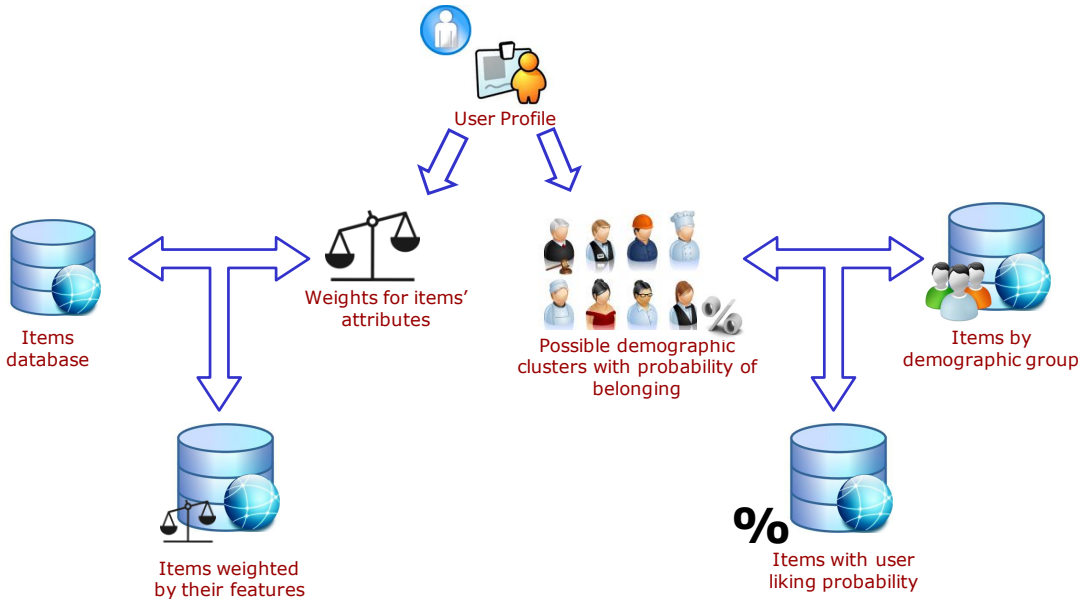


Figure 5.7: Proposed technique to assess items' utilities from the user profile

Algorithm to Obtain the Demographic and Weighted Utilities for all the Items

Requirements:

- User Profile of user *u* (*DemogClusters(u)* and *AttributeWeights(u)* vectors).
- Items database.

Output:

- The demographic and weighted utilities for all items in the database.

Algorithm:

1. Obtain the demographic utilities for all the items in the database.
   1.1. If the items to be recommended are included in each demographic cluster in the database:
      1.1.1. Calculate the demographic utility for each item for user *u*, using equation (5.9).
   1.2. Otherwise:
      1.2.1. Organize all the items under one or more non-changing items' category.
      1.2.2. Map each item category to one or more demographic clusters in the database.
      1.2.3. Using equation (5.10), calculate the probability that the user *u* would like each item category.
      1.2.4. Calculate the demographic utility for each item for user *u*, using equation (5.11).
2. Calculate the weighted utility for each item for user *u*, using equation (5.12).

Algorithm 5.4: Obtain the demographic and weighted utilities for all the items

5.1.4 Assess the Items' Utilities Considering the User Location

The fourth phase in the methodology involves assessing the utility for each item, based on the user's current location and mobility level. In this phase, each item is assessed under two additional criteria: the distance from the user, and the time it would take the user to reach it, considering the user's mobility level. Both the distance and time are considered two new user utilities for each item. However, it is important to note that these new utilities, expressed in kilometers and minutes from the user, are inversely proportional to the item utility for the user; that is, the closer an item is the higher is its utility for the user. From this point, we will refer to the item and distance utilities as *Time Utility* (*TU*) and *Distance Utility* (*DisU*) respectively, to differentiate them from one another. Thus, $TU(I_x, u)$ and $DisU(I_x, u)$ represent the *Time Utility* and *Distance Utility* of item *x* for user *u*.

To obtain the time from the user to each item, we developed a Java software module that connects to Google Maps® through its application programming interface (API). The module receives the user's current coordinates (i.e. latitude and longitude) and his/her mobility level (driving, walking or using the bus), and returns the time and the best route (including directions) from the user to each possible item in the database.

The distance from the user to each item is obtained by means of SQL spatial queries, performed over the spatial database using the current user coordinates. For more information on SQL spatial queries and spatial databases, we suggest [81]. Figure 5.8 and Algorithm 5.5 present the techniques to obtain the time and distance utilities for each item.

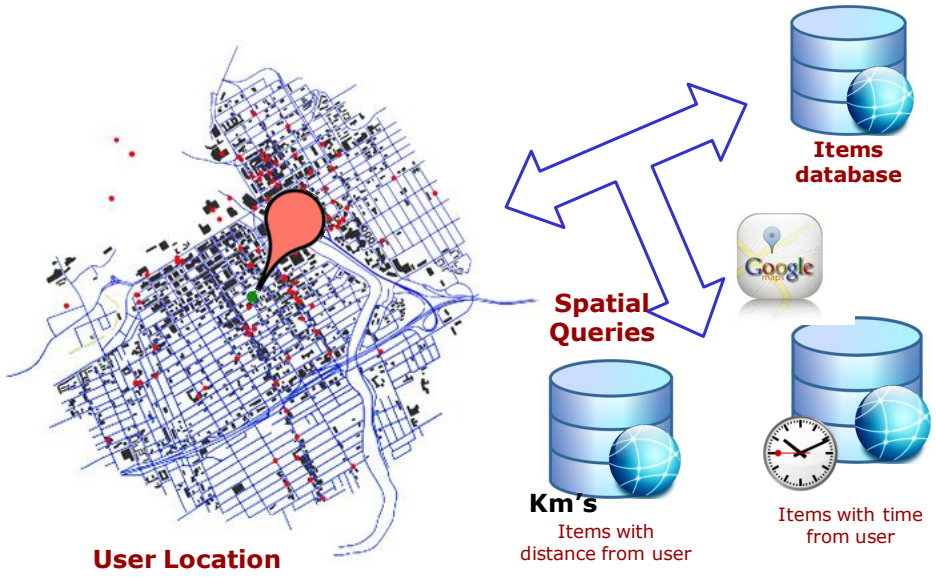


Figure 5.8: Methodology to obtain the time and distance utilities for each item

---

Algorithm to Obtain the Distance and Time utilities for all the items

---

Requirements:

- User Location (latitude and longitude spatial coordinates).
- User Mobility Level (driving a car, using the bus, or walking).
- Items database.

Output:

- The distance and time utilities for all the items in the database.

Algorithm:

1. Using a developed software module that interfaces with Google Maps®, send the user coordinates and mobility level, and obtain the time and best route from the user to each item in the database.
2. By means of SQL spatial queries, obtain the distance from the user to each item in the database.

Algorithm 5.5: Obtain the distance and time utilities for all the items

5.1.5 Create the User Preference Model

The item utilities that were obtained during the third and fourth phases represent the four criteria considered to create the user preference model. These criteria are: who the user is (demographic utility), special user preferences (weighted utility), distance from the user to the item (distance utility), and time from the user to the item (time utility). After these utilities have been calculated, the next phase is to obtain the user preference model, which represents how much each of these criteria is important to the user.

To obtain the user preference model we propose applying the UTA* algorithm, as presented in Section 3.1.1. However, in order to use this algorithm we first need the user's weak preference order, which represents the user preferences for some items, after considering the four criterions for each of them. Consequently, the system randomly selects ten items, along with their four utilities, to present to the user, who is then asked to rate them in non-descending order. The result of the UTA* algorithm is a vector of four values, which represent the weights that the user gives to each of the four considered criterions. These weights are calculated by means of linear programming techniques that aim to be as consistent as possible with the user provided weak preference order. The resulting vector from the UTA* algorithm has the following form:

$$PM(u) = \{W_{DU}, W_{WU}, W_{DisU}, W_{TU}\} \qquad (5.13)$$

Here, $PM(u)$ stands for *Preference Model* of user u, and W_{DU}, W_{WU}, W_{DisU} and W_{TU} represent the calculated weights for the user for the *Demographic Utility*, *Weighted Utility*, *Distance Utility* and *Time Utility* respectively. It is important to note that since these weights are expressed as percentages, their sum must equal *100%*. Figure 5.9 and Algorithm 5.6 present the technique to create the user preference model.

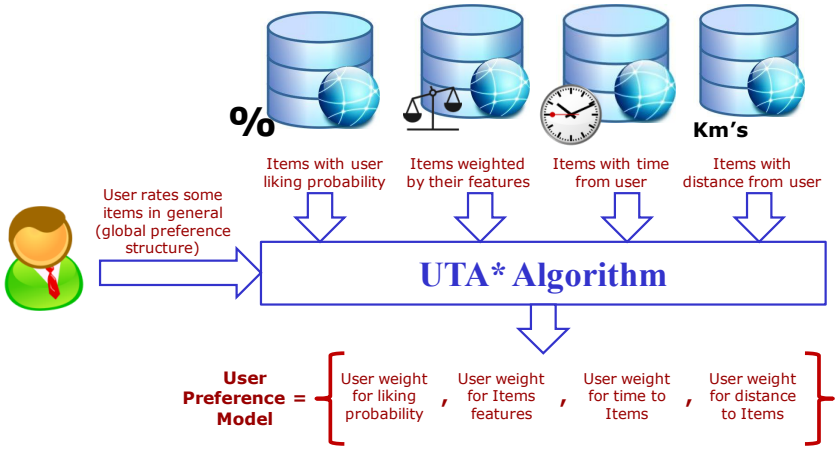


Figure 5.9: Technique to create the user preference model

Algorithm to Create the User Preference Model

Requirements:

- User *u*.
- Items database.
- Demographic, weighted, distance, and time utilities (criteria) for all the items in the database.

Output:

- The user preference model for user *u* (i.e. a vector of four user weights, one for each considered criterion).

Algorithm:

1. Randomly select ten items from the database, along with the demographic, weighted, distance, and time utilities for each of them.
2. Ask the user to rate these items in non-descending order, considering the information of the four criteria for each.
3. Obtain the user preference mode by applying the UTA* algorithm over the set of ten user rated items with their utilities.

Algorithm 5.6: Create the user preference model

5.1.6 Recommend the Items that Best Match the User Preferences

The last phase of the methodology is to assess an integrated final utility for each item, using the previously obtained user preference model. However, in order to apply the previously obtained weights from the model, the values of the four different utilities must be normalized so they are in the same range *{0,…,1}*, otherwise the final utility would be biased toward the criterion with the highest scale. Therefore, each utility is divided by its highest utility. Moreover, since the distance and time utilities are inversely proportional to the user utility, after dividing their values by their highest utility the resulting value is subtracted from *1*, and the remainder is the utility to be considered. The final integrated item's utilities are obtained by applying the following equation:

$$U(I_x,u) = W_{DU}\frac{DU(I_x,u)}{\max(DU(u))} + W_{WU}\frac{WU(I_x,u)}{max(WU(u))} + W_{DisU}\left(1-\frac{DisU(I_x,u)}{\max(DisU(u))}\right) + W_{TU}\left(1-\frac{TU(I_x,u)}{\max(TU(u))}\right) \quad (5.14)$$

Here, $\max(\mathrm{DU(u)})$, $\max(\mathrm{WU(u)})$, $\max(\mathrm{DisU(u)})$, and $\max(\mathrm{TU(u)})$ represent the maximum demographic, weighted, distance, and time utilities an item could obtain, respectively. After the utilities for all items have been assessed, the system recommends the items with the highest utility to

the user by presenting a list of all the items, ordered by their utility. Figure 5.10 depicts this final phase to recommend the items to the user. Algorithm 5.7 presents the methodology to recommend the items that best match the user preferences.

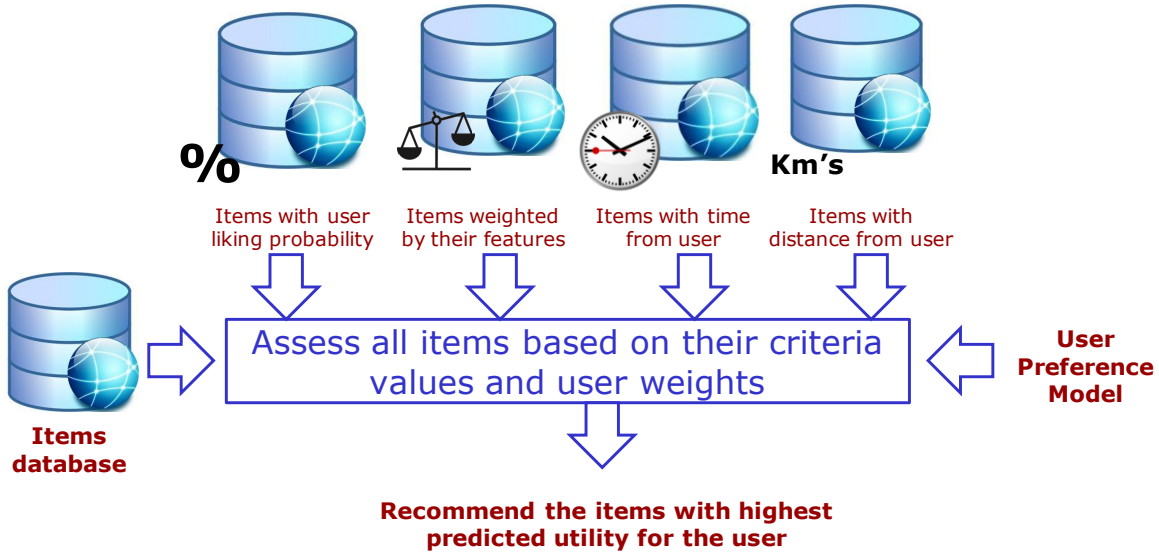


Figure 5.10: Recommending the items with highest utility to the user

Algorithm to Recommend the Items that best Match the User Preferences

Requirements:

- User Preference Model for user *u* (vector of user weights for each item criterion).
- Items database.
- Demographic, weighted, distance, and time utilities (criteria) for all the items in the database.

Output:

- Ordered ranked list of recommended items for user *u*.

Algorithm:

1. Using equation (5.14), obtain an integrated utility for each item in the database.
2. List all the items in the database in non-ascending order, based on their assessed integrated utility.

Algorithm 5.7: Recommend the items that best match the user preferences

5.2 Summary

This Chapter presented our proposed methodology to create a **Personalized Multi-Criteria Context-Aware Hybrid Recommender System**. Our methodology aims to combine the advantages of previous implementations of RSs found in the literature (as presented in Chapter 4). The RS resulting from applying our methodology has the following noteworthy characteristics: it considers the user's current location, it creates and bases its recommendations on a multi-criteria user preference model, it combines content-based and collaborative-based recommending techniques, and it generalizes the user profile from a large-scale demographic database.

We began the Chapter by describing our proposed methodology, presenting a high level model followed by a detailed description of each phase. The first phase consists of creating the user profile. We propose a user profile that combines the information contained in the demographic clusters the user could belong to, and the user's special preferences. We presented our proposed techniques, equations and algorithms to create this combined user profile. The next phase deals with obtaining the user's current location and mobility level. A demographic and a weighted utility are introduced and defined in the third phase of the methodology. These are proposed to measure the utility of each item to the user, based on the identified demographic clusters the user belongs to, and his/her given preferences, respectively. In the fourth phase of our technique we define two other item utilities, distance and time. These new utilities are introduced to represent how far each item is from the user, and how much time it would take the user to reach it, correspondingly. The fifth phase deals with creating the user preference model, considering the four previously obtained items' utilities as criteria. The final phase assesses an integrated utility for each item, considering the previously obtained utilities and the user preference model. The recommended items are those with the highest integrated utility.

The following Chapter presents our case study, an implementation of our methodology (as presented in this Chapter) in PeRS, a Personalized Recommender System that recommends events to attend to the users. We also introduce the design of an experiment performed over the case study to evaluate our methodology, and discuss our results.

Chapter 6
Experimental Evaluation

This Chapter introduces our case study, details our experimental design and discusses our experimental results.

6.1 PeRS Case Study

As a case study, and in order to evaluate our methodology for RSs, we created PeRS, a Personal Recommender System that recommends events to attend to the users. Within this Section we describe the hardware and software environment set up, the databases used, and the phases performed by the system to recommend an item.

The integrated system and the user interface were developed in Java® v.7.5 language to ensure its compatibility with different operating systems, and using the open source NetBeans® v.6.9.1 Integrated Development Environment (IDE). Microsoft SQL Server 2008 was used to create and manage the system databases. This database server was selected due its capabilities to manage spatial databases, and its straightforward connectivity to other used third party software systems. Additionally, WEKA, a collection of machine learning algorithms for data mining tasks, written in Java and developed at the University of Waikato [82], was used to apply the K-Means clustering algorithm as required. Finally, Google Maps® through its Application Programming Interface (API), was used to obtain the user position, and his/her times and directions to all the events. Next, we introduce the databases we used.

6.1.1 Databases

In order to implement our proposed methodology to create a RS that recommend events to the users, the following source datasets were used.

Ottawa Open Data, Events Database. This is our items dataset, which contains the events that will take place within the Ottawa/Gatineau region. This dataset is freely provided by the City of Ottawa, through the Ottawa Open Data foundation. The Ottawa Open Data foundation was created as an effort to improve citizen engagement, and enhancing transparency and accountability to its residents [83]. The events dataset is offered through the following five XML documents, events, categories, options, locations and city sectors. By events we refer to the following range of activities that will take place within the Ottawa/Gatineau region, festivals/fairs, film/new media, galleries, heritage and traditions, literary, museums, music, performance/theater, seasonal celebrations, sports/outdoors, and tours.

The Events XML file contains all events with their most relevant information. This information includes the start and end dates of an event, the English and French title, description and summary of the event, its schedule times, the locations where it is going to be presented, the categories under which it has been classified (one or more), and other event features such as the language of presentation, type of parking, cost, and so forth. Additionally, this dataset also provides information for each event, about the organizing company, the name of the person to contact for inquiries, contact phones and e-mails, and website URLs for further information.

Furthermore, the categories XML file includes the 14 identified categories under which all the events are classified. Some of these categories are dance, festival/fair, film/new media, museum, music, etc. For each one of these categories the dataset includes a brief description in English and French.

Additionally, the options XML file contains a hierarchical structure of possible options an event may have. Some of these options include whether the event will take place indoor or outdoor. It is also noted whether the event is going to be presented in English, French or both languages and if the location where it will be held has parking space and if it is free or paid. Other details include how accessible the event is for people with special needs, such as ramps or elevators for wheelchairs, closed captions for people with hearing problems, or any aids for visually impaired people. We also record whether the event is family friendly, or if it has an age restriction, etc.

Moreover, the locations XML file consists of all the venues and places where the events are going to be held. For each one of these venues, the dataset includes their name, full address, city, province, postal code, and street intersections.

The City Sectors XML file includes a hierarchical structure of city sectors where the events may take place. For instance if the event will take place within the Ottawa urban area, if it is going to be in downtown, south Ottawa, east Ottawa, or west Ottawa. Similarly, the Gatineau urban area has been divided between Aylmer, Hull, Gatineau, Masson-Anger, and Buckingham sectors.

Since all the information related to the events is contained in XML files, the XMLSpy® software system [84], was used to review the information and parse it to our SQL Server databases. Once the information was exported to SQL databases, a manual cleaning process was required, since it contained large number of errors, missing data, duplicated values, and non-consistent/structured information.

Finally, it is noteworthy to mention that the information about events is captured and managed by the City of Ottawa – Parks, Recreation and Cultural Services, and Cultural and Heritage Services Branch, in collaboration with the National Capital Commission and the City of Gatineau [83].

PRIZM C2 Database. This is our large-scale demographic database that the system will use to generalize the user profiles. This database classifies Canada's neighborhoods into 66 unique lifestyle types, providing insights into the behavior and mindset of the people. PRIZM C2 integrates information from the 2006 Canada Census, Social Values from Environics Research and EA's current year Demographic Estimates and Projections [85]. Table 6.1 shows an example of the lifestyles (i.e. demographic groups) included in the database.

The PRIZM C2 database contains both, demographic and non-demographic information for each one of the identified demographic groups in it. For each demographic cluster, the database includes the percentage of people from it, for whom each value of the contained demographic information holds. Some of the contained demographic information included in the database are the percentage of people within each group, divided by their age range {0-4, 5-14, 15-24, 25-44, 45-64, 65-74, 75-84, 85+}. Other information includes the mother tongue , considering English, French, and non-official languages, the marital status of the population, divided as single, married, widow/divorce/separated, the type of transport they most often use (car, public transit, etc.)

Additionally, the database also presents some non-demographic information for each cluster, including data about the social values, life style, preferences, and customs of the people in each group. Among this other type of information, we find data about leisure activities that the people within each cluster enjoy to do, the items that they are most likely to buy, their financial situation, the type of media they usually look at, the food and drink they usually consume, the type of automotive most used, and general attitudes or information about their personality. Figure 6.1, extracted from the database, depicts the type of information presented for each demographic group identified in the

dataset. Some values have been removed and changed from the figure to preserve the copyrights of the database.

Who They Are

Population	Cluster %	Index Canada
Age		
0-4		81
5-14	10.59	
15-24	15.35	
25-44		85
45-64	31.04	
65-74	8.53	
75-84		101
85+		
Mother Tongue		
English	70.97	
French		
Non-Official		107
Immigration		
Immigrant	27.31	
Arrived < 1961		
1961-1970	18.44	161
1971-1980		133
1981-1990		
1991-1995		
1996-2000	9.89	
2001-2006		56
Visible Minority		
Yes	18.24	

Adult Population	Cluster %	Index Canada
Marital Status		
Single		83
Married	62.93	
Wid/Div/Sep		63
Mode of Transport		
Car	79.58	
Public Transit		115
Class of Worker		
Employed	76.65	
Self-Employed		189
Unpaid	0.34	
Occupation		
Primary		
Blue Collar		57
Service Sector	32.20	
White Collar	49.11	
Education		
No cert/dipl/deg	8.96	
High school cert		73
Trade		34
College	13.35	
Some university		124
University degree	48.98	

Households	Cluster %	Index Canada
Maintainer Age		
<25		36
25-34		
35-44	15.81	
45-54	25.33	
55-64		130
65-74	14.03	
75+		105
Size		
1 Person	19.11	69
2 People		
3 People		
4+ People	33.19	145
Family Status		
Non-Family		70
Couples w/ kids	55.15	
Couples, no kids	36.70	
Lone parent		51
Age of Children		
<6	15.40	
6-14		97
15-17		
18-24		125
25+	10.67	

Dwellings	Cluster %	Index Canada
Tenure		
Owned	86.42	
Rented		44
Band Housing		1
Period of Construction		
<1946	17.77	
1946-1960		
1961-1970		72
1971-1980		55
1981-1990		
1991-1995	5.52	
1996-2000	7.13	
2001-2006		73
>2006		
Type		
Single		133
Semi		
Row		
Duplex	3.04	
Low-rise	7.11	
High-rise		84
Mobile	0.12	
Dwelling Value Index		275

How They Live

Leisure
theatres

baseball

travel to France

Shopping
personal video recorders
iPads

Best Buy

Media
sports newspaper sections

news/talk radio
online radio

Food/Drink
organic meat
fresh vegetables

olive oil

Financial
$500,000+ in securities and savings

online stock trading

donations to cultural groups

Automotive
sedans
compact premium cars
$30,000-$40,000 on latest vehicle
BMW

Attitudes

"I'm willing to pay more for enviro-friendly products"
"When shopping for clothes, I generally look for designer labels"

"I am careful of what I eat in order to keep my weight under control"

Figure 6.1: Demographic and life style information contained in PRIZM C2 [85]

Furthermore, the PRIZM C2 database also includes information about the distribution around Canada, per province, of the people belonging to each demographic group. Figure 6.2 depicts how this information is presented for each demographic cluster.

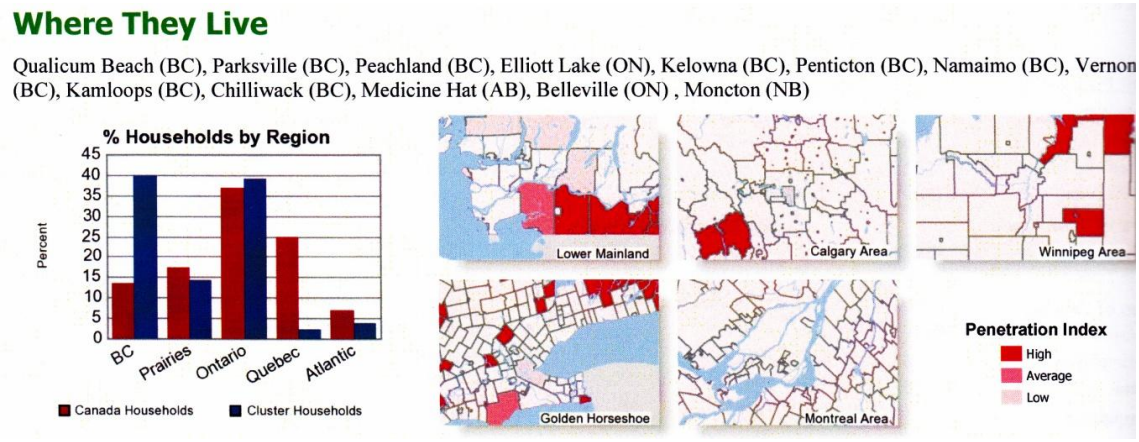


Figure 6.2: Demographic group people distribution from PRIZM C2 [85]

Ottawa Open Data, Geospatial Databases. These databases contain spatial information to geographically locate different elements of interest. Geospatial databases are also provided by the Government of Canada through the Ottawa Open Data foundation [83]. The selected spatial databases to be used in our case study are rivers, buildings, museums, municipalities, wards, roads, cities and country areas. These databases include, for each one of its elements contained in it, name, and geo-spatial figure, by means of geographic latitude and longitude coordinates. Geospatial databases are provided as shape documents (i.e. files with .SHP extension), which is the native format from the Environmental System Research Institute (ESRI) for Geographic Information Systems (GIS). In order to visualize these shape files, and store their spatial information in SQL Server databases, SqlSpatial® and Shape2Sql® software systems were used. For more information on these open source software systems we recommend [81]. It is important to mention that the venues where the events are going to be held, where manually mapped to the elements contained in the geo-spatial buildings and museums databases. This mapping was necessarily to obtain, by means of spatial SQL queries, the spatial distance from the user to each one of the possible events to be recommended. The other spatial databases (rivers, municipalities, wards, roads, cities and country areas) where only used by the system to locate and present to the user the elements surrounding him/her.

6.1.2 Application of our Algorithm to the Case Study

Within this Section we present the phases performed by the PeRS to recommend events to a user, in accordance to Algorithm (5.1) presented in Chapter 5. As a running experiment, while we present the phases followed by our methodology, we also provide in parallel the data introduced by a real user and show the results from the actual execution of the system.

Obtaining the possible demographic clusters where the user may belong

Following Algorithms (5.1) and (5.2), the first phase consists of selecting from our demographic database, the demographic variables that are more representative to classify a user within one or more demographic clusters. Table 6.1 presents some of the demographics groups contained in PRIZM C2 database.

Name	Description
Pets & PCs	Large, upscale suburban families
Mr. & Mrs. Manager	Upscale, dual-income exurban households
Newcomers Rising	Young, downscale city immigrants
Suburban Rows	Younger, thriving immigrant families
Winner´s Circle	Well-off, middle-aged exurban families
South Asian Society	Younger, upper-middle-class South Asian families
…	…

Table 6.1: Demographic groups contained in PRIZM C2 [85]

For our case study we selected, as the most representative demographic variables to classify a user, the information contained in the database that could lead one user to select a specific type of event over the others. For instance, information about whether the user is single or married, his/her mother tongue, his/her mode of transport, or his/her education level, are factors that may influence a person when deciding the event to attend. The data selected by inspection to this end, based on domain knowledge, are the following 9 attributes: population age, mother tongue, marital status, mode of transport, occupation, education, family status, age of children, and dwellings tenure. Figure 6.3 shows the demographic questionnaire asked to the user, along with his/her real answers. From now on we will refer to the user of our running example as user “A”.

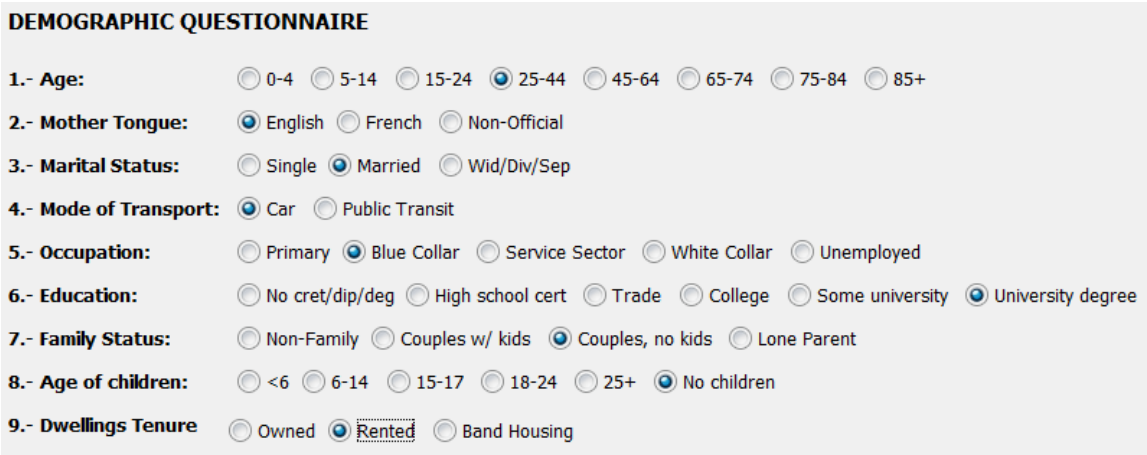


Figure 6.3: Demographic questionnaire applied with real answers from user "A"

After the user has answered this demographic questionnaire, the following phase is to determine the possible demographic groups where he/she may belong. This process is done using the probabilities contained in PRIZM C2 database for a user to belong to each cluster, based on the user given values for each demographic variable. However, these probabilities of belonging based on the answer for each selected demographic data are independent one from another. Therefore, the integrated probability for a user to belong into a cluster is given by the multiplication of those nine corresponding probabilities for each cluster. Consequently, the probability that this specific user "A" from our running example belongs to each cluster, is given by the multiplication of the probabilities of belonging to each group given his/her information, as presented in Figure 6.3. In this example, that he/she is between 25-44 years old, English speaking, married, uses a car, works in a blue collar activity, has an university degree, doesn't have kids, and rents the place where he/she lives. This probability is calculated for the 66 demographic clusters identified in the demographic database. Finally, as presented in equation (5.5), the clusters with a resulting probability higher than a selected threshold are chosen as the possible demographic clusters where user "A" may belong. For our case study, we decided by inspection to set this threshold as the half between the lowest and the highest probability from the sixty six calculated probabilities, one for each demographic cluster, for each user. The following equation presents how this threshold is obtained:

$$Threshold = \frac{\max(Prob) - \min(Prob)}{2} + \min(Prob) \qquad (6.1)$$

The threshold was calculated in this way to filter out the demographic groups with smaller or equal probability of belonging than the half of the highest probability obtained for a demographic group. If

this threshold is set higher, we would obtain less demographic groups per user, reducing the number of non-demographic information presented to the user in the following phase of the process. Recall from our methodology (as presented in Chapter 5) that this threshold represents only the first filter for the demographic clusters. The final percentages of belonging to each cluster will be calculated based on the groups, from the ones obtained by the clustering algorithm, where the user identifies the most, considering the non-demographic information in each of them. However, experimenting how the final recommendations would be affected by modifying this threshold is considered out of the scope of this thesis, and we propose it as a possible future work.

The resulting clusters with a probability of belonging higher than the obtained threshold, for user "A" in our running experiment, are shown in Table 6.2.

ID	Name	Description
DG-1	Grads & Pads	Young, lower-middle-class urban singles
DG-2	Electric Avenues	Young, middle-class urban singles and couples
DG-3	Grey Pride	Lower-middle-class, suburban apartment-dwelling seniors
DG-4	Startups & Seniors	Midscale mix of young and mature singles and couples
DG-5	Daytrippers & Nightowls	Young, mobile urban singles and couples
DG-6	Single City Renters	Young, apartment-dwelling urban singles and couples
DG-7	Young Digerati	Younger, upscale urban trendsetters
DG-8	Newcomers Rising	Young, downscale city immigrants
DG-9	Park Bench Seniors	Low-income seniors in urban high-rises
DG-10	White Picket Fences	Young, middle-income exurban families

Table 6.2: Possible demographic clusters [85] where user "A" may belong

Obtaining the probability of belonging to each demographic cluster

The following phase, as presented in Algorithms (5.1) and (5.2), consists of obtaining the probability of belonging to a cluster, based on the non-demographic information contained in the database. To calculate this probability, we first obtain the union of the Cartesian products of the non-demographic information for each one of the identified clusters. This is the union of the Cartesian products of the included leisure and shopping activities, media and food/drink consumed, and automotive preferences for each resulting cluster from the previous phase. An additional attribute is added to keep track for each instance of the Cartesian product, from which demographic cluster it was obtained. An example of this resulting dataset for our running example is shown in Table 6.3.

ID	Leisure	Shopping	Food/Drink	Media	Automotive
DG-1	Movies	Audio equipment	Scotch whiskey	Alternative rock	Honda
DG-1	Hiking	Computer software	Sports drinks	Famous	Subaru
DG-1	Nightclubs	Zara	Tortilla chips	National Post	Sport coupes
DG-1	Pilates	Health food stores	Premium ice cream	W Five	Own 1 vehicle
…	…	…	…	…	…
DG-2	Museums	Club Monaco	Tea	The Globe and Mail	Compact cars
DG-2	Music festivals	Comic books	Corn chips	Report on Business Magazine	Sport coupes
DG-2	Yoga	Digital cameras	Fish and seafood	Read blogs regularly	Honda
DG-2	Art galleries	RW & Co.	Organic vegetables	Online job search	Subaru
…	…	…	…	…	…

Table 6.3: Cartesian product of non-demographic information from the identified groups [85]

The following stage is to apply the k-means clustering algorithm over this obtained dataset. The k-means algorithm is configured, according to our proposed methodology, to create the same number of groups as identified demographic clusters (ten in our running example, as presented in Table 6.2), and to learn the demographic cluster ID. The result from applying this clustering algorithm are ten newly created groups, labeled by the most representative information contained in each one of them. Recall from our methodology, that the idea of applying a clustering algorithm to create the same number of groups as previously identified clusters, considering only the non-demographic information, is to find out the distribution of this data between the selected demographic groups. Consequently, when the user selects the groups with the activities where he/she identifies the most, we can calculate the probability of belonging to each demographic cluster, based on the distribution of these activities. After applying the clustering algorithm, the user is asked to select from these groups, those where he/she identifies the most. The resulting clusters for user "A" are presented in Table 6.4. This table also presents the selected clusters by the user in our running experiment.

Attribute	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
Leisure	Movies	Art Galleries	Symphonies	Pilates	Spent $4000+ on last vacation	Bus travel within Canada	Dancing	Museums	Community theatres	Museums
Shopping	RW $ Co.	Research products online	Gourmet food stores	Jacob	Research products online	Handheld organizers	Computer software	Fairweather	Winners	Bulk food stores
Food/Drink	Fish and seafood	Tea	Salsa	Vegetarian products	Microbrewery beer	Pita bread	Premium ice cream	Chocolate	Canadian wine	Vegetarian products
Media	The Globe and Mail	Business and financial section	W Network	Watch YouTube videos	Computer, science and technology magazines	Publish blogs regularly	Alternative rock	Communicate with people through social media	Classical music radio	Fashion magazine
Automotive	Sport coupes	Volkswagen	Sedans	Subaru	Midsize CUVs	Dodge	Under 10000 on latest vehicle	Honda	Nissan	Honda

Table 6.4: Resulting groups from the k-means algorithm and groups selected by user "A"

The following phase is to obtain the probabilities of belonging to each demographic cluster, based on the user selected groups. This is done using the values presented in the Table 6.5, which is also obtained as a result from applying the k-means clustering algorithm in our running example.

Total	C 1	C 2	C 3	C 4	C 5	C 6	C 7	C 8	C 9	C 10	
3125	1200	625	0	0	0	500	500	0	0	300	**DG-10**
4375	1072	0	2350	0	0	0	0	244	709	0	**DG-3**
5000	3256	656	0	496	224	0	112	256	0	0	**DG-2**
5000	1152	0	0	2125	0	0	1489	135	0	99	**DG-1**
6750	450	3457	0	750	1879	0	0	90	0	124	**DG-7**
3125	2725	0	0	400	0	0	0	0	0	0	**DG-4**
5000	1067	322	474	198	0	0	0	1825	0	1114	**DG-8**
3125	1280	0	0	0	0	1845	0	0	0	0	**DG-5**
3125	1845	500	0	300	0	240	0	0	240	0	**DG-6**
3125	1212	400	224	0	0	0	0	0	1289	0	**DG-9**
41750	**15259**	**5960**	**3048**	**4269**	**2103**	**2585**	**2101**	**2550**	**2238**	**1637**	**Total**

Table 6.5: Instances from each demographic cluster classified within each newly created group

In Table 6.5, the green cells represent the groups selected by user "A", as shown in Table 6.4. The blue cells stand for the demographic clusters from which some instances were classified into one or more user "A" selected groups. Finally, the yellow cells represent the values that will be used to obtain the probability of belonging to each one of these demographic clusters. The probability of belonging to each demographic cluster is obtained using equation (5.6). As an example, the following equation presents how the belonging probability for user "A" to the demographic cluster DG-1 is calculated:

$$P(u, DG-1) = \frac{1}{4}\left(\frac{0*100}{2585} + \frac{1489*100}{2101} + \frac{135*100}{2550} + \frac{99*100}{1637}\right) \quad (6.2)$$

Table 6.6 presents the resulting probabilities of belonging for user "A" to each one of the possible, previously obtained, demographic clusters (as presented in Table 6.2).

ID	Name	Description	Probability
DG-1	Grads & Pads	Young, lower-middle-class urban singles	20.55 %
DG-2	Electric Avenues	Young, middle-class urban singles and couples	3.84 %
DG-3	Grey Pride	Lower-middle-class, suburban apartment-dwelling seniors	2.39 %
DG-4	Startups & Seniors	Midscale mix of young and mature singles and couples	0.00 %
DG-5	Daytrippers & Nightowls	Young, mobile urban singles and couples	17.84 %
DG-6	Single City Renters	Young, apartment-dwelling urban singles and couples	2.32 %
DG-7	Young Digerati	Younger, upscale urban trendsetters	2.78 %
DG-8	Newcomers Rising	Young, downscale city immigrants	34.9 %
DG-9	Park Bench Seniors	Low-income seniors in urban high-rises	0.00 %
DG-10	White Picket Fences	Young, middle-income exurban families	15.37 %

Table 6.6: Probability of belonging to each possible demographic cluster for user "A"

The information presented in Table 6.6 can also be expressed as a vector, according to equation (5.2), as follows:

$$DemogClusters(u) = \left\{ \begin{array}{l} < DG-1, 20.55 >, < DG-2, 3.84 >, < DG-3, 2.39 >, \\ < DG-4, 0 >, < DG-5, 17.84 >, < DG-6, 2.32 >, \\ < DG-7, 2.78 >, < DG-8, 34.9 >, < DG-9, 0 >, < DG-10, 15.37 > \end{array} \right\} \quad (6.3)$$

Obtaining the user weights for the events' features

The following phase, according to the Algorithms (5.1) and (5.2), aims to obtain how important the different event features are for the user. In this case study, these features are represented and stored in our database as options for each event. Therefore, the user is now asked to rate each one of these options in terms of the importance thereof for him/her. This rate is done using numerical weights within the range $\{-\infty, \ldots, 0, \ldots, \infty\}$. Figure 6.4 presents the possible options for an event, along with the provided weights from user "A" in our running experiment.

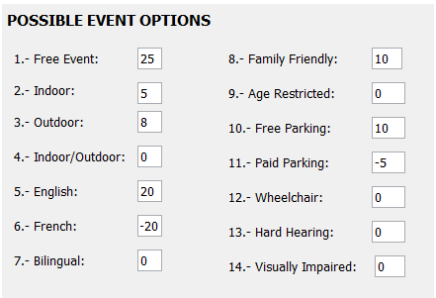


Figure 6.4: Possible event options with user weights in our running example

These weights assigned by user "A" can also be expressed as a vector, according to equation (5.8), as follows:

$$AttributeWeights(u) = \begin{Bmatrix} < FreeEvent, 25 >, < Indoor, 5 >, < Outdoor, 8 >, < English, 20 >, < French, -20 >, \\ < FamilyFriendly, 10 >, < FreeParking, 10 >, < PaidParking, -5 > \end{Bmatrix} \quad (6.4)$$

The user profile for user "A" in our running example, according to equation (5.1), is formed by these two previously created vectors, presented in equations (6.3) and (6.4). It is important to note that the user profile for each person is built only once, and remains stored by the system. Therefore, in further interactions with the system by the same user, he/she would only require to provide his/her current location and mobility level in order to obtain recommended events to attend.

Obtaining the user current position and mobility level

After the user profile has been built, the following phase, as presented in Algorithms (5.1) and (5.3), is to ask the user about his/her current position, as well as his/her mobility level. Figure 6.5 presents the current address and mobility level given by user "A" in our running example.

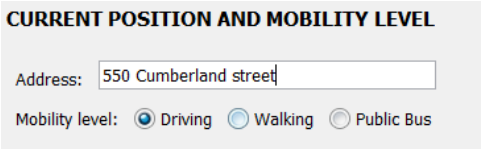


Figure 6.5: User "A" current position and mobility level

The latitude and longitude coordinates, corresponding to this address, are obtained using the developed Java software (as previously explained in Section 5.1). The corresponding coordinates for the address given by user "A", as depicted in Figure 6.5, are 45.425438 latitude and -75.685466 longitude.

Mapping the demographic clusters to the events

In order to be able to assess the demographic utilities for each item, first we require to link the events to be recommended, to each demographic cluster in the database. Recall, as explained in Section 5.1, linking item *types* to the demographic clusters instead of linking each item, presents several advantages. Therefore, in our case study we decided to link the identified items' categories in the database to the demographic groups in PRIZM C2. The mapping between both, demographic and events databases, was done considering the information of leisure activities from each demographic

group. We decided to use the leisure data, since it is the most suitable information contained in our demographic database that could be linked to our items' domain (events). As an example, Table 6.7 presents some of the created links between leisure activities from a demographic cluster and events' categories.

Leisure\Categories	Dance	Festival Fair	Films New Media	Seasonal celebrations	Sports and Outdoors	Museums	Music	Performance Theatre	...
Film festivals		X	X						...
Music festivals		X					X		...
Ballets	X							X	...
Museums						X			...
Whale watching				X	X				...
Theatre festivals		X						X	...
Skiing				X	X				...
Provincial parks					X				...
Nightclubs	X						X		...
...	...	...	...	...	...	...	...	...	...

Table 6.7: Mapping between leisure activities and event categories

Obtaining the demographic utility for each event

Using the mappings between leisure activities, of each demographic group, and the probabilities that user "A" may belong to each one of the demographic clusters, we proceed to assess the demographic utility of each event for this example user, as presented in Algorithms (5.1) and (5.4). However, in order to calculate this demographic utility, it is important to consider the following relation cardinalities. Each event may belong to one or more categories, each category, may be linked to one or more leisure activities, and each leisure activity may belong to one or more demographic clusters. Therefore, first we need to obtain the probabilities that user "A" likes each event category, using equation (5.10). Table 6.8 presents the resulting probabilities that user "A", from our running example, likes each item category based on his/her belonging probabilities to each demographic cluster (as presented in equation (6.3) and shown in Table 6.6).

Category	Probability
Sports and Outdoors	99.99 %
Music	62.31 %
Gallery	41.52 %
Museum	41.52 %
Dance	38.31 %
Film/New Media	24.39 %
Tour	20.55 %
Festival/Fair	6.16 %
Performance/Theatre	5.17 %

Table 6.8: Probabilities that user "A" likes each event category in our running experiment

After obtaining these probabilities, the following stage is to obtain the demographic utility to user "A" for each event. These demographic utilities are calculated using equation (5.11). Table 6.9 presents some events with their obtained demographic utility for user "A" in our running example.

Event Name	Categories	Probability
Begin2Believe Beach Volleyball with The Canadian Liver Foundation Stroll	Sport and Outdoor	99.99 %
Alterna Do It For Dad	Sport and Outdoor	99.99 %
Blue Rodeo	Music, Sport and Outdoor	81.15 %
The Wilderness Of Manitoba, and Orienteers live at Raw Sugar Cafe	Music	62.31 %
O'Brother, Sparrows and guests live at Cafe Dekcuf	Music	62.31 %
Lionheart, I Declare War, Molotov Solution and more live at Mavericks	Music	62.31 %
…		…

Table 6.9: Demographic utilities for some events for user "A"

Obtaining the weighted utility for each event

The following phase, as presented in Algorithms (5.1) and (5.4), is to obtain the weighted utility for the user for each item. This weighted utility, according to its definition presented in Section 5.1, is calculated by summing the user given weights for all the options contained by each event. This utility is obtained by applying equation (5.12) for each event. Table 6.10 presents some events with their weighted utility for user "A" in our running experiment.

Event Name	Free event	Indoor	Outdoor	English	Family friendly	Free parking	Paid parking	Wheelchair	Weight
Horse Show	X		X	X	X	X		X	**73**
National Tartan Day	X		X	X	X	X		X	**73**
Expression of Art	X	X		X	X	X		X	**70**
Sing me a Story - a spring concert	X	X		X	X	X		X	**70**
Ottawa Turkish Festival 2010	X	X		X	X	X	X	X	**68**
WESTFEST	X	X		X	X				**63**
…	…	…	…	…	…	…	…	…	**…**

Table 6.10: Weighted utilities for some events for user "A"

Obtaining the time and distance required to reach each event

In order to obtain the time the user would require to reach each event, as previously explained in Section 5.1, we developed a Java Software module that connects to Google Maps® and retrieves this

information. This software module sends two pairs of latitude and longitude coordinates, named as origin and destiny, along with the user mobility level, and retrieves the time required to go from one point to the other. Therefore, besides the user current position, we also require the spatial latitude and longitude coordinates for each possible venue or place where an event will take place. The spatial positions for the venues that were not mapped to any building or museum in the spatial database, were manually obtained from their address using the Google Earth® software system.

SQL spatial queries were used to obtain the distance from user "A" in our running example to each event. The events were grouped, according to their distance from the user within the following distance ranges, {0 to 0.5 km's}, {0.5 to 1.5 km's}, {1.5 to 3 km's}, {3 to 4.5 km's}, {4.5 to 6 km's}, {6 to 7.5 km's} and {more than 7.5 km's}. It is important to mention that the spatial information for buildings, museums, rivers, roads, etc. contained in the shape files, and obtained from the Ottawa Open Data foundation, are projected in MTM Zone 9 NAD83 (CSRS) reference system. This reference system is a Canadian Quebec and Ontario specific spatial reference system. However the user current location and the event venues' latitude and longitude coordinates, were obtained using the World Geodetic reference system (WGS 84) used by GoogleMaps® and Google Earth® systems. Therefore we developed a Java software module which converts spatial coordinates from the International reference system to the Canadian Quebec and Ontario specific reference system. For more information on the Canadian Quebec and Ontario specific reference system, we recommend [86]. Table 6.11 presents some events with their obtained distance range and time from user "A" in our running example.

Event Name	Distance (Km's)	Time (minutes)
A Celebration of Remembrance	0.5	2
Motets and Melodrama	0.5	3
Indonesian Night	1.5	5
War and Medicine	1.5	9
Ottawa Alleyways	3	9
Families in Nature	More than 7.5	41
…	…	…

Table 6.11: Distance and Time utilities for some events for user "A"

Creating the user preference model

After calculating the four items' utilities, which serve as criteria to create the user preference model, we now require to obtain from the user his/her weak preference model. The user weak preference model is a global ranking provided from the user to some events, considering the already obtained utilities for each one of them. The number of events that the user should be asked to rank, has to be carefully selected. This number of events has to be decided considering that, while more items the user ranks, the more elements the UTA* algorithm will have to learn a model, and therefore more accurate and better user preference models would be produced. However, on the other hand, if the user is presented with a large number of events to rank, he/she may get lost and confused between so many choices and end up with an order that doesn't represent the real user preferences. For more information about the tradeoffs that need to be considered when displaying information to the users, and the number of items and layout of them, we recommend [87]. Lakiotaki, et al. [11], proposes that the users should provide his/her weak preference order in at least 5 items, in order to obtain accurate preference models when applying the UTA* algorithm. However Lakiotaki, et al. [11], asked the users to rank an average of 10 items to create their user preference models. Therefore, following [11], we decided to ask the users to also provide their preferences over 10 items. Consequently, the system presents to the user ten randomly selected events, along with their calculated utilities and all their features used to obtain these utilities. The user is asked to rank those events in a non-descending order, based on which events he/she would prefer to attend. Table 6.12 presents the events that user "A" in our running example was asked to rank, along with the order provided by him/her.

After we have obtained the user weak preference order, the next stage, as presented in Algorithms (5.1) and (5.6), is to apply the UTA* algorithm, described in Section 3.1.1, to obtain the user preference model. The UTA* algorithm is applied using an open source Java software module. For more information on this Java software to implement the UTA* algorithm, we recommend [88].

The obtained user preference model for user "A" in our running example is presented in equation (6.5). Recall from our methodology as presented in Chapter 5, that the user preference model is a vector of four user weights. Each one representing how much the user "cares for" each criterion, when deciding an event to attend. Hence, the user preference model includes the user weights for the demographic, weighted, distance, and time utilities, assessed for each event.

$$PM(u) = \{< W_{DU} = 0.12 >, < W_{WU} = 0.56 >, < W_{DisU} = 0.14 >, < W_{TU} = 0.17 >\} \qquad (6.5)$$

Event Name	Categories	Weights	Free event	Indoor	Outdoor	English	French	Bilingual	Family friendly	Free parking	Wheelchair	User Rate
Jimmy Hunt **(6 km's, 24 minutes)**	Music **(62.31 %)**	**15**		X						X		**9**
Lost sox cabaret **(3 km's, 8 minutes)**	Music **(62.31 %)**	**0**										**8**
Stairwell Carollers Spring Concert **(4.5 km's, 16 minutes)**	Music, Seasonal Celebration, Other **(62.31 %)**	**10**						X	X			**6**
2011 Chinese New year Festival Event at Tudor Hall **(4.5 km's, 20 minutes)**	Festival/Fair, Heritage and Traditions, Other **(6.16 %)**	**60**	X	X		X		X		X	X	**3**
Mosaika - Sound and Light Show **(1.5 km's, 5 minutes)**	Film/New Media, Special Event **(24.39 %)**	**43**	X		X			X	X		X	**1**
54-40 **(7.5 km's, 19 minutes)**	Music **(62.31 %)**	**20**				X						**10**
Exhibit of Gatineau Hills by Margit Hideg **(1.5 km's, 7 minutes)**	Gallery **(41.52 %)**	**25**	X			X	X					**4**
Family astronomy and space workshops **(4.5 km's, 14 minutes)**	Museum **(41.52 %)**	**0**						X				**5**
Bob Walsh **(1.5 km's, 9 minutes)**	Music **(62.31 %)**	**15**		X						X	X	**7**
Black History Month Nigerian Panorama **(0.5 km's, 3 minutes)**	Film/New Media, Performance/Theatre **(14.78 %)**	**35**		X		X			X		X	**2**

Table 6.12: User "A" weak preference order

Recommend the events with the highest integrated utility

Finally the last phase, according to Algorithms (5.1) and (5.7) as presented in Chapter 5, is to obtain the integrated utility for each item for the user, using the previously created user preference model. The integrated utility for each item is calculated using equation (5.14). Once the integrated utilities for each item have been assessed, the system presents to the user a list of the events ordered by their utility to the user. Table 6.13 presents the top part of the recommended events list for user "A" in our running experiment. Figure 6.6 depicts the phases followed by the PeRS to recommend events to a user.

Event Name	Categories	Demographic Utility	Weighted Utility	Distance Utility	Time Utility	Final Utility
National Tartan Day	Music, Special Event	62.31	73	1.5	5	**0.920**
Capital Vélo Fest	Festival/Fair, Sport and Outdoor, Other	53.075	63	1.5	4	**0.833**
Documentary of Seb-i Arûs (Whirling Dervishes Ceremony)	Film/New Media	24.39	65	0.5	3	**0.832**
Austrian Embassy Arts Cafe jazz concert	Music	62.31	60	1.5	7	**0.817**
Ottawa Turkish Festival Flag Raising Ceremony	Festival/Fair	6.16	68	1.5	3	**0.814**
Ottawa Turkish Festival Opening Gala	Festival/Fair	6.16	68	1.5	4	**0.813**
Ottawa Turkish Festival 2010	Festival/Fair	6.16	68	1.5	5	**0.812**
SuzukiMusic Open House	Music, Tour	41.43	70	4.5	15	**0.804**
Tim Hortons Ottawa Dragon Boat Festival	Festival/Fair, Sport and Outdoor	53.075	68	4.5	15	**0.803**
Olympian Kristina Groves and Right To Play Coaches from Rwanda Discuss the Impact of Sport	Special Event, Sport and Outdoor	99.99	55	3	11	**0.793**
...	...	...	...	...	...	...

Table 6.13: Recommended events to user "A"

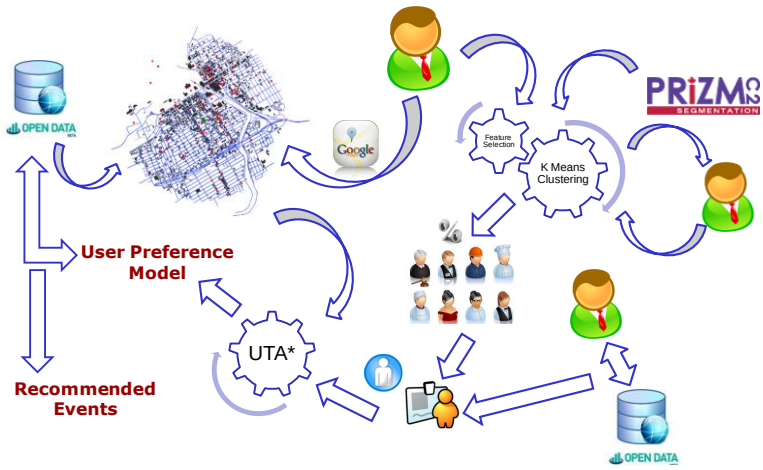


Figure 6.6: Process followed by PeRS to recommend events

6.2 Experimental Design

In order to evaluate our proposed methodology for RSs, we performed an offline experiment to our case study PeRS. We decided to perform an offline experiment because this kind of experiment only evaluates the recommendation technique being used, and therefore it is not biased by the system interface or possible system malfunctions. The offline experiment was performed with the goal of evaluate the system's prediction accuracy, coverage, and confidence properties, as previously described in Section 3.4.2. These properties were selected considering the scope of offline experiments, the specific application domain, and our proposed recommender engine. Therefore the user preference and trust properties, that cannot be measured using offline experiments, were not considered. Additionally, following the results and conclusions presented in [52], since our recommender engine generalizes the user profile from a large-scale demographic database, the final prediction accuracy is likely to be reduced. Therefore it becomes more important to measure the obtained prediction accuracy than other properties that sacrifice it, such as novelty, serendipity, or diversity. The risk and robustness properties, on the other hand, don't apply to our application domain (i.e. an event recommender). Finally, while the privacy, adpativity, and scalability properties were considered to be out of the scope for this evaluation, the utility metric is only useful when we are comparing more than one RS, due it partially depends on the items contained in the database.

Recall from Section 3.4.1 that in order to apply an offline experiment to evaluate a RS, we first require to gather some users' information. Therefore, we designed a questionnaire to be asked to some users, asking them information that we later used to create the user profiles and evaluate the system results. The following Sub-Section presents the structure of the designed questionnaire.

To determine to how many users we require to apply the questionnaire, we decided to first perform a pilot study with 30 users to obtain a punctual estimation of the standard deviation of the population. Recall from Section 3.4.2 that according to the central limit theorem, this punctual estimation can be used in equation (3.40) to obtain the number of users required to achieve a desired confidence in our results.

6.2.1 Designed Questionnaire for Users

The questionnaire applied to the users, was designed to simulate as closely as possible the real system execution. Therefore, it consists of four different parts, applied in two iterations, that simulate the interactions that the system would have with the users, in accordance to our algorithm, as presented in Chapter 5. The first part, asks the user 9 different questions about their personal information, as presented in Figure 6.3. With the purpose of analysis the data and results, besides the 9 questions presented in Figure 6.3, we also asked the users for their gender. The gender however, was not used to identify the demographic groups where the user may belong, since this information is not included in our demographic database. For our experiment, the current mobility level of the users was considered the same as their mode of transport provided as one of the 9 questions asked in this first part of the questionnaire. The second part of the questionnaire, asks the user to provide his/her weights to the identified events' attributes, as depicted in Figure 6.4. The end of part two also represents the end of the first iteration with the user. The following parts of the questionnaire were asked to the users, after the given information in parts one and two was processed.

The third part of the questionnaire presents to the user the resulting groups from applying the k-means clustering algorithm to the resulting demographic clusters, based on the provided information in part one. The user is asked to select the groups with the activities that he/she identifies the most, as shown in Table 6.4. Finally, the last part of the questionnaire presents to the user 20 events with their complete information about distance, time, attributes, and categories. The user is asked to rank these events, considering all the displayed information about them. It is important to notice that, following [11], and as previously explained in Section 6.1.2, only 10 from the 20 ranked events by the users will be used as the user weak preference order. Therefore, only 10 of them, randomly selected, will be considered by the UTA* algorithm to create the user preference model. However, the 20 events will later be used to evaluate the prediction accuracy of the system, by comparing the produced ordered ranked list against them. This idea of hiding some user ranked items when creating the user preference model, and then evaluate if the system is capable to predict their ordering, follows the

methodology to evaluate the prediction accuracy of a ranking system using offline experiments, as presented in Section 3.4.1.

In order to avoid bias in the results obtained from our experiment, the shown events to be ranked in the fourth part of the questionnaire were randomly selected for each user. Additionally, the names of the events that the users were asked to rank were hidden, forcing the users to rank them, based on their information and attributes, and not their title. Table 6.14 presents the structure of the questionnaire applied to the users.

Iteration	Part	Gathered Information
First Iteration	**Part one**: Personal Information	The user is asked 9 questions about his/her personal information, along with his/her mobility level.
	Part two: Preferences	The user is asked to provide his/her weights to the identifies events' attributes.
Second Iteration	**Part three**: Activities	The user is asked to select the groups with the activities that he/she identifies the most.
	Part four: Events	The user is asked to rate 20 different events, considering their information about distance, time, categories, and attributes.

Table 6.14: Structure of the questionnaire applied to the users

The following Section, presents the users' answers from the applied questionnaires, and an analysis of the obtained values for each one of the selected system properties to be measured.

6.3 Experimental Application and Analysis of Results

Within this section we present the information gathered from applying the designed questionnaire to the user, and show the metric values measured for each of the system properties selected to be evaluated in our system (i.e. prediction accuracy, coverage, and confidence). We provide an analysis of the obtained results with the following goals. Firstly, we aim to typify the users considered in our experiment. Secondly, we are interested in evaluating the system accuracy and its ability to cluster the user in the most adequate demographic groups. Another important issue is to learn the system accuracy from the users' information or events shown to them. Finally, we evaluate the system coverage in terms of user and item space coverage. Each one of these analyzes is presented in detail within the following Sections.

6.3.1 Typifying the Users

As a first step, we aim to learn who our users are and identify their distributions, in order to determine which type of users are being considered in our experiment. As described in the previous

Section we first applied, as a pilot study, the designed questionnaire to 30 users. Table 6.15 presents the user information collected from the first part of the questionnaire (i.e. personal information), after the first iteration was completed. From these information, we observe that we have 28 out of 30 different types of users (i.e. 93.33% of the users give different answers to the questions), only the user pairs 4, 21, and 25, 26, present the exact same information in all the 10 questions asked to them. This wide spread distribution of users, allow us to test the system behaviour and produce recommendations under different scenarios, and therefore draw more reliable conclusions that will be true for a larger number of people of the general population. Furthermore, in order to categorize the type of users that answered the questionnaire, and identify those that are not covered in our experiment, we analyzed the user distribution for each one of these 10 features. Figure 6.7 presents these user distributions for each criterion, obtained using WEKA.

Id	Gender	Age	Mother Tongue	Marital Status	Mode Transport	Occupation	Education	Family Status	Age Children	Tenure
1	Male	45-64	English	Married	Car	White Collar	University Degree	Couples With Kids	18-24	Owned
2	Female	45-64	English	Married	Car	White Collar	University Degree	Couples With Kids	18-24	Owned
3	Male	25-44	English	Single	Public	White Collar	University Degree	Non Family	0	Rented
4	Female	25-44	English	Married	Car	White Collar	University Degree	Couples No Kids	0	Rented
5	Female	45-64	Non Official	Single	Public	Service Sector	No Cert/Dipl/Deg	Non Family	0	None
6	Female	65-74	Non Official	Wid/Div/Sep	Car	None	No Cert/Dipl/Deg	Lone Parent	25+	Owned
7	Female	75-84	Non Official	Wid/Div/Sep	Car	None	Trade	Non Family	0	Owned
8	Male	45-64	Non Official	Married	Car	White Collar	University Degree	Couples With Kids	18-24	Rented
9	Female	45-64	English	Married	Car	Blue Collar	University Degree	Couples With Kids	18-24	Rented
10	Female	15-24	Non Official	Single	Car	None	High School Cert	Non Family	0	None
11	Male	15-24	Non Official	Single	Car	None	No Cert/Dipl/Deg	Non Family	0	None
12	Female	45-64	Non Official	Married	Car	Blue Collar	University Degree	Couples With Kids	15-17	Owned
13	Male	45-64	Non Official	Married	Car	White Collar	University Degree	Couples With Kids	15-17	Owned
14	Male	25-44	Non Official	Single	Public	Blue Collar	Some University	Couples No Kids	0	Rented
15	Female	15-24	English	Single	Car	None	Some University	Non Family	0	None
16	Male	45-64	English	Married	Car	White Collar	University Degree	Couples With Kids	15-17	Owned
17	Female	15-24	English	Single	Public	None	No Cert/Dipl/Deg	Non Family	0	None
18	Male	25-44	Non Official	Single	Car	Blue Collar	University Degree	Couples No Kids	0	Rented
19	Female	25-44	English	Single	Car	White Collar	University Degree	Couples No Kids	0	Rented
20	Male	45-64	Non Official	Wid/Div/Sep	Car	White Collar	Some University	Lone Parent	15-17	Rented
21	Female	25-44	English	Married	Car	White Collar	University Degree	Couples No Kids	0	Rented
22	Male	25-44	Non Official	Married	Car	White Collar	University Degree	Couples No Kids	0	Rented
23	Female	25-44	Non Official	Married	Car	White Collar	University Degree	Couples With Kids	<6	Rented
24	Male	25-44	Non Official	Single	Car	White Collar	University Degree	Non Family	0	None
25	Male	25-44	English	Single	Car	White Collar	University Degree	Non Family	0	None
26	Male	25-44	English	Single	Car	White Collar	University Degree	Non Family	0	None
27	Male	25-44	English	Single	Car	White Collar	University Degree	Lone Parent	6	None
28	Male	25-44	English	Married	Public	White Collar	University Degree	Couples No Kids	0	Rented
29	Male	45-64	English	Wid/Div/Sep	Car	Blue Collar	University Degree	Lone Parent	6-14	Rented
30	Female	25-44	English	Married	Car	White Collar	University Degree	Couples With Kids	<6	Rented

Table 6.15: Collected user information from the first section of the questionnaire

From the data distribution, as presented in Figure 6.7, we observe that we have approximately the same number of male and female users. Moreover, we find that the majority of our users are between

25 and 64 years old (i.e. 80%), drive a car (i.e. 83.33%), have a white collar education (i.e. 60%), and a University degree (i.e. 70%). Almost half of the users have no kids (i.e. 56.66%), and from the remaining 43.33% of users who have kids, 30.76% are single parents. Moreover, 75% of the single parents are widowed, divorced or separated users. On the other hand, users that are poorly represented in our experiment with just a 3.33%, work in the service sector, or have a trade education or a high school certificate, or have children between 6-14 years old, or older than 25 years. Finally the users not considered at all within this experiment are those older than 84 years, or whose mother tongue is French. It is important to note that the mother tongue represents the language of the place where the user was born, not if the user speaks or understands it.

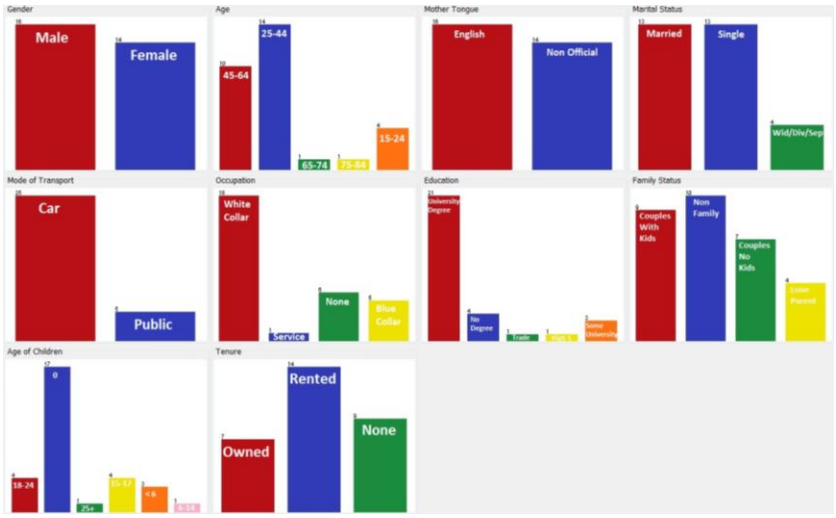


Figure 6.7: User distribution by personal information

Table 6.16 presents the information obtained from the users about their events' preferences, given as weights at the section two of the questionnaire during the first interaction with them. At the bottom part of this table we also present the average weight for each criterion for the users considered in this experiment. Figure 6.8 presents a chart of these obtained average weights for each criterion.

From Table 6.16 and Figure 6.8 we observe that our users are mainly interested to attend to free events with free parking and not in French. The fact that an event presented in French has a negative influence in the users, when deciding to which event to attend, was to be expected, since none of our users has French as a mother tongue. Additionally, we see that the only other event option that has a negative impact in our users, is paid parking. Moreover we observe how the range of ages from our users and the number of users with kids, as previously identified, is closely reflected in the weights assigned to the events options. For instance, the special accessibility options (i.e. wheelchairs, hard hearing and visually impaired), mostly used by seniors, obtained the lowest average weights. Recall

that only the 6.66% of our users are older than 64 years. Furthermore, the event option that received the next lowest weight is the age restricted feature, which matches with the fact that 86.66% of the users are older than 24 years, and only 43.33% of them have kids, therefore less than the half of the users have concerns about this issue.

User	Free Event	Indoor	Outdoor	Indoor and Outdoor	English	French	Bilingual	Family Friendly	Age Restricted	Free Parking	Paid Parking	Wheelchair Accessible	Hard Hearing	Visually Impaired
1	20	10	0	0	30	-30	20	0	0	15	-10	0	0	0
2	10	0	0	0	15	-30	30	20	-20	30	-10	10	0	0
3	20	20	10	12	25	0	0	10	18	30	-20	15	0	5
4	30	20	20	30	30	-30	20	15	-10	30	20	0	0	0
5	30	10	20	15	0	-30	0	20	0	0	0	0	0	0
6	0	20	-10	5	-10	-30	0	20	0	10	0	28	12	12
7	15	25	10	10	0	-20	0	15	5	20	-5	20	0	0
8	0	20	5	0	12	-25	0	20	15	20	-5	0	0	0
9	16	20	20	20	15	-20	15	25	5	15	-3	0	0	0
10	30	12	25	20	20	-30	15	18	20	20	-10	0	0	0
11	30	15	25	15	15	-30	10	27	-5	25	-10	0	0	0
12	15	0	0	0	-30	-30	0	0	0	20	-15	15	15	15
13	5	20	-5	5	-10	-30	15	0	0	20	-10	20	20	20
14	30	20	30	25	-5	-30	0	20	15	0	0	0	0	0
15	30	15	25	20	15	-25	0	15	25	20	-5	0	0	0
16	20	15	20	20	30	-30	20	0	0	20	-10	0	0	0
17	25	15	25	0	25	-25	20	25	-10	0	0	0	0	0
18	20	10	10	20	0	-30	10	0	0	30	-10	0	0	0
19	15	20	20	20	20	-20	10	15	15	15	-5	0	0	0
20	10	20	10	20	-10	-30	10	20	20	30	0	0	0	0
21	15	25	25	30	30	-30	0	20	20	20	15	0	0	0
22	20	25	25	30	30	-30	15	0	0	25	20	5	5	5
23	20	15	25	20	20	-20	10	20	15	25	-10	15	0	0
24	25	15	10	15	0	-25	0	10	20	25	12	0	0	0
25	20	0	0	0	0	-30	30	0	0	0	0	0	0	0
26	10	10	-10	20	10	-10	10	20	-10	15	-10	0	0	0
27	30	10	10	10	0	-30	10	20	5	30	-30	0	0	0
28	15	0	10	0	20	-30	15	20	10	5	-5	0	0	0
29	25	10	25	20	20	10	25	25	-10	15	-5	0	0	0
30	30	20	-10	10	30	-30	30	30	20	30	-5	0	0	0
Average:	**19.37**	**14.57**	**12.33**	**13.73**	**11.57**	**-25**	**11.33**	**15**	**5.43**	**18.67**	**-4.2**	**4.27**	**1.73**	**1.9**

Table 6.16: Users' given weights for each event option

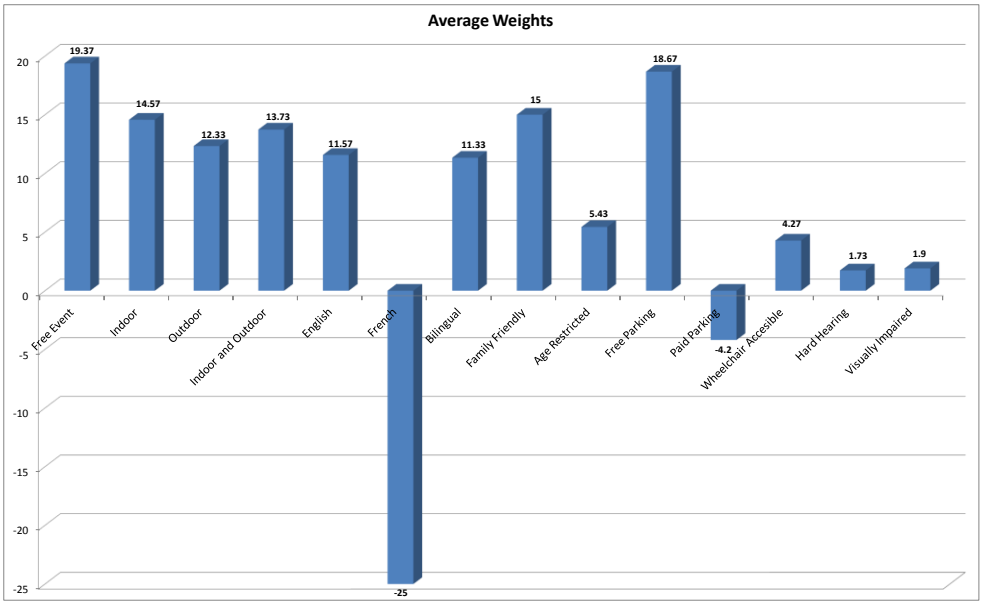


Figure 6.8: Average weights for each event criterion

6.3.2 System Ability to Cluster Users in Demographic Groups

Next, we seek to analyze how efficient the system is to identify the demographic clusters where the users may belong, based only on the 10 selected demographic questions asked to them during the first part of the designed questionnaire.

Once we have gathered the user information, following our proposed methodology as presented in Algorithm (5.1), we computed the possible demographic groups from the PRIZM C2 database where each user may belong. Table 6.17 presents the identified matching clusters for each user, based on their given personal information (as presented in Table 6.15).

User	Possible Matching Groups	# of Groups
1	U1-1, U1-2, S1-3	3
2	U1-1, U1-2, S1-3	3
3	U2-31, U7-44 ,U2-15	3
4	U2-4, U2-15, U2-31	3
5	U5-64, U5-42, U7-44	3
6	U3-33, U3-37, U3-18	3
7	U7-62, S4-38, U3-33	3
8	U1-9, S2-20, S2-5, U5-46	4
9	U1-9, U1-2, U4-28, S4-14, U1-1, U5-46, S2-29, S4-38, U1-8, E1-12	10
10	U5-64, R2-63, U7-60	3
11	R2-63	1
12	S2-5, S2-21, E1-7	3
13	S2-5, U1-1	2
14	U7-44, U5-64	2
15	U7-59, U7-60, R2-63, U7-49, U4-54, U4-53, U4-28, E2-41, S4-43, U7-62	10
16	U1-1, U1-2, S1-3	3
17	U5-64, U7-60, U5-42, U7-44, U7-59	5
18	U2-31, U7-44, U5-46	3
19	U2-31, U2-15	2
20	U5-46, U5-64, U7-62, U7-49, U7-60, U7-44, U5-42, U6-52, U6-48	9
21	U2-4, U2-15, U2-31	3
22	U2-4, U7-44, U2-31, U2-15, U5-46, U1-9	6
23	U5-46, U7-44, U2-4, U1-9	4
24	U2-31, U7-44, U2-4, U2-15	4
25	U2-31, U2-15, U2-4	3
26	U2-31, U2-15, U2-4	3
27	U2-31, U2-15, U2-4, U7-60, U3-13	5
28	U2-4, U2-31, U2-15, U7-44	4
29	U7-60, U7-59, U7-62, U7-49, S4-38, U4-53, U4-28	7
30	U2-4, U2-15, U1-9, U3-13, U2-31	5
	Average:	**4**

Table 6.17: Possible demographic groups where the users may belong

Table 6.17 also presents the average number of clusters selected by the system to which the users may belong. In this case, the average number of clusters a person belongs to is 4 (out of 66 contained in the demographic database). This average number of possible clusters allows us to measure the goodness of the system to filter out and identify the demographic groups where a user may belong; based only on the 9 previously selected questions and the established threshold, as presented in equation (6.1). This ability of the system to narrow the number of possible clusters can be expressed as a percentage, where 100% means that the system narrows the possible options to 1 single matching cluster, and 0% that the system selects the 66 groups as possible ones. This percentage, from now on referred as the system narrowing percentage, is computed using the following equation:

$$\text{Narrowing \%} = 100 - \frac{(MatchingGroups - 1) * 100}{65} \qquad (6.6)$$

Therefore, applying equation (6.6) to the calculated average number of possible demographic groups (i.e. 4 matching groups in average), we obtain a narrowing percentage of 95.38%. This indicates that our system is capable to narrow the possible matching groups in a 95.38%, with the selected 9 questions and the established threshold.

The computed possible demographic clusters for each user, according to our proposed methodology as presented in Chapter 5, serve to create the groups of life style activities (i.e. same number of groups as possible demographic clusters) from which the user selects the ones where he/she identifies the most. Therefore these life style groups were created for each user and form the third part of the questionnaire that the users answered during the second iteration.

The last part of our questionnaires is created for each user by randomly selecting 20 events from our database. Recall that the user is asked in this part of the questionnaire to provide his/her global preferences by ordering these events. Though, in order for a user to provide his/her preferences, we need to present him/her all the necessary information about each event. The information shown to the user for each event includes the categories and options linked to it, and the time and distance they would require to reach each event from their position. Figure 6.9 depicts as red points the users positions, as orange points the places and venues where the events are going to be held, the roads are painted in black, rivers in blue, and buildings in green. This figure was obtained using the SQL Spatial Query Visualizer® software and the spatial databases previously described in Section 6.1.

Figure 6.9: Users' positions

Once parts three and four of the questionnaire have been created for each user, the second iteration of the questionnaire was asked to them.

Table 6.18 presents the number of selected life style groups by each user in the third part of the questionnaire, during the second iteration. This table also presents the computed probabilities of belonging to each one of the identified possible demographic groups, presented in Table 6.17. Recall that the probability of belonging to each demographic group is computed using the equation (5.6) and the instances distribution table obtained from the k-means clustering algorithm, as exemplified in Table 6.5.

User	# of Possible Groups	# of Selected Groups	Probability of belonging
1	3	2	U1-1 (75.36%), U1-2 (13.94%), S1-3 (10.7%)
2	3	2	U1-1 (50%), U1-2 (14.19%), S1-3 (35.81%)
3	3	3	U2-31 (31.5%), U7-44 (26.71%), U2-15 (41.79%)
4	3	2	U2-4 (10.02%), U2-15 (33.09%), U2-31 (56.89%)
5	3	2	U5-64 (12.8%), U5-42 (56.1%), U7-44 (31.1%)
6	3	2	U3-33 (28.52%), U3-37 (26.25%), U3-18 (45.22%)
7	3	2	U7-62 (33.85%), S4-38 (49.92%), U3-33 (16.23%)
8	4	2	U1-9 (19.03%), S2-20 (40.45%), S2-5 (19.33%), U5-46 (21.18%)
9	10	7	U1-9 (1.66%), U1-2 (3.78%), U4-28 (6%), S4-14 (12.78%), U1-1 (18.56%), U5-46 (17.94), S2-29 (8.3%), S4-38 (16.15%), U1-8 (10.63%), E1-12 (4.2%)
10	3	2	U5-64 (100%), **R2-63 (0%), U7-60 (0%)**
11	1	1	R2-63 (100%)
12	3	2	S2-5 (12.61%), S2-21 (87.39%), **E1-7 (0%)**
13	2	2	S2-5 (40.22%), U1-1 (59.78%)
14	2	1	U7-44 (100%), **U5-64 (0%)**
15	10	7	U7-59 (13.78%), U7-60 (8.03%), R2-63 (4.26%), U7-49 (20.37%), U4-54 (9.05%), U4-53 (9.29%), U4-28 (10.16%), E2-41 (14.4%), S4-43 (3.97%), U7-62 (6.7%)
16	3	2	U1-1 (75.36%), U1-2 (13.94%), S1-3 (10.7%)
17	5	3	U5-64 (8.06%), U7-60 (25.65%), U5-42 (10.59%), U7-44 (43.73%), U7-59 (11.97%)
18	3	1	U2-31 (100%), **U7-44 (0%), U5-46 (0%)**
19	2	2	U2-31 (67.97%), U2-15 (32.03%)
20	9	4	U5-46 (24.13%), U5-64 (34.98%), U7-62 (3.99%), U7-49 (0.65%), U7-60 (8.95%), U7-44 (9.16%), U5-42 (1.75%), U6-52 (8.2%), U6-48 (8.2%)
21	3	3	U2-4 (27.27%), U2-15 (29.59%), U2-31 (43.14)
22	6	5	U2-4 (3.48%), U7-44 (3.93%), U2-31 (20.26%), U2-15 (28.33%), U5-46 (22.31%), U1-9 (21.69%)
23	4	3	U5-46 (7.31%), U7-44 (33.11%), U2-4 (46.32%), U1-9 (13.26%)
24	4	3	U2-31 (19.01%), U7-44 (22.11%), U2-4 (44.09%), U2-15 (14.79%)
25	3	2	U2-31 (52.42), U2-15 (16.68), U2-4 (30.89)
26	3	2	U2-31 (20.11%), U2-15 (38.98%), U2-4 (40.91%)
27	5	5	U2-31 (22.22%), U2-15 (13.7%), U2-4 (34.15%), U7-60 (20.69%), U3-13 (9.24%)
28	4	2	U2-4 (28.73%), U2-31 (18.34%), U2-15 (42.2%), U7-44 (10.73%)
29	7	6	U7-60 (18.29%), U7-59 (14.27%), U7-62 (20.98%), U7-49 (4.47%), S4-38 (18.02%), U4-53 (19.5%), U4-28 (4.47%)
30	5	5	U2-4 (29.67%), U2-15 (23.92%), U1-9 (6.95%), U3-13 (21.92%), U2-31 (17.54%)

Table 6.18: Probabilities of belonging to each possible demographic group

From the results obtained in the third part of the questionnaire, and presented in Table 6.18, we conclude that the system was capable, for an 86.66% of the users, to accurately identify the demographic groups where they belong. This statement follows from the observation that 26 out of 30 users identified themselves, and therefore present a probability of belonging, in every one of the

identified possible demographic groups for him/her. Only for the users 10, 12, 14 and 18, the system selected some demographic groups as possible ones, where they didn't identify themselves (i.e. 0% probability). These selected and non-matching demographic groups for a user are marked in bold in Table 6.18. Recall, from our methodology, that these probabilities are computed based on the life style groups of activities selected by the users. Furthermore, the life style activities considered to create the groups are the ones identified by PRIZM C2 for each of the obtained possible demographic groups where each user may belong.

Additionally, from the results presented in Table 6.18, we observe that the two pairs of users marked in the table (i.e. 4, 21 and 25, 26), that initially appeared to be the same user, since they had the exact same answers for the 10 asked demographic questions, now are clearly differentiated by their probabilities of belonging to each demographic group, even when the groups are the same. Consequently, these results confirm the importance of considering the non-demographic information (i.e. life style and social values data), contained in the large-scale demographic databases, when classifying a user into one or more of its identified groups.

6.3.3 System Accuracy

We next present the prediction accuracy of the system for each user considered in our pilot study. Furthermore we obtain the standard deviation of it to later use it as a punctual estimation of the standard deviation of the population, and obtain the number of users we require for our offline experiment to obtain results with a desired confidence.

According to our methodology, as presented in Algorithm (5.1), once we computed the probabilities of belonging to each demographic group, and gathered the user weights for each event option, we proceeded to create a user preference model for each user. Subsequently, using these users' preference models, we obtained the list of recommended events for each user. After obtaining these lists of recommended events, we computed the prediction accuracy for each one of them, using equation (3.39) and following the methodology presented in Section 3.4.2. Table 6.19 presents the prediction accuracies obtained for each one of the users considered in the pilot study, along with their average accuracy and standard deviation.

User	Accuracy (%)	User	Accuracy (%)
User 1	86.41%	User 16	75.41%
User 2	82.31%	User 17	93.88%
User 3	77.93%	User 18	71.07%
User 4	71.94%	User 19	91.44%
User 5	77.65%	User 20	73.63%
User 6	89.83%	User 21	76.65%
User 7	70.11%	User 22	63.69%
User 8	74.19%	User 23	94.12%
User 9	60.75%	User 24	87.50%
User 10	88.24%	User 25	67.50%
User 11	89.13%	User 26	62.89%
User 12	68.40%	User 27	66.18%
User 13	80.84%	User 28	73.51%
User 14	96.13%	User 29	77.97%
User 15	95.79%	User 30	89.58%

Average:	79.16%
Standard Deviation:	10.62

Table 6.19: Prediction accuracies obtained

The standard deviation of the obtained accuracies, from our pilot study, can now be considered as a punctual estimation of the standard deviation of the population, according to what is established by the central limit theorem. Additionally, in order to obtain the required number of users for our offline experiment, using equation (3.40), we decided to set a confidence of 95% and an allowed sample error of 5%. The percentages used for the confidence and the error, are the most used and accepted values for highly accurate statistical tests, according to Berenson, et al. [58]. Therefore, the number of required users for the offline experiment, to obtain a confidence of the 95% with an error of 5% is obtained as follows:

$$n = \frac{(1.96)^2 * (10.62)^2}{(5)^2} = 17.33 \approx 18 \qquad (6.7)$$

Here a z-value of 1.96 is used, since it is the corresponding value for an area of 95% in the normal distribution table. Recall that according with the central limit theorem, as presented in Section 3.4.2, when we consider a sample of users of at least 30 people, we can assume a normal distribution to infer results for the general population. From equation (6.7), we obtain that we require to apply our questionnaire to at least 18 users in our offline experiment. However, since the required number of users is less than the users considered in the pilot study, the users and results of the pilot study can be considered as the users and results of the offline experiment as well. Finally, the next stage consists

into obtain the confidence interval for the prediction accuracy for the population, using equations (3.41) and (3.42) as follows:

$$\mu = 79.16 \pm 1.96 \frac{10.62}{\sqrt{30}} = \{75.36,\ 82.96\} \qquad (6.8)$$

Consequently we conclude with a 95% confidence, that our PeRS system will have a prediction accuracy between 75.36% and 82.96%. Recall that the accuracy expresses the effectiveness of the system to rank the items in the same way the user would do it. Therefore we conclude that our methodology, as implemented in PeRS, is capable to produce for a 95% of the general population recommendations that will match in an average of 79.16% with what the user would have selected.

6.3.4 Learning the System Accuracy

The next step of our evaluation is to learn, by means of data mining and statistical techniques, the system accuracy from the users' data and considered events to create the user preference models. That is, we aim to analyze if the system performs better for specific types of users or if we could improve the prediction accuracy by selecting in a different way (i.e. not randomly) the events used to create the user preference models.

In order to conduct these analyses, the accuracies obtained were grouped in three categorical groups or classes to be learned from the data. The following are the selected class labels for each group: "Good" (i.e. accuracies between 80%-100%), "Fair" (i.e. accuracies between 70%-80%), and "Poor" (i.e. accuracies between 60%-70%). Figure 6.10 presents the distributions of obtained accuracies in the offline experiment, grouped by their class label. All data analyses were done using WEKA.

User	Accuracy label
User 1	Good
User 2	Good
User 3	Fair
User 4	Fair
User 5	Fair
User 6	Good
User 7	Fair
User 8	Fair
User 9	Poor
User 10	Good
User 11	Good
User 12	Poor
User 13	Good
User 14	Good
User 15	Good

User	Accuracy label
User 16	Fair
User 17	Good
User 18	Fair
User 19	Good
User 20	Fair
User 21	Fair
User 22	Poor
User 23	Good
User 24	Good
User 25	Poor
User 26	Poor
User 27	Poor
User 28	Fair
User 29	Fair
User 30	Good

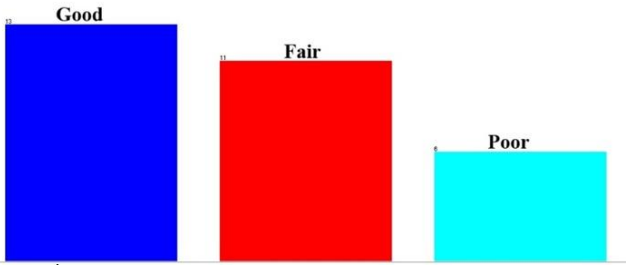


Figure 6.10: Distribution of obtained accuracy

Once the obtained accuracies were grouped within the proposed three classes, we proceeded to analyze their distribution over the considered users in this experiment and the types of events presented to them to be ranked (i.e. events used to create the preference models). These analyzes are presented in detail in the following Sub-Sections.

Finding Correlations between Users' Data and System Accuracy

We analyzed the obtained accuracies in combination with the user gathered data, in order to find out if the final accuracy of the system for a user depends on the user profile. This would let us know if the system performs better for a specific type of users. Figure 6.11 presents the distribution of the obtained accuracies by each criterion of the user personal information, and by the number of clusters to which it was determined that each user belongs.

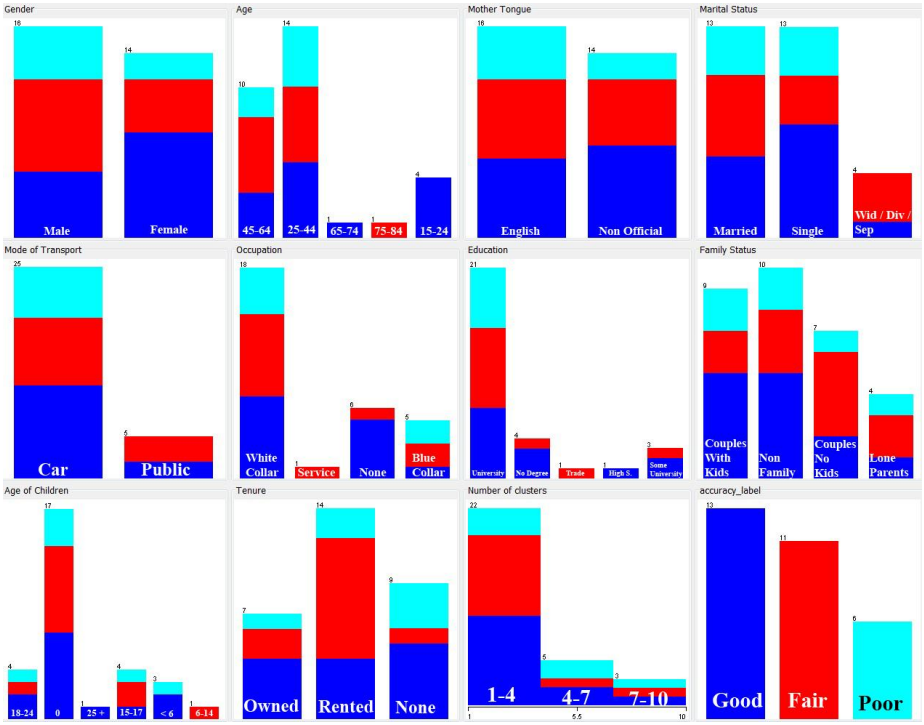


Figure 6.11: Obtained accuracy by personal information and number of clusters

In Figure 6.11 we observe that for each criterion of the personal information, as well as in the number of clusters where the user was classified, the distribution of accuracies is very uniform. The only criteria where a certain tendency is observed is in age, marital status, mode of transport, occupation and education. From the chart of age we find out that the accuracies for younger or older people the accuracy is good (i.e. represented as strong blue), or fair (i.e. represented as red), while for

the people between 25 to 64 years old, that represent the 80% of the users, the accuracies are uniformly distributed. On the other hand, from the charts of marital status, mode of transport, occupation and education, we observe that users with one or more of the following characteristics obtained only high accuracies (i.e. fair and good), namely widowed/divorced/separated users, or users that use public transport, or don't have a university degree, or don't have an occupation. However the people with these characteristics represent only the 13.33%, 16.66%, 30%, and 20% of the users in the experiment respectively, therefore these observations can't be considered as a tendency. On the other hand, in the other charts shown in Figure 6.11, we observe good, fair, and poor (i.e. represented as light blue) accuracies in most of the possible values for each criterion. Therefore we conclude that there is not a clear visual correlation between the obtained accuracy and the user personal information.

In order to find out if the accuracy can be learned from the user personal data, we applied the J48 tree classification algorithm configured to predict the accuracy class. Figure 6.12 presents the obtained classification model. In this figure we observe that the obtained model is only able to correctly classify 46.67% of the users, which is just 13.34% higher than a random guessing (i.e. 33.33%, due there are three different classes).

```
J48 pruned tree
------------------

Age = 45-64
|   Occupation = White Collar
|   |   Tenure = Owned: Good (4.0/1.0)
|   |   Tenure = Rented: Fair (2.0)
|   |   Tenure = None: Good (0.0)
|   Occupation = Service Sector: Fair (1.0)
|   Occupation = None: Fair (0.0)
|   Occupation = Blue Collar: Poor (3.0/1.0)
Age = 25-44
|   Tenure = Owned: Good (0.0)
|   Tenure = Rented
|   |   Age of Children = 18-24: Fair (0.0)
|   |   Age of Children = 0: Fair (8.0/3.0)
|   |   Age of Children = 25+: Fair (0.0)
|   |   Age of Children = 15-17: Fair (0.0)
|   |   Age of Children = <6: Good (2.0)
|   |   Age of Children = 6-14: Fair (0.0)
|   Tenure = None: Poor (4.0/1.0)
Age = 65-74: Good (1.0)
Age = 75-84: Fair (1.0)
Age = 15-24: Good (4.0)

Number of Leaves  :     17

Size of the tree :      22

Number of Leaves  :     17

Size of the tree :      22


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          14               46.6667 %
Incorrectly Classified Instances        16               53.3333 %
```

Figure 6.12: J48 Classification model considering the personal data

Subsequently, aiming to increase the goodness of the model, we ran the CFSubsetEval feature selection technique, in order to identify the features that are more relevant to predict the accuracy.

The selected features from this algorithm were age, mode of transport, occupation, education, age of children and tenure, which in some measure match with the attributes where we had already identified a partial tendency in their distribution graphs. Once these attributes were identified, we applied again the J48 classification algorithm considering only these features, however we obtained the exact same classification model as before (presented in Figure 6.12). Consequently we conclude that there is not a model that can predict the system accuracy for a user, based only on his/her personal information, with a confidence higher than 46.67%, which is too low to be considered as a good model to predict the system accuracy.

Finding Correlations between Events Ranked by Users and System Accuracy

Moreover, we analyzed whether the obtained system accuracy depends on the events presented to the user to be ranked, at the fourth part of the questionnaire. To this end, we computed for each user, considering only the ten events used to create the user preference model, the average utility and standard deviation for each one of the four criterions under they were evaluated. The obtained averages express how high the values for each criterion were, while the standard deviation states how different (i.e. distance from the average) the presented events were between them under each criterion. Additionally we computed an average standard deviation for each user considering the four standard deviations computed for each criterion. The average standard deviation expresses in general (i.e. considering all the criteria) how different the events used to create the preference model were between them. Table 6.20 presents these averages and standard deviations computed for each user. Recall from our methodology that the category utilities are obtained based on the clusters where the user belongs and the categories of the events. On the other hand, the weighted utilities depend on the user assigned weights and the options of the events presented to the user.

In Table 6.20, the blue columns show the values used to compute the average of standard deviations for each user (i.e. presented in green). Once we had computed these averages and standard deviations, we plotted in WEKA the accuracies distribution over these values. Figure 6.13 presents the distribution of the obtained system accuracies by average utility and by average standard deviation of the events used to create the user preference model for each user. In this figure we observe that as higher the average time, category and standard deviation is, the more accurate the system recommendations are. While, on the other hand, for the average weight and distance the accuracy distributions are more equally distributed over the range of obtained values.

User	Avg Category	Std Dev. Category	Avg Weight	Std Dev. Weight	Avg Distance	Std Dev. Distance	Avg Time	Std Dev. Time	Avg Std Devs.
1	55.91	21.50	4.40	18.00	9.10	26.95	13.55	1.76	13.15
2	24.84	36.50	3.75	13.00	8.90	19.12	32.41	1.06	14.31
3	40.60	67.80	5.60	67.20	9.80	35.84	35.46	1.33	28.24
4	26.49	43.00	6.50	28.70	11.20	35.04	27.41	1.29	18.64
5	29.36	3.00	5.40	63.30	11.00	41.87	4.83	1.26	19.18
6	13.13	37.20	4.65	15.50	7.70	13.83	36.25	1.80	14.33
7	36.72	42.50	6.10	21.70	9.50	25.31	33.60	1.70	16.87
8	16.76	15.40	6.00	25.90	9.70	18.07	25.62	1.22	12.45
9	24.56	25.10	4.50	16.40	8.70	13.12	18.53	1.22	9.66
10	20.00	43.60	4.65	23.40	8.90	34.96	22.84	0.85	16.19
11	35.00	31.60	5.10	18.80	9.30	47.43	36.65	1.05	22.35
12	4.83	-15.00	6.15	19.60	11.90	5.78	25.50	1.49	8.98
13	13.33	21.50	5.60	29.40	10.60	21.94	39.72	1.13	17.33
14	50.00	31.50	5.40	64.10	7.30	47.14	21.09	1.45	24.35
15	30.98	31.00	3.80	13.20	8.80	21.78	39.92	1.93	17.70
16	39.29	22.50	3.95	13.80	7.90	28.30	29.56	2.05	16.76
17	53.23	34.00	4.95	44.30	7.30	40.68	30.89	1.42	25.62
18	58.33	19.00	2.85	8.10	6.00	46.65	20.25	2.75	19.14
19	49.61	27.50	5.90	33.30	9.70	36.21	26.80	1.56	21.24
20	35.13	34.00	3.30	11.10	9.60	26.53	22.21	1.70	13.87
21	43.30	46.50	6.10	31.50	6.60	20.05	25.93	1.70	14.50
22	32.05	37.50	5.60	19.50	3.00	23.92	21.76	1.66	13.47
23	32.87	46.00	4.65	13.40	10.30	9.68	32.90	2.59	13.13
24	36.78	11.50	5.05	19.90	10.40	21.05	24.39	2.15	15.14
25	44.17	-1.00	5.30	29.50	8.50	24.99	28.46	1.18	16.45
26	42.56	30.00	5.75	22.80	10.70	26.71	8.82	1.77	11.47
27	38.27	32.00	5.25	24.80	14.30	30.75	16.87	1.06	13.76
28	45.57	12.50	6.50	73.40	11.60	26.41	27.21	1.29	20.22
29	38.82	41.50	4.35	15.30	5.50	29.39	25.50	1.31	15.34
30	32.48	49.00	5.10	27.00	6.10	26.05	34.46	1.05	17.06

Table 6.20: Average and standard deviations of the events presented to the user

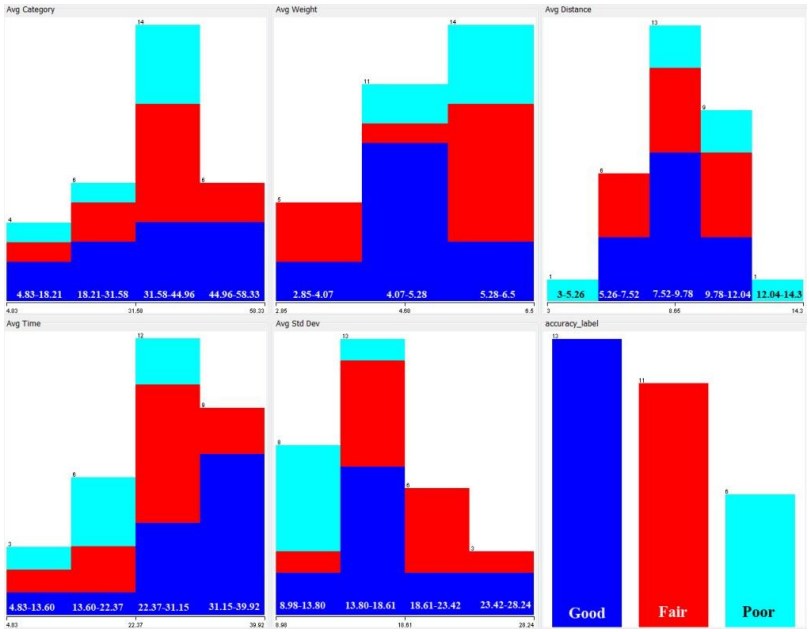


Figure 6.13: Accuracies by average utilities and average standard deviation

Similarly as we did in the accuracy analysis by user personal data, in order to determine if the observations presented above about the time, category and standard deviation are true, we applied the J48 tree classification algorithm configured to predict the accuracy class. Figure 6.14 presents the obtained classification model. In this figure we observe that, in contrast with the previously obtained model that only classified 46.67% of the users correctly, now this model is able to correctly classify 56.67% of the users. Therefore by using this model we can predict the accuracy label for a user, based on the types of events used to create his/her preference model, with a 23.34% higher confidence than a random guessing (i.e. 33.33%). However we judge that this value is still too low to consider this classification model as good enough to predict the system accuracy. Figure 6.15 depicts the tree representation of the obtained model.

```
J48 pruned tree
------------------

Avg Std Dev <= 13.757932
|   Avg Time <= 25.500545: Poor (6.0/1.0)
|   Avg Time > 25.500545: Good (2.0/1.0)
Avg Std Dev > 13.757932
|   Avg Weight <= 6
|   |   Avg Weight <= 3.3: Fair (2.0)
|   |   Avg Weight > 3.3: Good (16.0/5.0)
|   Avg Weight > 6: Fair (4.0)

Number of Leaves  :     5

Size of the tree :      9

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          17               56.6667 %
Incorrectly Classified Instances        13               43.3333 %
```

Figure 6.14: J48 Classification model considering the average utilities and standard deviation

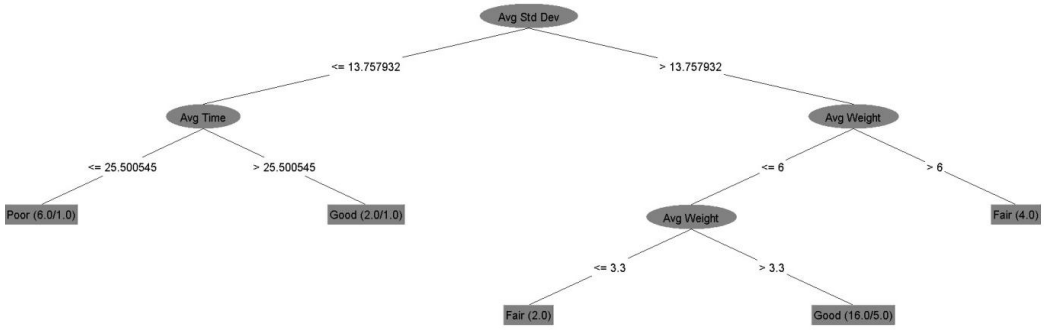


Figure 6.15: Classification tree considering the average utilities and standard deviation

Statistical Regression Analyzes

In order to verify whether or not there is a correlation between the system accuracy and the users' personal data, or the events used to create the user profile, we performed a statistical analysis to determine the correlation and dependence between this information. This analysis was done using the

statistical and graphical data analysis software tool StatGraphics® [89]. Using this software tool we obtained the regression model that best describes the system accuracy for each one of the considered criterions. Table 6.21 presents for each criterion the best regression model, along with its correlation and determination coefficients, and the obtained equation to predict the system accuracy. This table also indicates whether there is a significant statistical correspondence between the criterion and the obtained accuracy. Recall that in a regression model the correlation coefficient (i.e. represented by the variable R) represents how much the dependent variable (i.e. the accuracy) depends on the independent variable (i.e. the considered criterion). This correlation coefficient is expressed in a range from {-1,…,0,…,1}, where a value of 0 expresses that there is no relation between the two variables, and values of -1 and 1 express that the dependent variable completely depends on the independent one. The determination coefficient, on the other hand, is obtained by squaring the correlation coefficient, and it is presented as a percentage that expresses how much the dependent variable depends on the independent one. It is important to note that in order to create a regression model, the accuracy used has to be entered as a numerical value (i.e. the percentage as they were obtained), not categorical (i.e. grouped in classes). For more information on the used regression models, and the procedure to determine whether or not a criterion is statistically significant to predict the accuracy, we recommend [58].

Criterion	Model Name	Equation to predict the accuracy	Correlation	Determination	Confidence	Significant Statistical Correspondence
Avg Desv	Double Inverse	Accuracy = 1/(0.00943831 + 0.0533795/Avg Desv)	0.525292	27.59%	0.95	YES
Age	Inverse-X	Accuracy = 68.4911 + 21.6418/Age	0.452341	20.46%	0.95	YES
Avg Time	Inverse-Y	Accuracy = 1/(0.0152606 - 0.0000912704*Avg Tim)	-0.441891	19.53%	0.95	YES
Education	Inverse-X	Accuracy = 90.0265 - 13.6549/Education	-0.418356	17.50%	0.95	YES
Occupation	Lineal	Accuracy = 72.953 + 3.79757*Occupation	0.304068	9.25%	0.9	NO
Gender	Lineal	Accuracy = 82.1921 - 5.69339*Gender	-0.272042	7.40%	0.9	NO
Clusters	Double Inverse	Accuracy = 1/(0.0134989 - 0.00168574/Clusters)	-0.258607	6.69%	0.9	NO
Marital Status	Inverse-X	Accuracy = 73.2304 + 8.53235/Marital Status	0.222646	4.96%	0.9	NO
Mode of Transport	Inverse-Y	Accuracy = 1/(0.0120737 + 0.000942469*Mode of Transport)	0.203638	4.15%	0.9	NO
Avg Weight	Lineal	Accuracy = 90.9107 - 2.31703*Avg Wei	-0.20278	4.11%	0.9	NO
Avg Cat	Inverse-X	Accuracy = 80.8536 - 44.535/Avg Cat	-0.147742	2.18%	0.9	NO
Avg Distance	Double Inverse	Accuracy = 1/(0.012281 + 0.00477097/Avg Dis)	0.129847	1.69%	0.9	NO
Mother Tongue	Inverse-Y	Accuracy = 1/(0.0126503 + 0.000391479*Mother Tongue)	0.113232	1.28%	0.9	NO
Age of Children	Logarithmic-X	Accuracy = 79.9391 - 1.38642*ln(Age of Children)	-0.0917652	0.84%	0.9	NO
Family	Lineal	Accuracy = 77.5345 + 0.648462*Family Status	0.0764875	0.59%	0.9	NO
Tenure	Lineal	Accuracy = 76.9233 + 1.08017*Tenure	0.0752376	0.57%	0.9	NO

Table 6.21: Best statistical regression models for each criterion

The statistical analysis shows that the only criteria, on which the accuracy depends with a significant statistical correspondence, are the average standard deviation between the selected events to create the user preference model, the user age, the average time of the events, and the user education level. This results match with the attributes previously selected and used within the two classification models we had obtained, as presented in Figures 6.12 and 6.14. Figure 6.16 shows the graphs for the

obtained regression models for the average standard deviation, age, average time and education. In these charts, the blue line represents the prediction curve corresponding to the equation shown in Table 6.21. The red line represents the accuracy interval with a 95% confidence that an average value of the criterion would obtain. Finally the purple line shows the accuracy interval with a 95% confidence that a single value of the criterion would get.

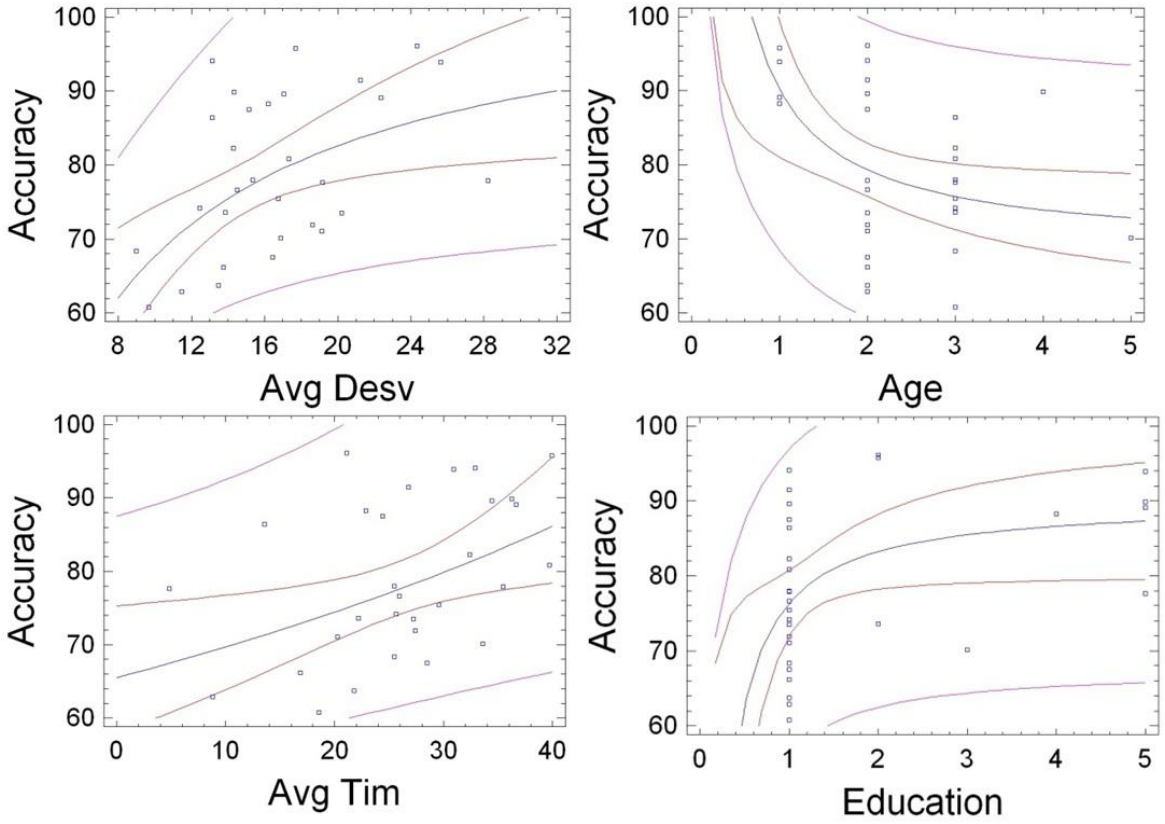


Figure 6.16: Regression model graphs for significant statistical criterions

From the previous statistical analysis and the two obtained classification models, we conclude the following. The accuracy obtained by the system seems to be influenced by two factors. They are, firstly, how long the user has to travel to reach an event and, secondly, how diverse the types of events are. Consequently, when creating the users' preference models, in order to increase the system accuracy, we should select events with large times from the user (i.e. that it will take a user long time to get there), and that are as different as possible from each other. That is, users will travel far for events that they really wish to attend. Further, the diversity of the events encourages users to strongly discriminate between them. This allows the system to easily identify the user preferences and create more accurate users' models that lead to generate better recommendations.

On the other hand, since the determination coefficients for the age and education of the users at the regression models are small, and considering that in our experiment we don't have an equal distribution of users in those two attributes, we conclude that the system accuracy doesn't depend on the user personal data. From this conclusion, it follows that the system produces equally accurate recommendation lists (i.e. average accuracy of 79.16%) for all the user types considered within our experiment.

Based on the obtained percentages of correctly classified users from the classification models, and determination coefficients from the regression models, we conclude from our offline experiment that the system accuracy mainly depends on factors other than the user profile or selected events when creating the user preference model. These factors may be how congruent with his/her preferences the user was when ranking the presented events, how difficult for the user was to discriminate between 20 events when ranking them, or how easily for the user was to answer the questions in the questionnaire using the proposed scales. Correspondingly we propose, as a future work, to run a user study or an online experiment, as presented in Section 3.4.1, to gather more instances and qualitative data from the users. Subsequently this data and qualitative metrics should be analyze in order to find if a classification algorithm is able to identify a model that correctly classifies a higher percentage of the users, or if there is a relation between the accuracy and a qualitative measure that cannot be evaluated using offline experiments.

6.3.5 System Coverage

In the final analysis we measured both, the item and user space coverage's. To evaluate the user space coverage, in accordance to the proposed metric for this property previously presented in Section 3.4.2, we computed the percentage of users from the offline experiment that obtained a prediction accuracy higher than a specific threshold. For this experiment, we decided to set this threshold to 70%, which is the lower limit of the previously defined "Fair" accuracy class. Therefore, by setting this threshold to 70% we are measuring the percentage of users from the offline experiment that obtained an accuracy "Fair" or "Good". Since there are 24 out of 30 users, with a prediction accuracy higher than the selected threshold, the user space coverage of the system in the offline experiment was of 80%. This means that for 80% of the users in our experiment the system was capable to identify their preferences, and create models that can be used to rank events with at least a 70% match with the way the users would rank them.

Finally, the item space coverage was measured using the previously proposed metric for this property in Section 3.4.2. Hence it was computed as how many items, from all the items included in the produced recommendation lists, were recommended to a user at least once, with a utility higher than a selected threshold. Recall that the produced items' utilities from our RS are within a range of *{0,...,1}*. Within this range, a value of 0 means that event doesn't have any features (i.e. categories, options, distance, or time) that match with the user preferences, while on the other hand, a value of 1 means that the event has all the features that the user prefers in an event. For this experiment, we decided to set a threshold of 0.5 for this measure. With this threshold we are measuring how many items from the recommended lists at least matched the half of a user given preferences. The RS, as implemented in PeRS, considered a total of 1'503 events to be recommended in the offline experiment. From these, 1'232 events matched at least the half of the preferences given by a user. Therefore we conclude that in the offline experiment the system presented an 81.97% of item space coverage. This means that 81.97% of the items in the database were recommended to a user at least once, with at least a 50% match of his/her preferences.

The user and item space coverage metrics, express the extent of users to whom the system produced highly accurate recommendations, and the extent of items that the system recommended with a specific match to their preferences, respectively. However these metrics, even though they express by their own the system coverage on users and items, are of special use when comparing two or more RSs over the same users and items datasets.

6.3.6 Discussion of Results

Table 6.22 presents a summary of the evaluation results for our RS obtained from the performed offline experiment. From these results we concluded that, based on their demographic information, 93% of the users considered in our offline experiment have differences in their profiles. This allowed us to test the system for a wide spread distribution of users, and therefore potentially draw more reliable conclusions for the general population. Additionally, we concluded that the system was able to narrow down the possible matching clusters where a user may belong (from the 66 demographic groups included in the database), using only the selected nine demographic variables asked to the users. Furthermore, the users identified themselves within 86% of the system selected clusters, based on the characteristic activities of the people within each group, which means that the system accurately classified the users into the demographic groups where they belong to.

Following the methodologies presented in Section 3.4.2, to measure some system properties, we determined that the system has a prediction accuracy between 75% and 82% with a confidence of 95%. This means that the system is able to identify the preferences for 95% of the general

population, and generate recommendations that match them with an average of 79%. Moreover for 80% of the users considered in our experiment the system was able to rank the items with at least 70% match with the way they would have done it (i.e. user coverage). Finally, the system recommended 81% of the items in the database to at least one user with 50% match of his/her preferences (i.e. item coverage).

From the performed data mining analyzes, we concluded that the accuracy of the produced recommendations does not significantly depend neither on the user profile (i.e. who the user is), nor the events considered to create the user preference model. This conclusion follows from the inability to create models, with high percentages of correctly classified instances, to predict the system accuracy from the user data or the events asked to the users to rank. However, based on the determination coefficients obtained from the statistical analyzes we determined that the accuracy could be improved by asking the users to rank events as different as possible between them, and that would require that the user travels a long time to get to them.

Property	Measured Value	Interpretation
Users Distribution	93.33% of the users are different between them.	The users present an equal distribution in most of the considered personal variables. Therefore this wide spread distribution of users allows the system to be tested for different user types and therefore draw more reliable conclusions.
Narrowing Percentage	95.38%	The system is able to narrow in a 95.38% the possible demographic groups where a user may belong, during the demographic generalization process.
Accurate matching clusters	86.66%	86.66% of the users identified themselves with the activities included in the selected demographic groups by the system.
Prediction Accuracy and Confidence	**Accuracy:** {75.36 % – 82.96%} **Confidence:** 95%	The system would recommend items that match the user preferences with an average of 79.16%, for 95% of the general population
Classification Model for Personal Data	46.67% Correctly classified instances.	The obtained model allows us to predict the system accuracy class from the user personal data with just 13.34% higher accuracy than a random guessing.
Classification Model for Events' Utilities	56.67% Correctly classified instances	The obtained model allows us to predict the system accuracy class from the type of events, used to create the user preference model, with 23.34% higher accuracy than a random guessing.
Statistic Models	**Avg Std Desv.:** 27.59% **Age:** 20.46% **Avg Time:** 19.53% **Education:** 17.5%	Based on the determination coefficients of the regression models and the users distributions, we conclude that the system accuracy doesn't depend on the user profile. However it presents a small dependence to the time to the events, and to how different the events used to create the user preference model are between them.
User Space Coverage	80%	For 80% of the users the system created models that can rank the events with at least a 70% match with the way the users would rank them.
Item Space Coverage	81.97%	81.97% of the items in the database were recommended to a user at least once, with at least a 50% match of his/her preferences.

Table 6.22: Summary of evaluation results

6.4 Summary

Within this Chapter we presented the experimental evaluation of our proposed methodology for RSs, as presented in Chapter 5. In Section 6.1, we describe our case study, a Personal Recommender System (PeRS), that implements our described methodology for RSs to recommend events to a user. In this Section we start by describing the hardware and software environment set up and our selected datasets, providing information about their attributes, data contained and formats. PRIZM C2 was selected as our large scale demographic database, and Ottawa Open Data foundation as our source for events and spatial datasets. Afterwards, we conclude this Section presenting a stage by stage execution of the PeRS to recommend events to a user. At each stage of the execution, as a running experiment, we highlight the required user interactions, present the data introduced by a real user, and show the system results.

In Section 6.2 we present the design of the performed offline experiment to evaluate our methodology for RSs, as implemented in our case study (PeRS). The offline experiment was executed, with the goal of evaluate the RS under its prediction accuracy, coverage and confidence properties. These properties were selected, based on the specific application domain, the proposed recommender engine and the desired scope of the evaluation. Additionally, we present the structure of the designed questionnaire to be asked to the users, in order to obtain the information required to implement the offline experiment, and aiming to represent as closely as possible how the system would interact with the users.

Finally we conclude this Chapter presenting, in Section 6.3, the results obtained from the performed offline experiment. We provide a detailed analysis of the obtained results, aiming to achieve the following goals: typify the users considered in our experiment, evaluate the system accuracy and its ability to cluster the user in the most adequate demographic groups, learn the system accuracy from the users' information or events shown to them, and evaluate the system coverage.

From the results obtained from the offline experiment and the performed analyzes, we concluded that our system was able to identify the matching demographic clusters where a user may belong narrowing the possible clusters in 95% and with an accuracy of 86%. Furthermore, the system is able to identify, for 95% of the general population, their preferences and generate recommendations that match them with an average of 79% (i.e. prediction accuracy). Moreover for 80% of the users considered in our experiment the system was able to rank the items with at least 70% match with the way they would have done it (i.e. user space coverage). Finally, the system recommended 81% of the items in the database to at least one user with 50% match of his/her preferences (i.e. item space coverage). Furthermore, we concluded that the accuracy of the produced

recommendations does not significantly depend neither on the user profile (i.e. who the user is), nor the events considered to create the user preference model. However it could be improved by asking the users to rank events as different as possible between them, and that would require that the user travels a long time to get there.

Chapter 7
Conclusion

RSs are software systems designed to understand and model the user, considering his/her context, preferences and constraints, with the aim to provide him/her personalized recommendations of items. RSs exploit the large amount of information in the databases that the increasing use of World Wide Web has made available, and the critical role that web-based systems play in our lives to aid users through different decision-making processes. In order for a RS to generate a personalized recommendation, it first has to understand and model the user and then select out of thousands of items the ones that best match his/her preferences. Therefore, RSs may be seen as data mining user front ends that present to the user in an easy and personalized way, the items resulting from applying several data mining and statistical techniques, with the goal of finding the most suitable items for each user.

RSs remove from the user the difficult, time-consuming, and error-prone task of manually analyzing every possible item and consider a large number of features and criteria in order to make a decision, ensuring to the user that the recommended items are the most suitable options for him/her. Consequently, RSs are of special use in domains where the user has a large number of options to select from, or where the rapidly changing context plays a critical role in the decision, making impossible for a human to consider every option at every time. Moreover, companies are now using RSs to point their customers towards not-yet experienced items that might be relevant for them, providing the users with a valuable personalized service that makes them feel unique and comfortable. However, RSs constantly face the following challenges: find better techniques to understand and model the users, generate recommendations considering more information during process, speed the process of analyzing thousands of items, decide the most suitable trade-off

between recommending novel or similar items to the use, or between provide quick recommendations or ensure the user that all items were analyzed, etc. Consequently, several techniques and implementations have been proposed.

In this thesis we propose a methodology, which follows the general structure for RSs, to create a **Personalized-Multi-Criteria-Context-Aware-Hybrid-Recommender-System**. This methodology creates accurate multi-criteria user preference models, considering the users' profiles and locations, in order to recommend items to them. The user profile includes information about who the user is, generalizing his/her data from a demographic database, and about his/her specific preferences. The user location, on the other hand, includes information about where the user is, and about his/her mobility level. The proposed process to recommend an item, as identified in the methodology, has the following phases. First, create the user profile and obtain the user's location and mobility level. Subsequently, assess the utilities for each item, based on the user profile, user location, specific preferences, and mobility level. Once the utilities for each item have been computed, create a user preference model, considering each utility as a criterion. Finally, using the user preference model, obtain an integrated utility for each item, in order to recommend the items with the highest integrated utilities.

7.1 Thesis Contributions

In the effort to better understand the user and provide him/her tailored services and suggestions, a number of Recommender Systems have been proposed and implemented in the research communities.

The main contribution of this thesis is that it presents a methodology for RSs, that proposes the creation of user models that consider the user context and preferences to recommend the items. Our proposed methodology, as concluded from the performed offline experiment, is capable to accurately narrow and identify the groups from a demographic database where a user may belong. Furthermore, it produces highly accurate recommendation lists of items that match the user preferences with an average of 79.16%, for 95% of the population.

The proposed methodology, presents the following noteworthy characteristics. **It uses more than one decision criteria**. In contrast with some proposed RSs, that consider single-criterion ratings, our system considers multi criteria to create user preference models. Consequently it is able to identify those characteristics of the items that the user considers most important when selecting an item over the others. This allows the system to trade-off between the item's properties to finally recommend items that, as a total, present the highest utility for a user. In this sense, the user can rely

that the items being recommended are the best options for him/her, considering the current context and items' characteristics. **It creates user preference models**. The final recommendations are obtained by assessing each item against the created user preference model, instead of just use discovered patterns in the user previous behaviour. Therefore the system is capable to understand the user interests and preferences, and not be biased from previous user actions that may not necessarily represent what the user likes. **It generalizes the user profile from a large-scale demographic database**. This method encompasses the benefits of content-based and collaborative-based RSs, overriding each other's disadvantages. Since the user profile is created by classifying the user into one or more demographic groups, the system is capable to reason from the commonalities between them, on the contrary with content-based RSs. Furthermore, in contrast with collaborative-based RSs, it uses a demographic database to generalize the user profile, and therefore it doesn't require information from a large number of users in order to be applied. Additionally, by using this method, the obtained user profiles contain all the richness included in the demographic database, allowing the system to generate recommendations of areas not directly asked to the users. **It is location aware**. The system considers the user current location and mobility level as two criteria when creating the user preference model. Therefore it can recommend different items to the same user, while he/she travels. This allows to the users in the system to have different profiles, depending on the context where they are at each moment, and obtain recommendations according to it. **It can be applied to different application domains**. The proposed methodology is based on the creation of multi criteria user preference models to recommend the items. Hence, by adding or removing the considered criterions, depending on the available information, it can be used to create RSs to be applied within different application domains.

Additionally, this thesis also presents the following contributions:

A general framework for RSs. It identifies and presents in detail a general framework presented in all types of personalized RSs. This framework consists of user, user profile, item profile, and recommender engine as main components, and identifies the following algorithm to recommend items to the users. First, obtain the user information to create his/her profile. Subsequently, depending on the available information, apply the most suitable recommendation algorithm to assess the utility for each item. Finally recommend the items with the highest utility.

A comparison of types of RSs. It presents a detailed comparison between the three existing types of RSs, content-based, collaborative-based and Hybrid systems, based on their limitations, strengths, and recommendation engines used. From this comparison we conclude that, while content-based techniques are more suitable for applications with a large number of information for each item,

collaborative-based RSs are more appropriate for applications where the user profiles contain a large number of data. Collaborative-based methods are further classified between model-based and memory-based methods. The first, are preferred for applications where a large number of item's utilities are going to be assessed, and the rate of new added users is low. The latter, on the other hand, are preferred for applications where the number of items to estimate the utility is small, or the rate of new added users is high.

A comparison of previously proposed RSs. This thesis presents a detailed comparison of some previously RSs implementations, by means of their proposed methodology, advantages and disadvantages. The RS implementations discussed in this thesis are the following. A multi-criteria RS that creates user preference models to group similar users, and then applies a traditional collaborative algorithm to recommend items to the users in each group. A RS that generalizes the user profile and recommends to each user the items included in the demographic database for the group where the user was classified. A system that creates a hierarchical classification model to initially categorize the users based on their context and subsequently on their preferences, to finally recommend them the preferences of the users in the same group. Finally a system that automatically changes a web site structure, based on user patterns continuously learned from user web logs.

7.3 Future Work

Although the methodology for RSs provided in this thesis presented several advantages and achieved high prediction accuracy levels, as presented in Section 6.2, it still may be furthered in several directions. Some suggestions for future work include the following:

Perform user studies and online experiments to discover patterns or correlations. As previously stated in Section 6.2, an interesting future work would be to execute a user study or an online experiment, as presented in Section 3.4.2. It would be worthwhile to analyze with more user results and qualitative data if there exists a significant correlation between the accuracy of the produced recommendations and the user personal information, system interface, or number of presented events. These correlations, if they exist, would allow us to modify how the system interacts with the users and/or to target a specific population segment for which the system recommends items with the highest accuracy levels.

Find the most appropriate number of events to be presented to the user. It would be important to analyze the correlation between the accuracy increasing rate, and the number of instances (i.e. events) used by the UTA* algorithm to create the user preference models. On the other hand, measure the accuracy decreasing rate due the number of events presented to the user to be

rated, as analyzed by the Human Computer Interaction (HCI) field of study. These analyses would allow us to select the number events used to create the user preference models, obtaining the best trade-off between instances used by the UTA* algorithm and events asked to the user to be rated.

Expand the UTA* algorithm capabilities. Another possible direction would be to extend the UTA* algorithm capabilities, so it would be able to handle an imbalanced dataset of user preferences and still produce accurate preference models. This would imply that the algorithm should be able to detect the user preferences between items even when they may be very similar in all or some criterions. Furthermore, it would also be interesting to use other kind of UTA algorithms, such as Meta-UTA techniques, Stochastic UTA methods, or UTA-type sorting methods, to create the user models, and measure if the recommendation accuracies increase. For more information of these UTA algorithms and techniques, we recommend [38].

Use feature selection techniques. As presented in Section 6.1.2, the information from the demographic database, selected as the most representative to classify a user within one or more demographic clusters, was selected by inspection based on domain knowledge. However, it would be interesting to apply a Feature Selection technique over the demographic database to compare if the resulting variables match with the ones select in our case study. Moreover, in case they would be different, analyze if the system narrowing percentage, the cluster matching accuracy, or the final recommendations accuracy changes by asking the user the resulting information instead, during the demographic generalization phase.

Apply the proposed methodology to different application domains. The proposed system, as applied in the presented case study, has only being used and tested within a single application domain (i.e. events recommender). As a future work, it may be interesting to apply this methodology to different application domains, and measure its prediction accuracy, in order to determine how much the types of items being recommended affect the performance of the system.

Produce recommendations for more than one user at a time. In this work we aimed to produce accurate recommendations for a single user. As a future work, it would be worthwhile to extend the presented system to recommend items of interest to a group of users, considering the preference models and locations of all of them at the same time. More precisely, estimate joint item's utilities for a group of users, aiming to please as much as possible all the different user preferences and contexts. One possible approach to achieve this integration of users' preferences would be to apply again the UTA* algorithm in a second iteration, now using to order them (i.e. as a weak preference order) the average of the integrated utilities computed for each item using the preference models created for each user. With this approach we would be considering every user with the same weight and each one's preference and context weights.

Couple the presented RS as part of a bigger integrated intelligent system. Probably the most useful and interesting direction for future work, is how to integrate our RS into other existing intelligent systems. This is that the RS may be seeing as a black box that receives some structured input information and outputs the recommended items in a structured manner as well. This integration would require accomplishing the following two tasks. First, identify and define the structure and the type of data gathered by the user interactions, in order to look for alternative sources (or systems) from where we could retrieve this information. Subsequently, the second task would deal with formatting the current system output (i.e. an ordered ranked list of recommended items), into an already known form such as OLAP cubes data models, as presented in Section 3.2.1. This would allow that other systems and OLAP tools may access, display and analyze the produced lists of recommendations.

References

[1] J. Han and M. Kamber, Data Mining: Concepts and Techniques., second edition., Morgan Kaufmann., 2006.

[2] F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, Recommender Systems Handbook., Springer, 2011.

[3] T. Mahmood and F. Ricci, Improving recommender systems with adaptive conversational strategies., in *HT '09 Proceedings of the 20th ACM conference on Hypertext and hypermedia.*, New York, NY, USA., 2009.

[4] P. Resnick and H. Varian, Recommender systems., in *Communications of the ACM.,* vol. 40, no. 3, pp. 56-58, 1997.

[5] R. Burke, Hybrid web recommender systems., in *The AdaptiveWeb.*, pp. 377-408, Heidelberg., Springer., Berlin, 2007.

[6] F. McSherry and I. Mironov, Differentially private recommender systems: building privacy into the net., in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining.*, New York, NY, USA, 2009.

[7] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions., in *IEEE Transactions on Knowledge and Data Engineering.,* vol. 17, no. 6, pp. 734-749, 2005.

[8] J. Schafer, D. Frankowski, J. Herlocker and S. Sen, Collaborative filtering recommender systems., in *The Adaptive Web: Methods and Strategies of Web Personalization.*, pp. 291-324, Heidelberg., Springer., Berlin, 2007.

[9] S. Schiaffino and A. Amandi, Intelligent User Profiling., in *Artifitial Intelligence.*, pp. 193-216, Heidelberg., Springer-Verlag., Berlin, 2009.

[10] W. Paireekreng and K. W. Wong, Client-side Mobile User Profile for Content Management Using Data Mining Techniques., in *Eight International Symposium on Natural Language Processing.*, Bangkok, Thailand., 2009.

[11] K. Lakiotaki, N. F. Matsatsinis and A. Tsoukias, Multi-Criteria User Modeling in Recommender Systems., in *Intelligent Systems,* vol. 26, no. 2, pp. 64-76, *IEEE.,* 2011.

[12] D. Billsus and M. Pazzani, Learning probabilistic user models, in *UM97 Workshop on Machine Learning for User Modeling*, Chia Laguna, Sardinia., 1997.

[13] G. Fisher, User modeling in human-computer interaction., in *User Modeling and User-Adapted Interaction.,* vol. 11, no. 1-2, pp. 65-86, 2001.

[14] S. Berkovsky, T. Kuflik and F. Ricci, Mediation of user models for enhanced personalization in recommender systems., in *User Modeling and User-Adapted Interaction.,* vol. 18, no. 3, pp. 245-286, 2008.

[15] S. Berkovsky, T. Kuflik and F. Ricci, Cross-representation mediation of user models., in *User Modeling and User-Adapted Interaction.,* vol. 19, no. 1-2, pp. 35-63, 2009.

[16] D. Xu, H. Wang and K. Su, Intelligent Student Profiling with Fuzzy Models., in *HICSS '02 Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)*, Washington, DC, USA., 2002.

[17] A. Arya, N. Jefferies, J. Enns and S. DiPaola, Facial actions as visual cues for personality., in *Computer Animation and Virtual Worlds.,* vol. 17, no. 3-4, pp. 371-382, 2006.

[18] M. Balabanovic and Y. Shoham, Content-based, collaborative recommendation., in *Communications of the ACM.,* vol. 40, no. 3, pp. 66-72, 1997.

[19] G. Salton, Automatic Text Processing: the transformation, analysis, and retrieval of information by computer., Addison-Wesley, Boston, MA, USA., 1989.

[20] J. J. Rocchio, Relevance Feedback in Information Retrieval., in *SMART Retrieval System-Experiments in Automatic Document Processing.*, pp. 313-323, Prentice Hall., 1971.

[21] M. Pazzani and D. Billsus, Learning and revising user profiles: The identification of interesting web sites., in *Machine Learning - Special issue on multistrategy learning.,* vol. 27, no. 3, pp. 313-331, 1997.

[22] D. Muhivuwomunda, Data De-Duplication through Active Learning, University of Ottawa, Master Thesis, Ottawa, ON, Canada, 2010.

[23] M. Y. Bilenko, Learnable Similarity Functions and Their Applications to Record Linkage and Clustering., The University of Texas at Austin. PhD thesis., Austin, TX, USA., 2006.

[24] E. F. Krause, Taxicab Geometry: An Adventure in Non-Euclidean Geometry., Dover Plublications., New York, NY, USA., 1986.

[25] R. Baeza-Yates and B. Ribeiro-Neto, Modern Information Retrieval., Addison-Wesley., Harlow, England., 1999.

[26] U. Shardanand and P. Maes, Social information filtering: Algorithms for automating word of

mouth., in *Proceedings of the SIGCHI conference on Human factors in computing systems.*, 1995.

[27] J. S. Breese, D. Heckerman and C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in *UAI'98 Proceedings of the Fourteenth conference on Uncertainty in artificial Intelligence*, San Francisco, CA, USA., 1998.

[28] J. Delgado and N. Ishii, Memory-based weighted-majority prediction for recommender systems., in *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation.*, Berkeley, CA, USA., 1999.

[29] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, GroupLens: An open architecture for collaborative filtering of netnews., in *CSCW '94 Proceedings of the 1994 ACM conference on Computer supported cooperative work.*, New York, NY, USA., 1994.

[30] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, Item-based Collaborative Filtering Recommendation Algorithms., in *WWW '01 Proceedings of the 10th international conference on World Wide Web.*, New York, NY, USA., 2001.

[31] D. Billsus and M. Pazzani, Learning collaborative information filters, in *ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning.*, 1998.

[32] W. Lee, Collaborative Learning and Recommender Systems., in *Proccedings of the 18th International Conference on Machine Learning.*, San Francisco, CA, USA., 2001.

[33] A. Ansari, S. Essegaier and R. Kohli, Internet recommendations systems., in *Journal of Marketing Research.,* vol. 37, no. 3, pp. 363-375, 2000.

[34] C. Andrieu, N. de Freitas, A. Doucet and M. I. Jordan, An Introduction to MCMC for Machine Learning., in *Machine Learning.,* vol. 50, no. 1, pp. 5-43, 2003.

[35] B. A. Berg., Markov Chain Monte Carlo Simulations and Their Statistical Analysis., World Scientific., Singapore., 2004.

[36] M. Köksalan, J. Wallenius and S. Zionts, Multiple Criteria Decision Making: From Early History to the 21st Century., World Scientific., Singapore., 2011.

[37] P. Yu and M. Zeleny, The Set of All Non-Dominated Solutions in Linear Cases and a Multicriteria Simplex Method., in *Journal of Mathematical Analysis and Applications.,* vol. 49, no. 2, pp. 430-468, 1975.

[38] Y. Siskos, E. Grigoroudis and N. Matsatsinis, UTA Methods., in *Multiple criteria decision analysis: State of the art surveys.*, pp. 297-344, Springer., New York, NY, USA., 2005.

[39] E. Jacquet-Lagrèze and Y. Siskos, Preference disaggregation: 20 years of MCDA experience., in *European Journal of Operational Research.,* vol. 130, no. 2, pp. 233-245, 2001.

[40] B. Roy, Multicriteria Methodology for Decision Aiding., Economica., Paris, France., 1985.

[41] E. Jacquet-Lagrèze and J. Siskos, Assessing a set of additive utility functions for multicriteria decision making: The UTA method., in *European Journal of Operational Research.,* vol. 10, no. 2, pp. 151-164, 1982.

[42] Y. Siskos and D. Yannacopoulos, UTASTAR: An ordinal regression method for building additive value functions., in *Investigação Operational.,* vol. 5, no. 1, pp. 39-53, 1985.

[43] J. Lee, R. M. A. Mateo, B. D. Gerardo and S.-H. Go, Location-Aware Agent Using Data Mining for the Distributed Location-Based Services., in *Lecture Notes in Computer Science.,* vol. 3984, no. 2006, pp. 867-876, 2006.

[44] B. Schilit and M. Theimer, Disseminating active map information to mobile hosts., in *IEEE network.,* vol. 8, no. 5, pp. 22-32, 1994.

[45] R. Xu and D. I. Wunsch, Survey of clustering algorithms., in *IEEE Transactions on Neural Networks.,* vol. 16, no. 3, pp. 645-678, 2005.

[46] A. K. Jain, M. N. Murty and P. J. Flynn, Data clustering: A review., in *ACM Computing Surveys.,* vol. 31, no. 3, pp. 264-323, 1999.

[47] D. Koller and M. Sahami, Toward optimal feature selection., in *Proceedings of the 13th International Conference on Machine Learning.*, 1996.

[48] H. Liu and H. Motoda, Feature selection for knowledge discovery and data mining., Kluwer Academic Publishers., Norwell, MA, USA., 1998.

[49] S. Chatterjee, A. Hadi and B. Price, Regression analysis by example., John Wiley and Sons., 2000.

[50] G. Adomavicius, R. Sankaranarayanan, S. Sen and A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach., in *ACM Transactions on Information Systems (TOIS).,* vol. 23, no. 1, pp. 103-145, 2005.

[51] G. Adomavicius and A. Tuzhilin, Incorporating context into recommender systems using multidimensional rating estimation methods., in *Proceedings of the 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPR-SIUI 2005).*, 2005.

[52] B. Krulwich, Lifestyle Finder. Intelligent User Profiling Using Large-Scale Demographic Data., in *AI Magazine.,* vol. 18, no. 2, pp. 37-45, 1997.

[53] Y. Hijikata, T. Shimizu and S. Nishida, Discovery-oriented collaborative filtering for improving user satisfaction., in *IUI '09 Proceedings of the 14th international conference on Intelligent user interfaces.*, New York, NY, USA., 2009.

[54] C. J. Van Rijsbergen, Information Retrieval, Butterworth-Heinemann., Newton, MA, USA., 1979.

[55] Y. Yao, Measuring retrieval effectiveness based on user preference of documents., in *Journal of the American Society for Information Science.,* vol. 46, no. 2, pp. 133-145, 1995.

[56] D. Fleder and K. Hosanagar, Recommender systems and their impact on sales diversity., in *EC '07: Proceedings of the 8th ACM conference on Electronic commerce.*, New York, NY, USA., 2007.

[57] J. Herlocker, J. Konstan and J. Riedl, Explaining collaborative filtering recommendations., in *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work.*, New York, NY, USA., 2000.

[58] M. Berenson and D. Levine, Basic Business Statistics: Concepts and Applications., Prentice Hall., 1995.

[59] D. Barron's Accounting, Normal Distribution., [Online]. Available: http://www.answers.com/topic/normal-distribution. [Accessed 5 oct 2012].

[60] H. Cramer, V. Evers, S. Ramlal, M. Someren, L. Rutledge, N. Stash, A. L. and B. Wielinga, The effects of transparency on trust in and acceptance of a content-based art recommender., in *User Modeling and User-Adapted Interaction.,* vol. 18, no. 5, pp. 455-496, 2008.

[61] P. Pu and L. Chen, Trust building with explanation interfaces., in *IUI'06 Proceedings of the 11th international conference on Intelligent user interfaces.*, New York, NY, USA., 2006.

[62] Y. Zhang, J. Callan and T. Minka, Novelty and redundancy detection in adaptive filtering., in *SIGIR '02 Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.*, New York, NY, USA., 2002.

[63] O. Celma and P. Herrera, A new approach to evaluating novel recommendations., in *2008 ACM Conference on Recommender Systems.*, New York, NY, USA., 2008.

[64] N. Jones and P. Pu, User Technology Adoption Issues in Recommender Systems., in *Networking and Electronic Commerce Research Conference.*, Riva del Garda, Italy., 2007.

[65] G. Shani, M. Chickering and C. Meek, Mining recommendations from the web., in *RecSys '08 Proceedings of the 2008 ACM Conference on Recommender Systems.*, New York, NY, USA., 2008.

[66] M. Zhang and N. Hurley, Avoiding monotony: improving the diversity of recommendation lists., in *RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems.*, New York, NY, USA., 2008.

[67] K. Järvelin and J. Kekäläinen, Cumulated gain-based evaluation of ir techniques., in *ACM Transactions on Information Systems (TOIS).,* vol. 20, no. 4, p. 422–446, 2002.

[68] S. McNee, J. Riedl and J. Konstan, Making recommendations better: an analytic model for human-recommender interaction., in *CHI EA '06 CHI '06 extended abstracts on Human factors in computing systems.*, New York, NY, USA., 2006.

[69] S. Lam and J. Riedl, Shilling recommender systems for fun and profit., in *WWW'04 Proceedings of the 13th international conference on World Wide Web.*, New York, NY, USA., 2004.

[70] P. Chirita, W. Nejdl and C. Zamfir, Preventing shilling attacks in online recommender systems., in *WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management.*, New York, NY, USA., 2005.

[71] D. Frankowski, D. Cosley, S. Sen, L. Terveen and J. Riedl, You are what you say: privacy risks of public mentions., in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.*, New York, NY, USA., 2006.

[72] T. Mahmood and F. Ricci, Learning and adaptivity in interactive recommender systems., in *ICEC'07 Proceedings of the ninth international conference on Electronic commerce.*, New York, NY, USA., 2007.

[73] G. Karypis, Evaluation of Item-Based Top-N Recommendation Algorithms., in *CIKM'01 Proceedings of the tenth international conference on Information and knowledge management.*, New York, NY, USA., 2001.

[74] J. Herlocker, J. Konstan and J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms., in *Information Retrieval.,* vol. 5, no. 4, pp. 287-310, 2002.

[75] T. Robal and A. Kalja, Applying User Profile Ontology for Mining Web Site Adaptation Recommendations., in *ADBIS Research Communications.*, pp. 126-135, Technical University of Varna, 2007.

[76] A. Mikroyannidis and B. Theodoulidis, Web usage Driven Adaptation of the Semantic Web., in

End User Aspects of the Semantic Web Workshop, 2nd European Semantic Web Conference (ESWC 2005)., Heraklion, Greece., 2005.

[77] T. Robal and A. P. J. Kalja, Analysing the Web Log to Determine the Efficiency of Web Systems., in *Databases and Information Systems : Seventh International Baltic Conference on Databases and Information Systems.*, Vilnius, Lithuania., 2006.

[78] Y. Li and N. Zhong, Mining Ontology for Automatically Acquiring Web User Information Needs., in *IEEE Trans. on Knowledge and Data Engineering.,* vol. 18, no. 4, pp. 554-568, 2006.

[79] N. Farhanaz Azam, Spectral Clustering: An Explorative Study of Proximity Measures, University of Ottawa, Master Thesis, Ottawa, ON, Canada, 2009.

[80] I. Peña, Utility-based Data Mining: An anthropometric case study., University of Ottawa, Master Thesis., Ottawa, ON, Canada., 2008.

[81] S. S. 2. Microsoft, SQL Server 2008 Spatial Tools, [Online]. Available: http://www.sharpgis.net/page/SQL-Server-2008-Spatial-Tools.aspx. [Accessed 5 oct 2012].

[82] I. H. Witten and E. Frank, Data Mining: Practical machine learning tools and techniques., Morgan Kaufmann., 2005.

[83] City of Ottawa, Ottawa Ottawa Open Data., [Online]. Available: http://www.ottawa.ca/online_services/opendata/index_en.html. [Accessed 5 oct 2012].

[84] ALTOVA, XML Editor, [Online]. Available: http://www.altova.com. [Accessed 05 oct 2012].

[85] A. Environics, PRIZM C2 Segmentation System, [Online]. Available: http://www.environicsanalytics.ca/. [Accessed 5 oct 2012].

[86] GeoRepository, GeoRepository., [Online]. Available: http://georepository.com/projection_17709/MTM-zone-9.html. [Accessed 5 oct 2012].

[87] T. Holden and J. Adolf, Human Research Facility (HRF) Human-Computer Interface (HCI) Design Guide., in National Aeronautics and Space Administration (NASA)., Houston, TX, USA., 1997.

[88] A. Pavlos, UTASTAR in Java, [Online]. Available: http://utastarinjava.sourceforge.net/. [Accessed 5 oct 2012].

[89] StatGraphics, StatGraphics, [Online]. Available: http://www.statgraphics.net/. [Accessed 5 oct 2012].