Acknowledgments

Resumen

Abstract

List of Figures

List of Tables

Contents

Chapter 1

Introduction

It is a reality that the World Wide Web in recent years, is growing exponentially, which means the presence of millions of users on Web sites, Web applications, Web systems, etc. []. There is a wide variety of Web systems, where we have different users interacting with them. These users have different goals when using these Web systems. For example do a search in Google [] of particular topic, make a reservation for a room in a luxury resort, check your bank account or simply view your status on your Facebook account [].This variation of users represents a complex diversity as individuals []. This diversity lies in different skills, interests, preferences and ways of thinking, learning and knowledge [].For this reason users need different ways to interact with the information presented by the great variety of Web systems that exist.

When we intend to customize any element in Web system, we need to know the user's personal information. This information is a collection of needs, characteristics, feelings, tastes, etc. This information is required to be able to form the representation of knowledge about users. This is what is known as user modeling (UM).

A user modeling can be as simple as a profile systems where is basic knowledge of users. Also can be as complex as represent its characteristics, needs, interests, ways to feel. In order to understand specific users. The main goal of user modeling is to represent aspects of the real world of the user's in autonomous automatically way.

In this document we present a user modeling in the context of Web-based interactive evolutionary computation.

Interactive evolutionary computation (IEC) is a branch of evolutionary computation where users become a part of the evolutionary process by replacing the fitness function; evaluating individuals of a population based on their personal preferences[13]. These evaluations are subjective according to the user point of view based on their perceptions, interests and desires.

Normally such systems require users to evaluate large amounts of individuals iteratively, causing them to lose interest for participate by fatigue that is generated[13]. Nowadays some of these systems are migrating to Web technologies looking for volunteers users to collaborate in the evaluations for distribute the load and lower the fatigue. Having Web- based interactive evolutionary systems open the possibility for linked to social platforms in order to involve the largest number possible of users to assist in the evaluation of individuals produced by these systems applications.

Chapter 2

State of the art

Some 1986's Dawkins's research was the pioneer of a significant addition to the 1990s IEC algorithms research works[][Dawkins 1986].

There is two key research approach about his field:

Creative Approach: The Artificial Life (AL) was the base of creative approach. AL uses complex algorithms for biological life models emulation. To perform this task, it is needed to include some of the different techniques starting from right image treatment. Good graphic creation as well as a great music and quality sounds, [Sims 1991b], [Sims 1991c], [Sims 1994], [Dawkins 1986], [Disz 1997], [Unemi 2000]and [Unemi 2003].

Humanized technology approach: The concept of humanized technology approach comes from the approach that is focused on the IEC algorithms interface, this is the research of interaction between humans and computer systems. The main goal of this was to reduce the user's fatigue and to promote the inputs and outputs of algorithms to improve the efficiency of them. IEC has made his own way in practical fields such as engineering, education,etc., [Parmee 1993], [Ventrella 1994a], [Takagi 1996], [Poli 1997], [Parmee 1998] and [Takagi 1998].

Computer graphics (CG) The Biomorph of Dawkins was the first IEC research, from this research comes to many motivated works mostly about the Selfish Gene, come of these works are: [Ochoa 1998], [Mccormack 1993], and [Smith 2003].

In Dawkins work a conventional recursive algorithm was used as a baseline maintaining the main target of trees with an L-system Lindenmayer.

Some 1986's Dawkins's research was the pioneer of the significant addition to the 1990s IEC algorithms research works[Dawkins 1986].

There is two key research approach about his field:

Creative Approach: The Artificial Life (AL) was the base of creative approach. AL uses complex algorithms for biological life models emulation. In order to perform this task it is needed to include some of the different techniques starting from right image treatment , good graphic creation as well as a great music and quality sounds, [Sims 1991b], [Sims 1991c], [Sims 1994], [Dawkins 1986], [Disz 1997], [Unemi 2000]and [Unemi 2003].

Humanized technology approach: The concept of humanized technology approach comes from the approach that is focused on the IEC algorithms interface, this is the research of interaction between humans and computer systems. The main goal of this was to reduce the user's fatigue and to promote the inputs and outputs of algorithms to improve the efficiency of them. IEC has made his own way in practical fields such as engineering, education,etc., [Parmee 1993], [Ventrella 1994a], [Takagi 1996], [Poli 1997], [Parmee 1998] and [Takagi 1998].

Computer graphics (CG) The Biomorph of Dawkins was the first IEC research, from this research comes to many motivated works mostly about the Selfish Gene, come of these works are: [Ochoa 1998], [Mccormack 1993], and [Smith 2003].

In Dawkins work, a conventional recursive algorithm was used as a baseline maintaining the main target of trees with an L-system (Lindenmayer). This same L-system was the base for another experiment to create 2-D CG forms insects from a system called Blind Watchmaker who used L-system angles from L-system output intuitively selected; the creation was called biomorphs. These creations reach his target with the multiple selections of the users based on their preferences; all these selections acted like a natural adaptation filter.

We can find plenty of applications and works for fractal generation [Sims 1991a] and [Sims 1992], [Baluja 1993] and [Baluja 1994], [Lund 1995], or [Angeline 1996],[Raynal 1999] and [Lutton 2003], for rendering in tridimensional, [Todd 1991],[Broughton 1997], [Das 1994] and [Tam 2002], for generation of virtual creatures, [Sims 1994], [Rowland 2000], or aerodynamic surface design (wings), [NGuyen 1993], [NGuyen 1994] and [NGuyen 1997].

We can discover more than one additional way to use this research in the artistic field with several applications of IEC who are used for cartoon face construction and animations matters, like Mutator [Todd 1991], [Todd 1994] and[Todd 1999] or [Bentley 1999a].

The genetic programming (GP) applications offers a category called Interactive Genetic Programming (IGP) with many examples of successful application in tridimensional artwork for artistic animations or construction using mathematical equations as CAVE [Das 1994], [Papka 1996] and [Disz 1997], [Sims 1991], [Sims 1991a], [Sims 1992],[Sims 1993] and [Min 2004]. As this work consequence, Panspermia orPrimordial Dance was created.

Imagen

Imagen

The artistic field is only the first step of a great IEC implementation; it is important to mention another relevant projects called Galapagos, [Sims 1997], and SBART, [Unemi 2000]. The IEC application Galapagos Project is the exhibit in Tokio Multimedia Museum, (NTT Intercommunication Center) and this project originates engaging images to all visitors based on L-systems.

Imagen

There are created after one selection, to get a good solution through multiple repetitions. This action is performed with Genetic Programming (GP), after the calculation of each pixel value using trees of equations combining logarithm, maximum, and minimum, sine, root, cosine, exponential arithmetic operators. AnimationLab is found as an outstanding work who offer figures that can run or walk working with the user to receive more opportunities to be picked. A particular characteristic of all of the figures is that the figures extremities Mentioning open source works, we can find SBART as an IGP [Unemi 2000] tool to create graphics. SBART allow to users to evaluate 20 two-dimensional images, subsequently twenty new image has direction and angles.

Imagen

There are many examples for this field application as [McKenna 1990], [Ventrella 1994a], [Ventrella 1994b], or [Ventrella 1995], [Lim 1999] and [Lim 2000]. One of the Interactive Evolutionary Programming (IEP) artistic application was created by [Angeline 1996], as a fractal generation where the system allows the evolution of animations for the ones who were selected from the user, the application initially show only 10 animations to rate.

Music and sound

It is important to know how IEC was implemented in music generation, with several applications in this field. We will start mentioning the pioneer application

GENJAM, [Biles 1994], [Biles 1996] or [Biles 1999] and [Biles 2000]. Some other attractive works are Sonomorph, [Nelson 1993] and[Nelson 1995], or SBEAT, [Unemi 2003], [Horowitz 1994], [Onisawa 2000], [Tokui 2000] and [Fels 2002]. It is possible to hear a part of the music songs of these previously mentioned works broadcasted in the radio station WDYN. (100.1, New York, USA, WEBPage:http://www.wdyn.net/).

The IEC algorithms are the base for the functionality of the music generation systems, a visual representation of this is given in the below figure:

Imagen

Table 2.1: Comparison of context-aware recommender systems.

Application	Contextual Factor	Domain	Paradigm	Device
CoMoLE	**Time, available time, place, device, level of knowledge, learning style.**	**E-learning**	**Pre-filtering**	**Mobiles, PC, laptop.**
Moma-System	**Location, time.**	**E-commerce**	**Post-filtering**	**PC, laptop.**
UbiquITO	**Season, time, temperature.**	**Tourism**	**Post-filtering**	**Mobiles**
ReRex	**Distance of the point of interest, temperature, weather, season, weekend, companion, travel goal, transport.**	**Tourism**	**Model-based**	**Mobiles**
LifeTrack	**Location, time, day of the week, traifc noise(level), temperature, weather.**	**Music**	**Post-filtering**	**PC, Mobiles.**
CARS	**Location and season.**	**Restaurants**	**Post-filtering**	**PC, laptop.**
InCarMusic	**Driving style, road type, landscape, sleepiness, traffic conditions, mood weather and natural phenomena.**	**Music**	**Model-based**	**Mobiles**
REJA	**Location.**	**Restaurants**	**Pre-filtering and Post-filtering**	**PC, laptop, mobiles.**
CiberGuide	**Location, time, weather.**	**Tourism**	**Post-filtering**	**Mobiles**
PECITAS	**Location, routes.**	**Transport**	**Post-filtering**	**Mobiles**
LARS	**Tourists location and time.**	**Tourist packages**	**Post-filtering**	**Mobiles**
I'm feeling LoCo	**Location, transportation.**	**Tourism**	**Model-based**	**Mobiles**
MOPSI	**Location**	**Tourism and transport**	**Post-filtering**	**Mobiles**

Chapter 3

Background

This chapter present the fundamental concepts related this work. The formal definitions referring to fuzzy systems, contextual factors and recommender system techniques used in the proposed method.

3.1 Production systems and fuzzy models

3.1.1 Traditional Production Systems

Production Systems represent knowledge in form of rules, which specify actions that will be executed when certain conditions are met. Experts in certain domain identify a set of rules based on their experience to resolve different kinds of problems. Also known as rule based systems, many implementations consist mainly of these three components [12] [27]:

1. **Production Rules (PR)**. A set of production rules (also known as IF-THEN rules) having a two part structure; the antecedent, conformed by a set of conditions and a consequent set of actions.

2. **Working Memory (WM)**. Represents the current knowledge or facts that are known to be true so far. These facts are tested by the antecedent conditions of the rules and the consequent part can change them.

3. **Inference Engine (IE)**. This interpreter matches the conditions in the production rules with the data/instantiations found in the WM, deriving new consequences.

The basic operation of these systems is described as a cycle of three steps [12]:

1. **Recognize**: Find which rules are satisfied by the current WM. The antecedent part of the productions consists of a set of clauses connected by AND operators, when all these clauses have matching data on the WM the production has a chance of firing.

2. **Conflict Resolution**: Only one production can be fired at a time, so when two or more rules can be fired concurrently a conflict occurs. Among the production rules found in the first step, choose which rules should fire.

3. **Actions**: Change the working memory by performing the actions specified in the consequent part of all the rules selected in the second step. Changes occur by adding or deleting elements of the WM.

This cycle continues until no further production rules can be fired. This control strategy is data driven because whenever the antecedent part is satisfied the rule is recognized, this strategy is also named chain-forward. Other strategy is chain-backward in this case the work is done from the conclusion to the facts, to chain-backward, goals in working memory are match against consequents of the production rules.

A drawback that has been recognized in these traditional productions systems, is that some times rules are not fired in the Recognize step because no appropriate match exists in the WM. Partial matching of rules is not possible and this can be a limitation in some systems because premature termination of the cycle is not desired. An approach to handle partial matching is using fuzzy logic [27]. In the next section a review of the extension of production systems with fuzzy logic is preasented.

3.1.2 Fuzzy Production Rules

Fuzzy production rules use fuzzy logic sets to characterize the variables and terms used in the propositions of the rules. Fuzzy production rules or fuzzy *IF-THEN* rules are expressions of the form *IF* antecedent *THEN* consequent, where the antecedent is a proposition of the form *"x is A"* where x is a linguistic variable and A is a linguistic term. The truth value of this proposition is based on the matching degree between x and A. Propositions are connected by *AND*, *OR* and *NOT* operators. Some implementations of fuzzy rule-based systems also include other kinds of data types in their propositions, for example the FLOPS system includes fuzzy numbers, hedges, and non fuzzy data types (integers, strings and float) [39]. Depending on the form of the consequent, two main types of fuzzy production systems are distinguished [6]:

- **Linguistic fuzzy model**: where both the antecedent and consequent are fuzzy propositions.

- **Takagi-Sugeno fuzzy model**: the antecedent is a fuzzy proposition; the consequent is a crisp function.

As before, other non-fuzzy consequents can also be implemented, like the execution of commands or the addition of new data.

Linguistic Variables (LV) are variables that can be assigned linguistic terms as values, i.e. if we define a linguistic variable *SPEED* we can assign it the linguistic terms *SLOW*, *MEDIUM* or *FAST*. The meaning of these linguistic terms is defined by their membership functions (MF). *LV* can be defined as a *5-tuple*

$LV = < v, T, X, g, m >$ where v is the name of the variable, T is the set of linguistic terms of *v,* X is the domain (universe) of *v,g* is a syntactic rule to generate linguistic terms, m is a semantic rule that assigns to each term t its meaning *m(t)*, which is a fuzzy set defined in X.

3.1.3 Fuzzy Inference Systems

Fuzzy Inference Systems (FISs) also called *Fuzzy Models* are fuzzy production systems used for modeling input-output relationships. From this input-output view, Babuka [6] describes these systems as *"flexible mathematical functions which can approximate other functions or just data (measurements) with a desired accuracy"*. Fuzzy Productions Rules define the relationship between input and output variables. Input variables are defined in the antecedent part of the rule and the consequent part defines the output variables. These FIS are used mainly in control systems, and are basically composed of five modules[6]:

1. **Rule Base.** The set of fuzzy production rules.

2. **Database.** Where the membership functions are defined.

3. **Fuzzy Inference Engine.** This module executes the fuzzy inference operations.

4. **Fuzzifier.** This interface transforms the inputs of the systems (numerical data) into linguistic values.

5. **Defuzzifier.** This interface transforms the fuzzy results into numerical data.

Usually the Rule Base and Data Base modules are collectively called the Knowledge Base module. The steps involved in fuzzy inference in a FIS are [20]:

1. Compare the input variables with the membership functions in the antecedent, to obtain the membership values of each linguistic term. This step is frequently called fuzzification.

2. Compose through a specific T-Norm operator (mainly max-min or max-product) the membership values to obtain the degree of support of each rule.

3. Generate the qualified consequence (fuzzy or numeric) of each rule depending on the degrees of support. These outputs are then aggregated to form a unified output.

4. Then the output fuzzy set is resolved or defuzzified to a single numeric value.

Three main inference systems can be described:

- **Tsakumoto**: The output is the average of the weights of each rule numeric output, induced by the degree of support of each rule, the min-max or min-product with the antecedent and the membership functions of the output. The membership functions used in this method must be non-decrease monotonic.

- **Mamdani**: The output is calculated by applying the min-max operator to the fuzzy output (each equal to the minimum support degree and the membership function of the rule). Several schemes have been proposed to choose the numeric output based on the fuzzy output; these include the centroid area, area bisection, maximum mean, maximum criteria.

- **Sugeno**: The fuzzy production rules are used. The output of each rule is a linear combination of the input variables plus a constant term, and the output is the average of the support degree of each rule.

3.2 Context

The application of contextual information in recommender systems, there are previous approaches by assuming the existence of certain contextual factors, such as time, location, and the purchasing purpose, that identify the context in which recommendations are provided. An assumption for each of these contextual factors can have a structure; the time factor, for instance, it can be defined in terms of seconds, minutes, hours, days, months, and years. The classification of context that is proposed by Adomavicious[2] is based on the following two aspects of contextual factors: 1) what a RS may know about these contextual factors and, 2) how contextual factors change over time.

1. **What a recommender system may know about these contextual factors.** A recommender system can have different types of knowledge, which may include the exact list of all the relevant factors, their structure, and their values, about the contextual factors. Depending on what exactly the system knows (that is, what is being observed), it can classify the knowledge of a recommender system about the contextual factors into three categories:

    - **Fully observable**: The contextual factors relevant to the application, as well as their structure and their values at the time when recommendations are made, are known explicitly. For example, when recommending the purchase of a certain product, like a shirt, the recommender system may know only the `Time, PurchasingPurpose, and ShoppingCompanion factors matter in this application. Further, the recommender system may know the structure of all these three contextual factors, such as having categories of weekday, weekend, and holiday for Time. Further, the recommender system may also know the values of the contextual factors at the recommendation time (for example, when this purchase is made, with whom, and` for whom).

- Partially observable: Only some of the information about the contextual factors described above, is explicitly known. For example, the recommender system may know all the contextual factors, such as Time, PurchasingPurpose, and ShoppingCompanion, but not their structure. Note that there can possibly be different levels of partial observability. In this article we do not differentiate between them and group various cases of partially observable knowledge into this general category.
- Unobservable: No information about contextual factors is explicitly available to the recommender system, and it makes recommendations by utilizing only the latent knowledge of context in an implicit manner. For example, the recommender system may build a latent predictive model, such as hierarchical linear or hidden Markov models, to estimate unknown ratings, where unobservable context is modeled using latent variables.

2. **How contextual factors change over time.** Depending on whether contextual factors change over time or not, there are two categories:

   - **Static**: The relevant contextual factors and their structure remains the same (stable) over time. For example, in case of recommending a purchase of a certain product, such as a shirt, we can include the contextual factors of Time, PurchasingPurpose, ShoppingCompanion and only them during the entire lifespan of the purchasing recommendation application.

- **Dynamic**: This is the case when the contextual factors change in some way. For example, the recommender system (or the system designer) may realize over time that the ShoppingCompanion factor is no longer relevant for purchasing recommendations and may decide to drop it. Furthermore, the structure of some of the contextual factors can change over time (for example, new categories can be added to the PurchasingPurpose contextual factor over time).

On the other hand, Fling[21] considers four types of context that can be used by different applications:

- **Physical context**: representing the time, position, and activity of the user, but also the weather, light, and temperature when the recommendation is supposed to be used.

- **Social context**: representing the presence and role of other people (either using or not using the application) around the user and whether the user is alone or in a group when using the application.

- **Interaction media context**: describing the device used to access the system (for example, a mobile phone or a kiosk) as well as the type of media that are browsed and personalized. The latter can be ordinary text, music, images, movies, or queries made to the recommender system.

- **Modal context**: representing the current state of mind of the user, the users goals, mood, experience, and cognitive capabilities.

The contexts classification reachs to a general context definition adopted like the most suitable definition proposed by A. K. Dey and it was mentioned in chapter 1.

Then, an example to explain context is considering a context-aware application, an indoor mobile tour guide. Here, the entities are the user, the application and the tour sites. We will look at two pieces of information weather and the presence of other people and use the definition to determine if either one is context. The weather does not affect the application because it is being used indoors. Therefore, it is not context. The presence of other people, however, can be used to characterize the users situation. If a user is traveling with other people, then the sites that they visit may are the points of interest for the user. Therefore, the presence of other people is context because it can be used to characterize the users situation.
Previously understanding the context, it is likely to define **context-aware recom-**

mender systems, it is viable to adopt the definition of **A. K. Dey et.al**[19] to formalize what features it has a Context-aware recommender system: *"a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users task."*

This definition is closer to the real about behaviour of **context-aware recommender system** when it incorporates contextual information to get recommendations, in addition, is fewer confused and specific than other author's definitions.

3.3 Recommender systems

3.3.1 Collaborative Filtering algorithm

The idea behind collaborative recommendation approaches is to exploit information about past behavior or opinions of an exisiting user community for predicting which items certain user of the system will most probably like or be interested in [**?**]. Recommender systems are useful in several types of applications, however, their biggest impact has been mainly in ecommerce web sites in order to personalize the information for a particular user as the system can help to promote several items of his or her interest, thus increasing the sales of the on-line store. In traditional implementations a Collaborative Filtering (CF) algorithm takes as input a given *user-item* matrix of ratings to generate a prediction for each item-user pair indicating to what degree the current user will like or dislike an item. Subsequently with that information a list of the top n recommended items for the user can be generated. The generated list contains only those items that have not been reviewed by the user. Differents approaches are utilized for CF such as: a) user-based nearest neighbor recommendation, b) Item-based nearest neighbor recommendation and c) model-based recommendation.

a) *User-based nearest neighbor* is approach the is used the most because is relatibly easy to implement and offers acceptable results. Another advantage is tha only the rating matrix is needed to obtain recommendations. The neighborhood selection consists in taking the k nearest neighbors into account usind the threshold to define the size of the neighborhood. A neigborhood of small can not make accurate predictions, and on the other hand if the neighborhood is too large the information about the nighbours could not be significant.

To obtain the similarity value between a user and his neighbors, the Pearson correlations measure is commonly used, taking the values from $+1$ (strong positive correlation) to -1 (strong negative correlation) to define how similar a neighbor is. The similarity $sim(a, b)$ of users a and b, given the rating matrix R is denoted by the following equation:

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r_a})(r_{b,p} - \bar{r_b})}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r_a})^2}\sqrt{\sum_{p \in P}(r_{b,p} - \bar{r_b})^2}} \tag{3.1}$$

Where the symbol $\bar{r_a}$ corresponds to the average rating of user a. Subsequently, a formula to calculate the prediction of the user a for item p that also factors the relative proximity of the nearest neighbors N and $a's$ average rating $\bar{r_a}$ is denoted by the following equation:

$$pred(a,b) = \bar{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \bar{r_b})}{\sum_{b \in N} sim(a,b)} \tag{3.2}$$

b) *Item-based nearest neighbor* is the same idea than the *user-based*, the difference is that this approach tries to find similar items instead of similar users to make a prediction using the rating matrix. Then, in a *item-based* recommendation is to compute predictions using the similarity between items and not the similarity between users. To find similar items cosine similarity measure is defined, this metric measures the similarity between two *n-dimensional* vectors based on the angle between them. Therefore, the similarity between two items a and b viewed as the corresponding rating vectors a and b is formally defined as follows:

$$sim(\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a} * \overrightarrow{b}}{|\overrightarrow{a}| * |\overrightarrow{b}|} \tag{3.3}$$

The * symbol is the dot product of vectors. $|a|$ is the Euclidian length of the vector, which is defined as the square root of the dot product of the vector with itself.

c) *Model-based approach*, in this technique the raw data are first processed offline, as described for *item-based* filtering or some dimensionality reduction techniques. At run time, only the learned model is required to make predictions. Although *memory-based* approach is theoretically more precise because full data is available for generating recommendations, such systems face problems of scalability when databases of tens of millions of users and items are used. An example of this approach is matrix factorization or latent factors model, normally used to fill a rating matrix to calculate predictions taking in account the latent factors.

3.3.2 Content-based algorithm

In content-based the recommendation task then consists of determining the items that match the users preferences best. Although such an approach must rely on additional information about items and user preferences, it does not require the existence of a large user community or a rating history that is, recommendation lists can be generated even if there is only one single user. In practical settings, technical descriptions of the features and characteristics of an item such as the genre of a book or the list of actors in a movie are more often available in electronic form, as they are partially already provided by the providers or manufacturers of the goods. What remains challenging, however, is the acquisition of subjective, qualitative features. In domains of quality and taste, for example, the reasons that someone likes something are not always related to certain product characteristics and may be based on a subjective impression of the items exterior design.

Content representation. The simplest way to describe catalog items is to maintain an explicit list of features for each item (also often called attributes, characteristics, or item profiles). For a book recommender, one could, for instance, use the genre, the authors name, the publisher, or anything else that describes the item and store his information in a relational database system. When the users preferences are described in terms of his or her interests using exactly this set of features, the recommendation task consists of matching item characteristics and user preferences.

Vector space model. Content-based systems have historically been developed to filter and recommend text-based items such as e-mail messages or news. The standard approach in CB recommendation is, therefore, not to maintain a list of *meta-information features*, but to use a list of relevant keywords that appear within the document. The main idea, of course, is that such a list can be generated automatically from the document content itself or from a free-text description thereof.

3.3.3 Hybrid recommender systems

Each recommender system technique has its pros and cons for instance, the ability to handle data sparsity and cold-start problems or considerable efforts for knowledge acquisition and engineering.

User models and contextual information, community and product data, and knowledge models constitute the potential types of recommendation input. However, none of the basic approaches is able to fully exploit all of these. Consequently, building hybrid systems that combine the strengths of different algorithms and models to overcome some of the afore mentioned shortcomings and problems has become the target of recent research. Hybrid recommender systems are technical approaches that combine several algorithms or recommendation components.

3.3.4 Context-aware recommender systems

Traditionally, the recommendation problem has been viewed as a prediction problem in which, given a user profile and a target item, the recommender systems task is to predict that users rating or that item, reflecting the degree of users preference for that item.

Specifically, a recommender system tries to estimate a rating function: $R : Users * Items \leftarrow Ratings$, that maps *user-item* pairs to an ordered set of rating values.

In contrast to the traditional model, context-aware recommender system tries to incorporate or utilize additional evidence (beyond information about users and items) to estimate user preferences on unseen items.

When such contextual evidence can be incorporated as part of the input to the recommender systems, the rating function can be viewed as *multidimensional*: $R : Users * Items * Contexts \leftarrow Ratings$, where **contexts** represents a set of factors that further delineate the conditions under which the *user-item* pair is assigned a particular rating.

The underlying assumption of this extended model is that user preferences for items are not only a function of items themselves, but also a function of the context in which items are being considered[29].

3.3.5 Paradigms

When recommender system uses the contextual information, it starts with the data having the form $U * I * C * R$, where C is additional contextual dimension and end up with a list of contextual recommendations i_1,i_2,i_3...i_n for each user. However, when the recommendation process does not take into account the contextual information, is posible to apply the information about the current (or desired) context c in various stages of the recommendation process. Adomavicious defines three paradigms to the context-aware recommendation process that is based on contextual user preference:

- **Contextual pre-filtering (or contextualization of recommendation input).** In this recommendation paradigm, contextual information drives data selection or data construction for that specific context. In other words, information about the current context c is used for selecting or constructing the relevant set of data records (i.e., ratings). Then, ratings can be predicted using any traditional 2D recommender system on the selected data.

- **Contextual post-filtering (or contextualization of recommendation output).** In this recommendation paradigm, contextual information is initially ignored, and the ratings are predicted using any traditional 2D recommender system on the entire data. Then, the resulting set of recommendations is adjusted (contextualized) for each user using the contextual information.

- **Contextual modeling (or contextualization of recommendation function).** In this recommendation paradigm, contextual information is used directly in the modeling technique as part of rating estimation.

Chapter 4

Proposed method

4.1 Data models

The data models was implemented in the DBMS with PostgreSQL database. All the information in the context-aware recommender system was managed in a scheme of a relational database. Each model is referring in its section in order to have a better comprehension of each one.

4.1.1 Restaurant model

An effective on-line recommender system must be based upon an understanding of consumer preferences and successfully mapping potential products into the consumers preferences[2]. Pan and Fesenmaier[34] argued that this can be achieved through the understanding of how consumers describe in their own language a product, a place, and the experience when they are consuming the product or visiting the place.

Traditionally, choosing a restaurant has been considered as rational behavior where a number of attributes contribute to the overall usefulness of a restaurant. For example: food type, service quality, atmosphere of the restaurant, and availability of information about a restaurant, plays an important role at different stages in consumers desitions making[5]. While food quality and food type have been perceived as the most important variables for consumers restaurant selection, situational and contextual factors have been found to be important also. Due to this in Kivela[26] identifies 4 types of restaurants: 1) fine dining/gourmet, 2) theme/atmosphere, 3) family/popular, and 4) convenience/fast-food; and Auty[5] identifies 4 types of dining out occasions: 1) namely celebration, 2) social occasion, 3) convenience/quick meal, and 4) business meal.

Taking in account the context, the restaurant model proposed for context-aware recommender system was definded with 55 attributes about the restaurants features. An exploration about contents of models of others works were compared to define the suitable information into the model. Therefore, the restaurant model is a binary vector with the following contextual attributes: 1) price range, 2) payment type, 3) alcohol type, 4) smoking area, 5) dress code, 6) parking type, 7) installations type,

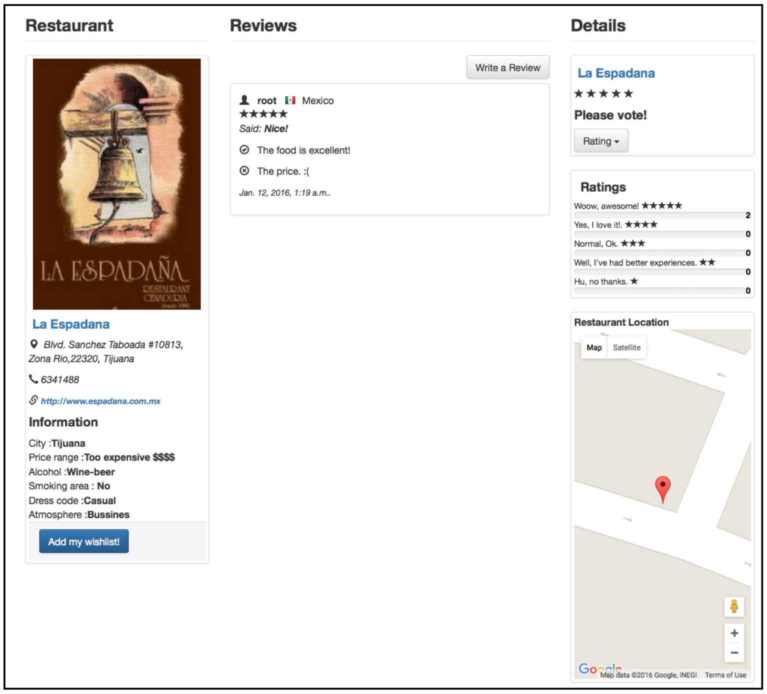


Figure 4.1: User interface of the restaurant model.

8) atmosphere type, and 9) cuisine type. An example of restaurant model in the context-aware recommender system is depicted in figure 4.1 with some domain values of the context represented by a binary vector where 1 means that the restaurant has the property that corresponds to the position value. Additionally, the restaurant model contains contextual information such as users's reviews, ratings average, and geographycal location.

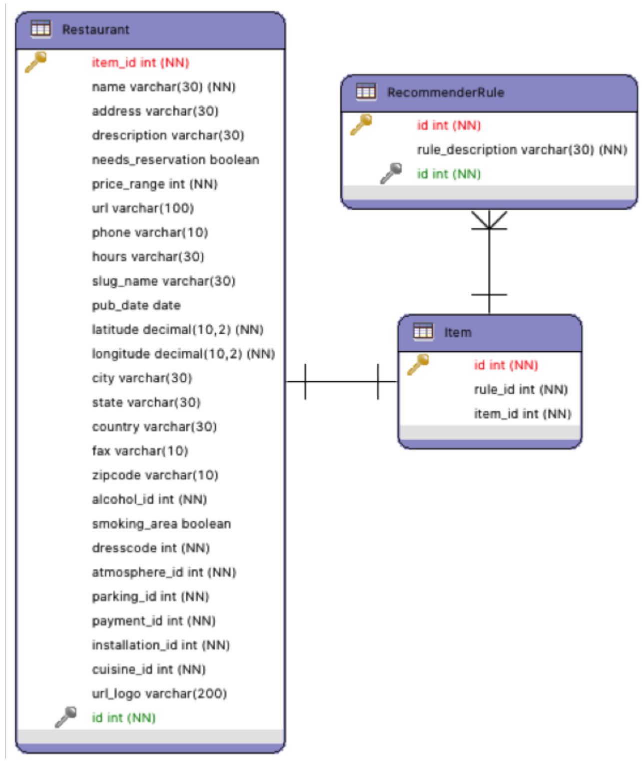


Figure 4.2: The data model of restaurant.

Data model

The data model in postgreSQL is depicted in the figure 4.2, the model contains the restaurant entity and its attributes. The restaurant entity is related to *Item entity* in a *"one-to-one"* relation that at the same time is related to the *RecommenderRule entity* which specifies some restrictions for item recommendations. A large view of all the entities related is depicted in the whole scheme refered in figure 4.5.

Some related entity corresponds to the proposed catalogues, that are defined as following:

- **Installations:** garden, terrace, indoor, outdoor.
- **Atmosphere:** relax, familiar, friends, bussines, romantic.
- **Parking:** no parking, free parking, valet parking.
- **Payment:** credit/debit card, cash.
- **Smoking area:** yes, no.
- **Price:** cheap, regular, expensive, too expensive.
- **Dresscode:** casual, informal, formal.
- **Alcohol:**no alcohol, wine-beer.
- textbfCuisine: japanese, chinese, italian, argentinean, cantonese, mandarin, mongolian, arabic, greek, spanish, brasilian, swiss, szechuan, asian, international, steak grill,vegetarian, natural/healthy/light, traditional mexican, tacos, mediterranean, middle eastern, american/fast food, gourmet, pizza, bar/beer, tapas cafe/bar, french, birds, seafood.

The cuisines were defined according the food variety of restaurants in Tijuana, there are 30 kinds of cuisines defined in the system.

The smoking area is the unique attribute with boolean value, it defines if a restaurant has a smoking area into its installation.

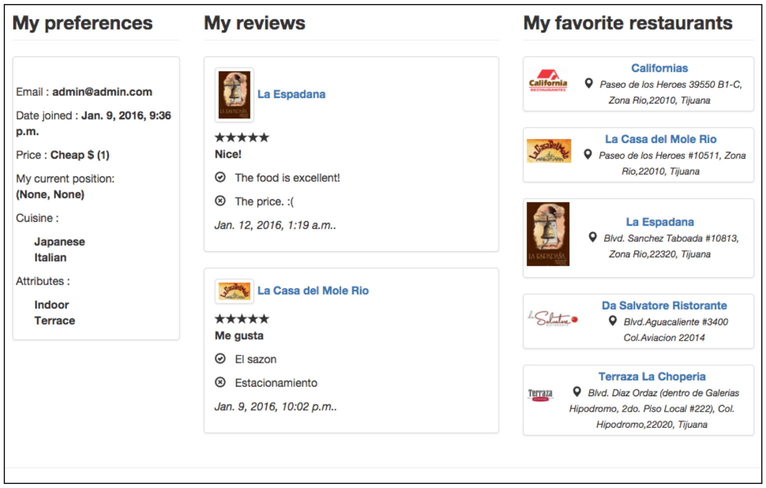


Figure 4.3: Example of user interface for user profile.

4.1.2 User model

The user's profile is derived from the ratings matrix. Let $U = [u_1, u_2, ...u_n]$ the set of all users and $I = [i_1, i_2,...i_n]$ the set of all items, if R represent the ratings matrix, an element $R_{u,i}$ represents a users rating $u \in U$ in a item $i \in I$. The unknown ratings are denoted as $\neq$. The matrix R can be decomposed into rows vectors, the row vector is denoted as $\overrightarrow{r_u}$=$[R_{u,1}...R_{u,|I|}]$ for every $u \in U$. Therefore, each row vector represents the ratings of a particular user over the items. Also denote a set of items rated by a certain user u is denoted as $I_u = i \in I | \forall i : R_{u,i} \neq \emptyset$. This set of items rated represents the user preferences where for each domain element $R_{u,i} \in [1 - 5]$ represents the intensity of the user interest for the item.

In context-aware recommender system, user profile has contextual information such as: 1) price range, 2) current location, 3) cuisine types, 4) attributes or features of restaurants that the user want, 5) the reviews posted, and 6) the favorite restaurants list. The user profile is stored in database and it is available for queries request, and it can be changed by users many times in a session. The information used to recommendations is the last one register stored. The user interface is represented in figure 4.3.

Data model

The user's data model in postgreSQL is represented in the figure 4.4, the model involves the entities: *User, UserProfile, and Friends. UserProfile entity* provides the contextual information of user, *User entity* is the default model defined in the system and is related to userProfile for suplies valuable information. The *Friends entity* represents the social aspect into the userProfile, Friends involves the users related to the current user taking in account the preferences of each other.

The user profile entity is related with 3 catalogues: price and cuisine are the same that in restaurant model, attribute groups corresponds to restaurant model mentioned (section 4.1.1). A total of 55 attributes(or features) could be contained in user profile, this information is used such as contextual information also. The domain values of the related catalogues are following:

- **Price:** cheap, regular, expensive, too expensive.

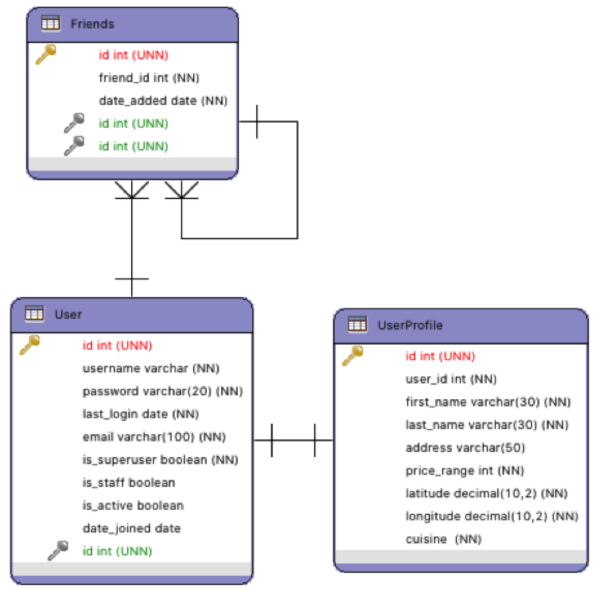


Figure 4.4: The data model of user profile.

- **Cuisine:** japanese, chinese, italian, argentinean, cantonese, mandarin, mongolian, arabic, greek, spanish, brasilian, swiss, szechuan, asian, international, steak grill,vegetarian, natural/healthy/light, traditional mexican, tacos, mediterranean, middle eastern, american/fast food, gourmet, pizza, bar/beer, tapas cafe/bar, french, birds, seafood.

- **Attribute groups:** Installations, atmosphere, parking, payment, smoking area, dresscode, alcohol.

4.1.3 Relational data model

A complete database relational scheme is represented in the figure 4.5. This model involves the whole database for context-aware recommender system, as well as the entities and relations among them.

The context is modeled as a relational database, each user context is a new register into data table to store user contexts.

Contextual information is also stored in the entities: *Reviews, CurrentLocation, DistancePoi and Ratings.* For instance, *Reviews entity* describes the users opinion about visited restaurants, this information contributes to have additional information about recent preferences of diners.

CurrentLocation entity stores the geographical position of user to get a *"nearby recommendation"*, the system locates restaurants around 2 kilometers from the user position. The position is changed frequently, in this manner, it allows the adaptation for each particular situation. *Distance Poi entity* stores the distances (kilometers) between the user and restaurants, this information is used to calculate *"nearby recommendation"*, each recommended restaurant ought be over the threshold defined. Finally, *Rating entity* represents the user preferences in a vector of scores, ratings could be increased in time and the user's preferences patterns could be changed in time also.

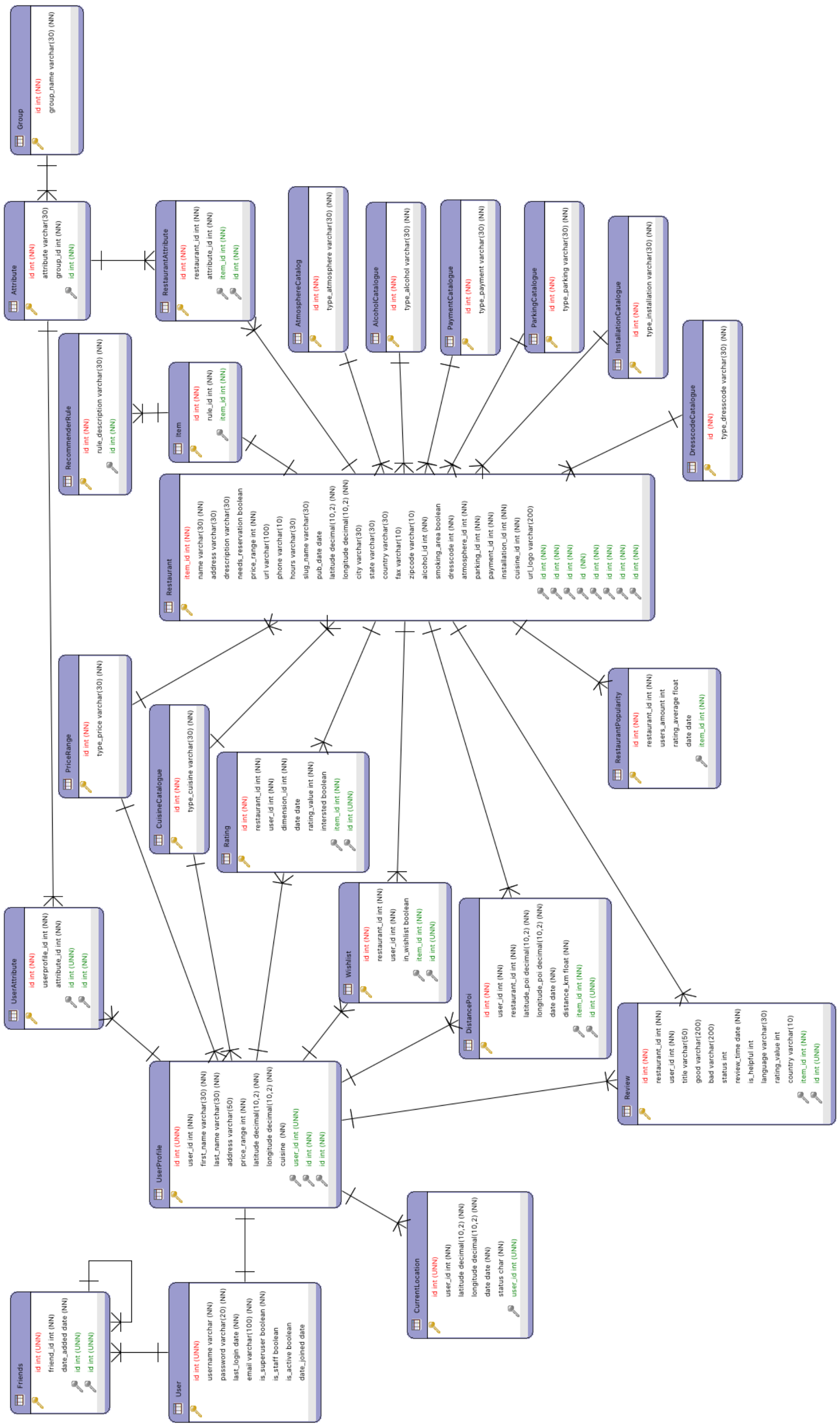


Figure 4.5: The relational database of context-aware recommender system.

4.2 Expert recomendation

Fuzzy logic is a methodology that provides a simple way to obtain conclusions of linguistic data. Is based on the traditional process of how a person makes decisions based in linguistic information.

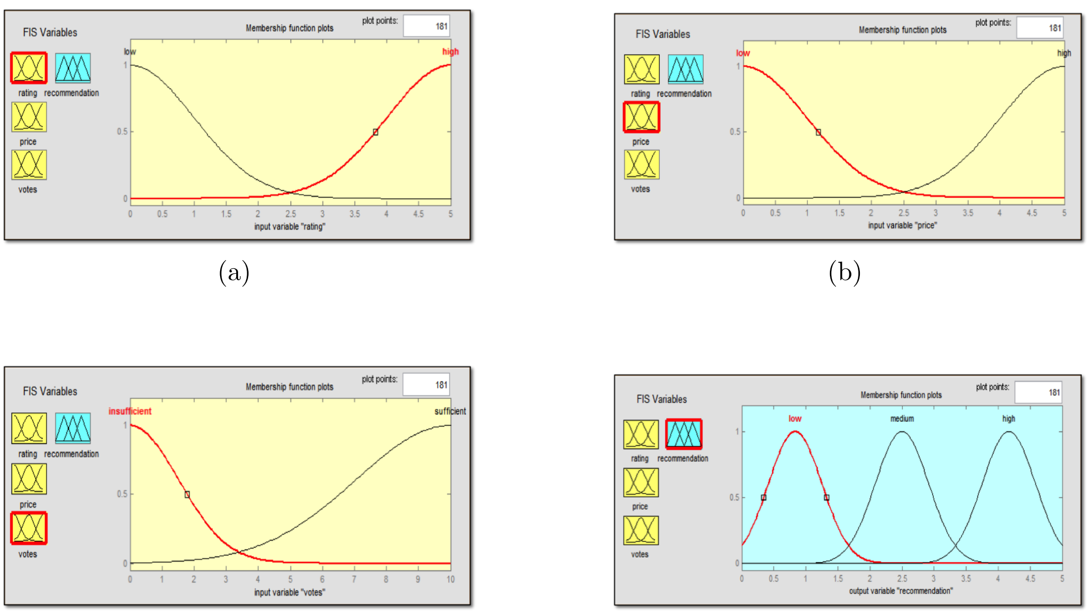
Fuzzy logic is a computational intelligence technique that allows to use information with a high degree of inaccuracy; this is the difference with the conventional logic that only uses concrete and accurately information [46].

In this work, fuzzy logic is used to model fuzzy variables that highligh in the popularity of a restaurant. The context-aware recommender system has implemented a Fuzzy Inference System that represents the expert recommendation.

The expert Fuzzy Inference System generates recommendations when the recommendation techniques (collaborative filtering, content-based) are not getting results because of the cold start problem.

The Fuzzy Inference System proposed has 3 **input variables** (such as in previous work realized[22]): 1)*rating* is an average of ratings that has a particular restaurant inside the user community; the domain of variable is 0 to 5 and contains 2 membership functions labeled as *low* and *high*(figure 4.9a), 2)*price* represents the kind of price that has a particular restaurant; the domain of variable is 0 to 5 and contains 2 membership functions labeled as *low* and *high* (figure 4.9b), and 3)*votes* is used to measure how many items have been rated by the current user, i.e., the participation of the user, if the user has rated few items (less than 10) is not sufficient to make accurate predictions(figure 4.9c), the domain of variable is 0 to 10 and contains 2 membership functions labeled as *insufficient* and *sufficient*.

The **output variable** is *recommendation*, represents a weight for each restaurant proposed by the expert considering the inputs mentioned above, the domain of variable is 0 to 5 and contains 3 membership functions labeled as *low*, *medium* and *high* (figure 4.10c).



(a) (b) (c) (d)

Figure 4.6: The Gaussian membership functions of the expert system.

The proposed Fuzzy Inference System(figure 4.7) represents the users experience and their knowledge about restaurants. This factors are considered important for users that visiting a restaurant. This information is recovered of user profile and restaurant profile; and the system uses this information to get weights that influence in the final recommendations. The Fuzzy Inference System uses 5 inference rules that involve the variables of inputs and output.The input variables determine the recommendation activation; each input variable contains labels as *low* and *high* that also correspond to memberships functions of Gaussian type. For the output variable *recommendation* the labels *low*, *medium*, and *high* are used with membership functions Gaussian type also. The rules are:

1. *If* ***rating*** *is high and* ***price*** *is low then* ***recommendation*** *is high.*

2. *If* ***rating*** *is high and* ***votes*** *is sufficient then* ***recommendation*** *is high.*

3. *If* ***rating*** *is high and* ***votes*** *is insufficient then* ***recommendation*** *is medium.*

4. *If* ***rating*** *is low and* ***price*** *is high and then* ***recommendation*** *is low.*

5. *If* ***rating*** *is low and* ***votes*** *is insufficient then* ***recommendation*** *is low.*

4.3 Fuzzy Inference System to assing weights

The main goal of this Fuzzy Inference System is to define weights for each recommendation list. The recommendation technique is based in the amount of available

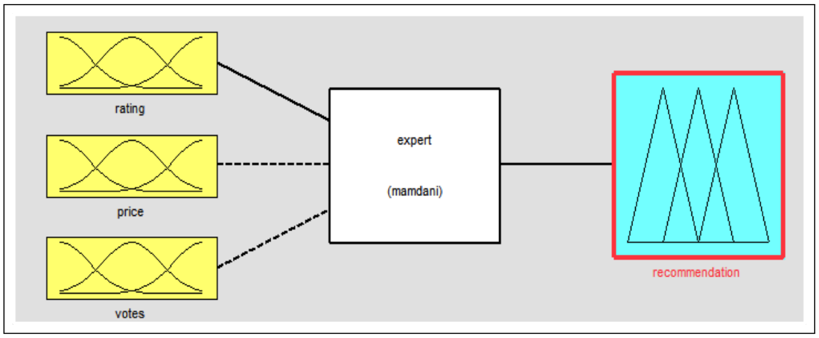


Figure 4.7: Fuzzy Inference System of expert.

information stored, so each technique utilizes this information to provide a list of restaurants as well as a weight for each one, therefore, these is used for recommendations if its weight is upper the threshold. The Fuzzy Inference System has inputs and outputs to infer each list's weight, its variables are depicted in figure 4.8. There are 3 membership functions for inputs and 3 for outputs. The input variables are: *userSimilarity, restaurantSimilarity and Participation* and are depicted in figure 4.9. The (4.9.a) and(4.9.b) are in a range from 0 to 1 to define the similarity average among users and restaurants, respectively. The figure (4.9.c) has a range from 0 to 15 to represent the ratings of the user(participation). This information is stored in the Popularity entity (see figure 4.5).

By other side, the output variables are: *Expert, RestaurantProfile and Correlation*, these are depicted in figure 4.10. The figure (4.10.a) represents the weight for expert recommendation list, figure (4.10.b) represents the weight of the content-based list and figure (4.10.c) represents the weight of collaborative recommendation list, their

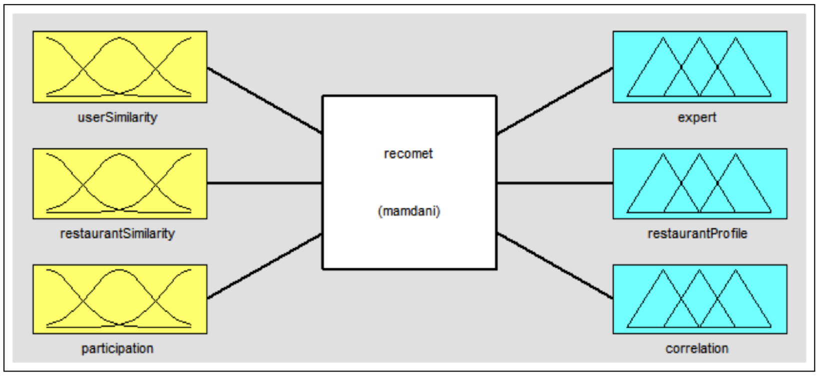


Figure 4.8: Fuzzy Inference System to assign weights.

membership functions are in a range from 0 to 1 to get the value. Taking in account

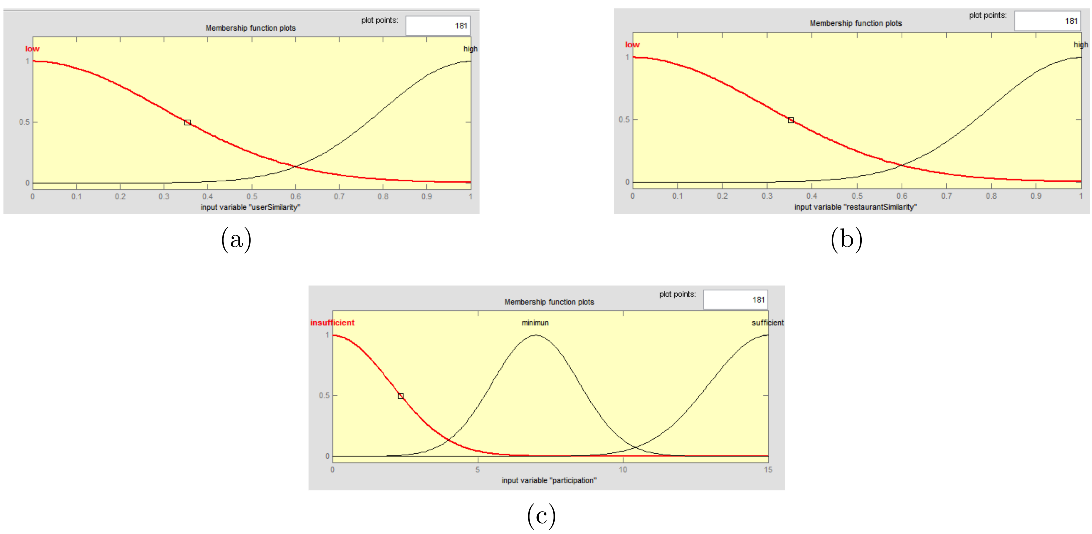
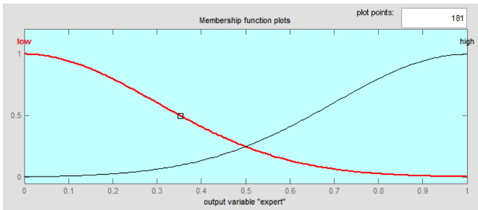


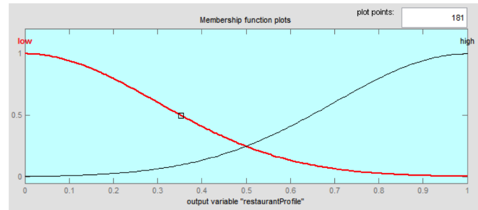
Figure 4.9: The Gaussian membership functions of input variables.

the input variables, the rules utilized to infer the output values are following:

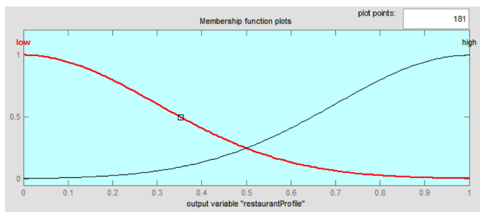
1. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is low and* ***participation*** *is insufficient then* ***expert*** *is high,* ***restaurantProfile*** *is low,* ***correlation*** *is low.*



(a)



(b)



(c)

Figure 4.10: The Gaussian membership functions of output variables.

2. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is low and* ***participation*** *is sufficient then* ***expert*** *is low,* ***restaurantProfile*** *is low,* ***correlation*** *is high.*

3. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is low and* ***participation*** *is minimun then* ***expert*** *is low,* ***restaurantProfile*** *is low,* ***correlation*** *is high.*

4. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is high and* ***participation*** *is insufficient then* ***expert*** *is low,* ***restaurantProfile*** *is high,* ***correlation*** *is low.*

5. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is high and* ***participa-***

***tion** is minimun then **expert** is low, **restaurantProfile** i s high, **correlation** is low.*

6. *If **userSimilarity** is low and **restaurantSimilarity** is high and **participation** is sufficient then **expert** is low, **restaurantProfile** is high, **correlation** is low.*

7. *If **userSimilarity** is high and **restaurantSimilarity** is low and **participation** is insufficient then **expert** is low, **restaurantProfile** is low, **correlation** is high.*

8. *If **userSimilarity** is high and **restaurantSimilarity** is low and **participation** is minimun then **expert** is low, **restaurantProfile** is low, **correlation** is high.*

9. *If **userSimilarity** is high and **restaurantSimilarity** is low and **participation** is sufficient then **expert** is low, **restaurantProfile** is low, **correlation** is high.*

10. *If **userSimilarity** is high and **restaurantSimilarity** is high and **participation** is insufficient then **expert** is low, **restaurantProfile** is low, **correlation** is high.*

11. *If **userSimilarity** is high and **restaurantSimilarity** is high and **participa-***

__tion__ is sufficient then __expert__ is low, __restaurantProfile__ is low, __correlation__ is high.

12. *If __userSimilarity__ is high and __restaurantSimilarity__ is high and __participation__ is minimun then __expert__ is low, __restaurantProfile__ is low, __correlation__ is high.*

4.4 Contextual Recommendation

The interface of the system(figure 4.11) allows to collect contextual information such as type of price, restaurant's attributes, type of cuisine and geographical location. The context- aware recommender system uses pre-filtering paradigm, then the contextual information is used for adjust the final recommendations list. For example, geographical location is used to get restaurants around 2 kilometers of distance, next, the list of nearby restaurants is displayed for the user. If context-aware recommender system considers another attributes as type of price and type of cuisine preferred by the user, the system gets restaurants matched in the context especified by the user in this time. In the attributes box, the user can chose any preference about what things are importants to select a restaurant. The features are collected from the dataset of Tijuana restaurants. In the cuisine box, the user choosen his/her favorite cuisine, it can be one or more cuisines such as in attributes also.

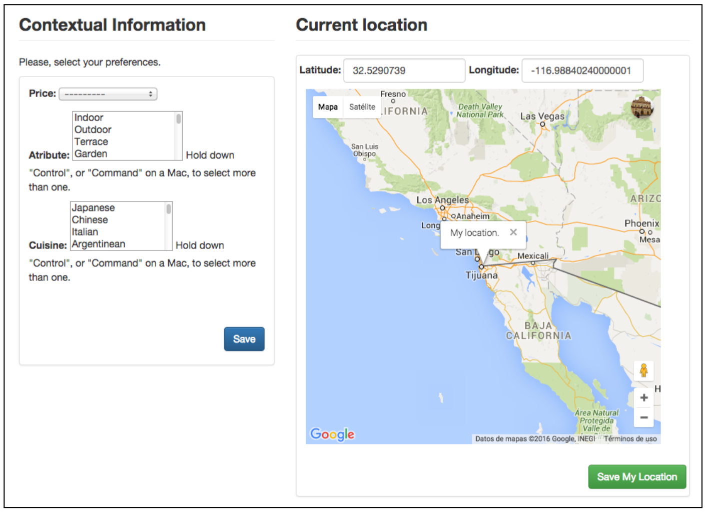


Figure 4.11: System interface to collect contextual information.

The context changes constantly, indeed, the users migh change it many times such as them wish. After the post-filtering, the system displays the recommended restaurants according the information provides by the user. The context-aware recommender system contains 4 techniques to display recommendations. The interface in figure 4.12 shows recommendations: *1) Expert, 2) Content- based, 3) Collaborative filtering and 4) Nearby.* Each one was explained above, except the nearby recommendations. For nearby recommendations the system calculates the approximate distance between the current geographical location of the user and the available restaurants in the area. The threshold is 2 kilometers around the user position to determine what restaurants will be recommended. The geographical position is obtained throught Google maps.

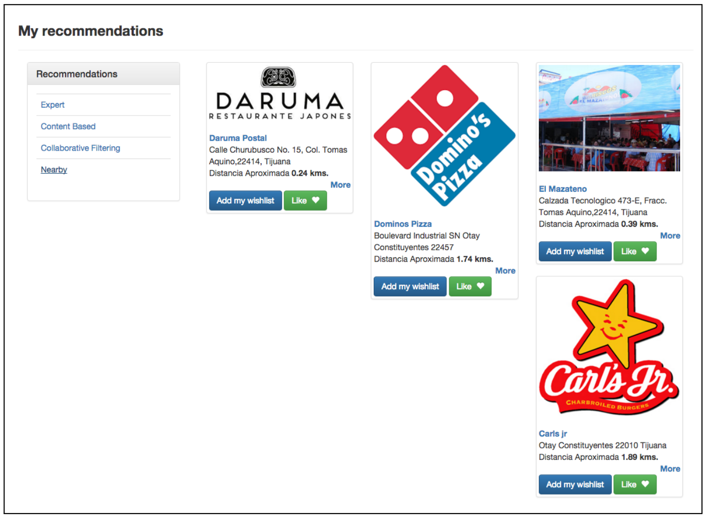


Figure 4.12: System inferface of recommendations for the user.

4.5 Architecture

The architecture for poposed method is depicted in the figure 4.13. In the first part, the three techniques of recommendations are suplied by the rating matrix to obtain the recommendation list. Ratings matrix makes that Fuzzy Inference System can obtain the inputs values to calculate the output value. The Content-based utilizes the rating matrix and user profiles to compare the similarity among the restaurants through cosine similarity. The collaborative filtering is based in rating matrix (user profiles) to predict ratings for restaurants using Pearson correlation to get the K neighbors.

The second part shows the recommendation lists for the user getting of each algorithm. Subsequently, the recommendation lists are reduced when filter context is

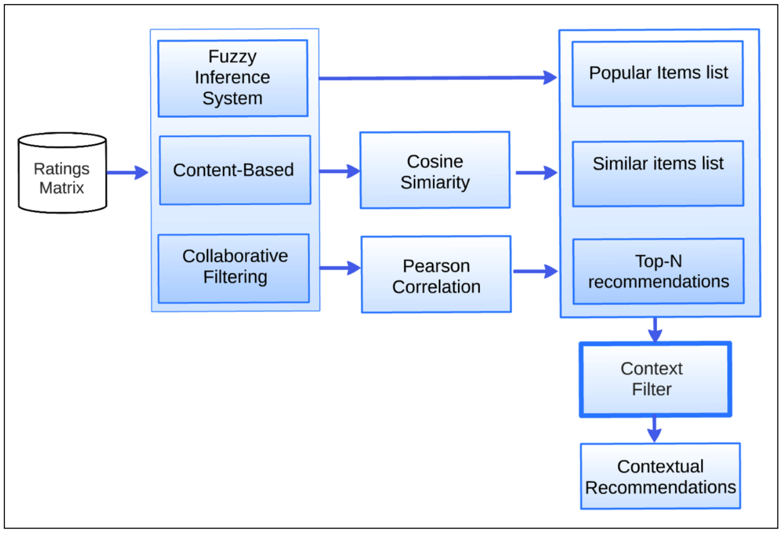


Figure 4.13: Context-aware recommender system architecture.

applied, i.e., the recommendations are adjusted for the user current context. Finally, the contextual recommendations list is displayed in the user interface (figure 4.12).

Chapter 5

Experiments and Results

In this chapter the results of the experiments described in chapter 4 are presented.

5.1 EvoDrawings01

In the experiment EvoDrawings01 the following data are generated, this data is presented in Table 5.1.

These data are the total of nodes and relations generated in EvoDrawings01.

The total number of active users who participated voluntarily meners are pre-

Table 5.1: Data generated in graph based user modeling.

	Data
Nodes	595
Relations	2220

Table 5.2: Total number of volunteers active users.

	Users
Total number of users	54

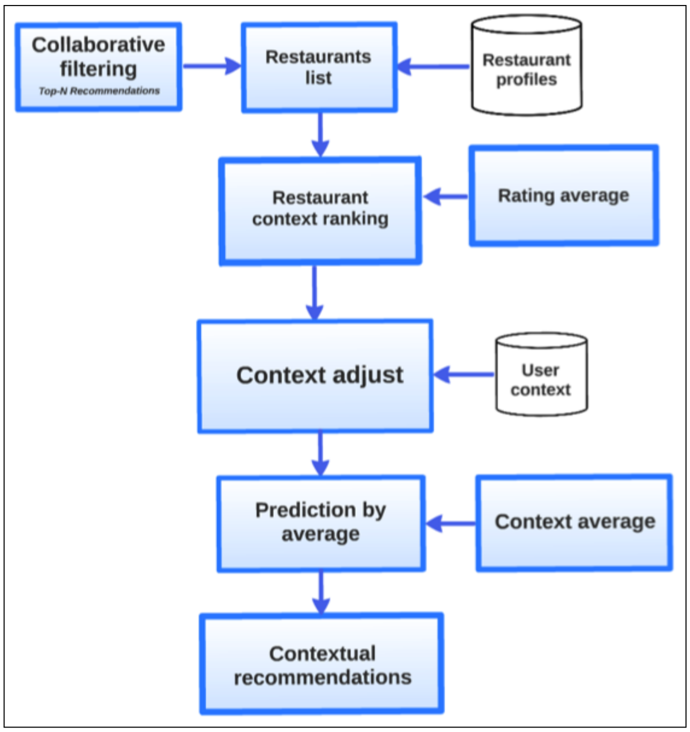


Figure 5.1: The Post-Filtering architecture for Tijuana restaurants.

sented in Table 5.2.

In order to validate the proposed approach, data about restaurant preferences of users in different contexts was collected. The study subjects were students enrolled in a computer engineer major, a masters program and professors of the Tijuana Institute of Technology. A total of **50 users** answered a questionnaire; the questions were about their preferences for nearby restaurants and the technology used by them.

The *questionnaire* consisted of **8 questions** and they rate restaurants from a list of 40 restaurants. Each restaurant chosen was rated 6 times one per context considered, a matrix rating with *1,422 ratings* were collected. The questions are shown in the table **??**.

The user's answers from question 1 to question 6 are represented in the figure 5.2. **Figure 5.2a** represents the percentage of surveyed students and teachers; **figure 5.2b** the percentage of the element that users consider the most important to visit a restaurant; **figure 5.2c** represents the preferences of devices when are using Internet for restaurant recommendations; **figure 5.2d** represents the percentage of operating system that often used, **figure 5.2e** shows the percentage of users that use the Internet to search restaurants in Tijuana; and **figure 5.2f**, shows the percentage of users that would like using a restaurant recommender system of Tijuana. For questions 7 and 8 only the top-ten restaurants are shown, without/with the contextual situation. In figure 5.3a, the favorite restaurant is **Daruma**(178 votes), whereas in figure 5.3b, **Daruma** does not appear in the top-ten. When considering the context *midweek*, the favorite restaurant was **Carls Jr.**, which appears in both graphs; this restaurant was also the most voted in the different contexts. Contextual recommendations of post-filtering approach depends of context *midweek* or *weekend*, which is the day when the restaurants were rated. Subsequently, the result of the query is refined according to the user context; the 6 contexts mentioned correspond

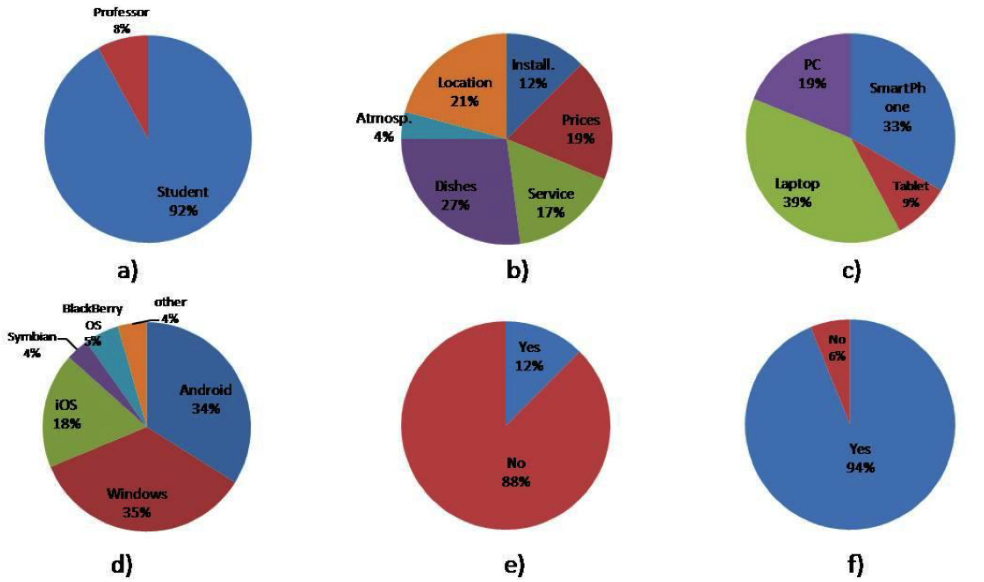


Figure 5.2: The chart shows the users preferences for questions from 1 to 6.

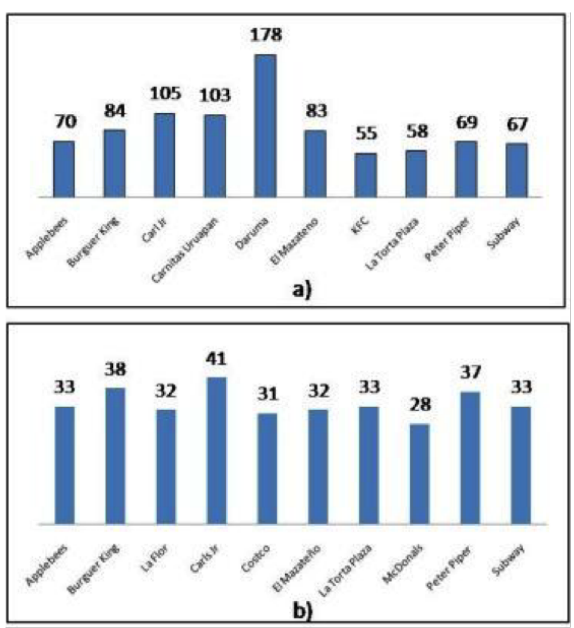


Figure 5.3: The chart shows the users preferences for questions 7 and 8.

Table 5.3: Contextual factors considered in the questionnaire.

Contextual Factor	Context
Day	1.Midweek(Monday, Thuesday,Wednesday and Thursday) 2.Weekend(Friday,Saturday and Sunday)
Place	1.School 2. Home 3.Work

to combinations of contextual factors shown in table 5.3. The dataset was explicitly collected from **50 users** whom answered questionnaire (see table **??**). A total of 172 predictions was made for different users and the error **MAE=0.5859** when the context **midweek** for current user was considered. The observation for this result is that using a small dataset the performance of the method proposed is limited. By other hand, having only one contextual factor does not improve the accuracy of the recommendations in this domain.

5.2 MovieLens dataset

GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org).

The data sets were collected over various periods of time, depending on the size of the set.

- **MovieLens 1M Dataset**: Stable benchmark dataset, 1 million ratings from

6000 users on 4000 movies. Released 2/2003.

Downloaded from *http://grouplens.org/datasets/movielens/1m/*.

- **MovieLens 10M Dataset**: Stable benchmark dataset, 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released 1/2009.

  Downloaded from *http://grouplens.org/datasets/movielens/10m/*.

The recommender system proposed for MovieLens uses post-filtering and time segmentation. Time in recommender systems is used as a contextual factor in the research reviewed [9], [8], [28], and [25], results vary according the techniques that were done.

In [25] the pre-filtering approach was used, time was divided in time intervals and the size of time intervals is directly proportional to the distance from initiating the historical information to the current user context. In [28] a tracking model of user behavior over the life-time of data is proposed, in order to exploit the relevant components of all data instances , while discarding only what is modeled as being irrelevant.

In [9] it is shown that the time division is beneficial and its performance depends on the items selection method and influence of contextual variables in item ratings. In [8] the user profile is segmented into micro-profiles corresponding to a particular

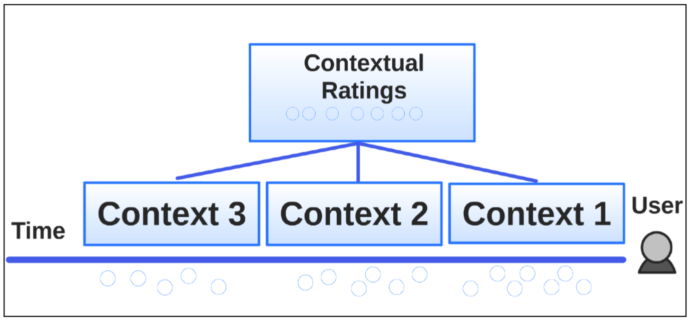


Figure 5.4: Time segmentation of contexts based on current user context.

context, each context represents a time span in which recommendations for users are derived.

This experiment implements fuzzy logic on time segmentation, in order to improve user satisfaction by providing recommendations based to context and recents user preferences without discarding tastes in the past, as they include important information for the recommender system proposed. The first phase is division of three time segments based on the current context of the user is performed such as in is depicted in figure 5.4.

In recommender system, the first step is get the current user context (user-application interaction), from this information three contexts (figure 5.4) will be obtained that representing a time segment of three months each one, in total the algorithm considers all the ratings users did during nine months prior the current context. Subsequently, ratings are classified by contexts and reused as contextual rating matrix being, a ratings matrix for each context.

The size of matrix depends of users participation during the last nine months. One of the aims is to identify the user behavior through recent information, in order to, for instance, know whether the user changes ratings constantly; whether usually assign high, low or mixed ratings; whether user likes to see different items or whether have a favorite category.

Recommender systems use the collaborative filtering algorithm in order to find relevant items for the user [43]. User's profiles are used for determine the similarity between users calculated with Pearson correlation. The similarity between users can provide valuable information as long as user participation is enough (less than 10 ratings). The next step is to obtain recommendations list (Top-N), three contextual lists are the outputs of collaborative filtering algorithm and contain items with user's predictions for each context.

Popularity's prediction considers other variables: 1) users participation in respect of an item in the context and, 2) the rating's average that users have given for item in the same context.

A Fuzzy Inference System (FIS) uses these parameters to assign a weight within a scale from 1 to 5 (prediction value). These recommendations are used when the ratings matrix is sparse, a popularity prediction is done.

Finally, the system gets the recommendations list for each user in different contexts. The recommendation process of pre-filtering is depicted in figure 5.5. The dataset

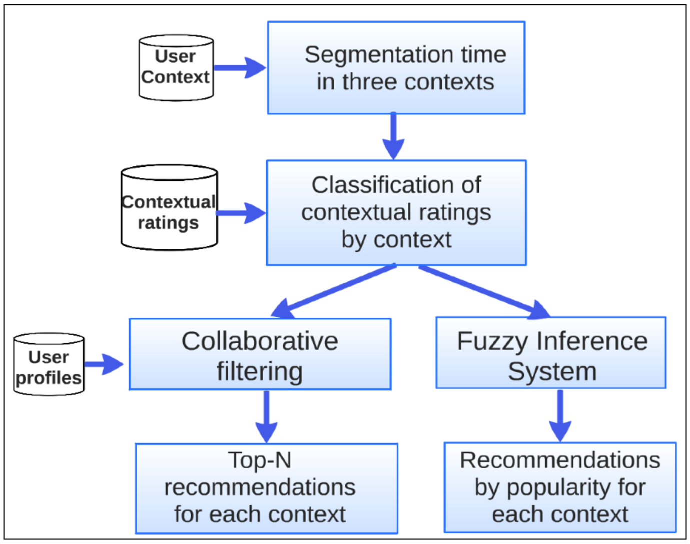


Figure 5.5: Pre-filtering process for context-aware recommender system.

used to test the algorithm was MovieLens(100000 ratings) with 943 users and 1,682 movies. The ratings were collected in a period of 2 years. MovieLens is not a contextual dataset, however, the timestamp was used to determine the rating time, i.e., in this way it was noted the day to know whether the rating time was in weekday or weekend. In this terms, the context was used. Then, the time for each context was divided in 3 months each one, this span covers 9 months before the user's current context.

The neighbors in each context are considered to recommend movies in that context only. An average of predictions are considered for add a movie to the top-N list of contextualized recommendations.The result in table 5.4 shows the error in three contexts. The error increase in context 3, in this context the ratings matrix is a

Table 5.4: Results of comparison by contexts in MovieLens dataset.

Context	# Preditions	MAE
1	12235	0.28
2	21049	0.24
3	1075	0.38

little bit sparse; the error is justifiable because user has less participations.

5.3 Tripadvisor dataset

The dataset used to evaluate the algorithm was TripAdvisor in two versions downloaded [47], this datasets was used in [49], [48] to evaluate the performance of context-aware recommender systems.

The first dataset contains 4669 contextual ratings, 1202 users and 1890 hotels; the second dataset contains 14175 contextual ratings, 2731 users and 2269 hotels. Data were collected of reviews online in tripadvisor.com. There is only one context: type of trip (family, friends, bussines, romantic and relax).

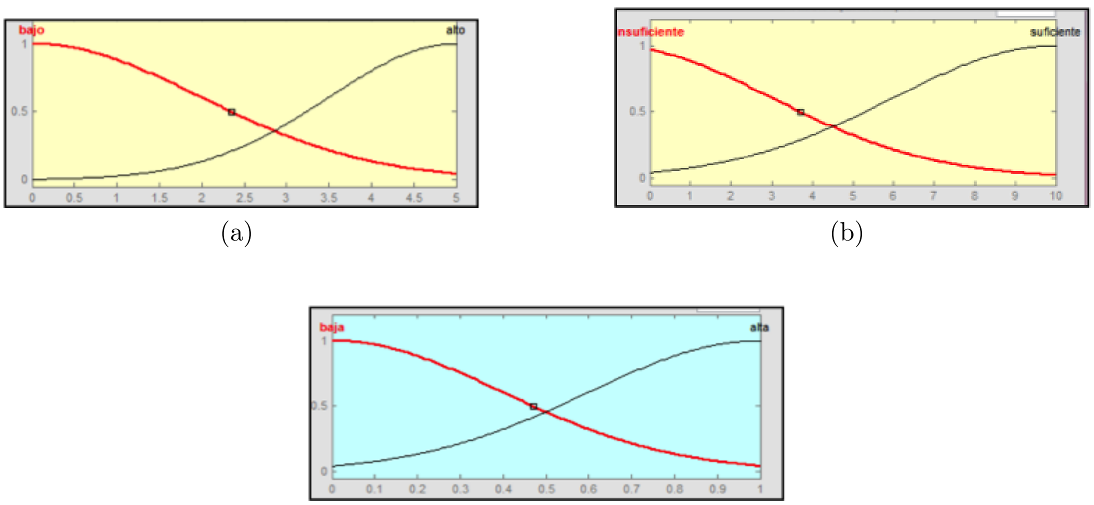
The proposed method consists of three algorithms to recommend: Fuzzy Inference System, collaborative filtering and content-based. Each one uses rating matrix to get recommendations.

The context-aware recommender system uses the post-filtering paradigm[2] for adjust recommendations in context. The recommendation by popularity is through the

Fuzzy Inference System depicted in figure 5.7, the Fuzzy Inference System contains the variables that are involved in the process to recommend in a human interaction, this process is the same that the recommender system does.

The output represents how matter each item into the users community, i.e. if it was a popular item for users.

The FIS has Gaussians membership functions and are depicted in figure 5.6. The



(a) (b) (c)

Figure 5.6: Gaussian Membership functions in the input are: a) RatingAverage, b) UserParticipation, and an output: c) Recommendation.

Fuzzy Inference System uses fuzzy rules to infer the inputs and output (a numeric value) that represents the weight of the recommendation. The rules are following:

1. *If* ***RatingAverage*** *is low and* ***UserParticipation*** *is insufficient then* ***recommendation*** *is low.*

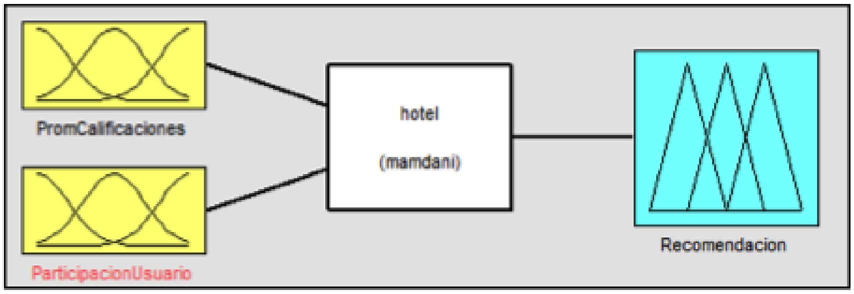


Figure 5.7: Fuzzy Inference System.

2. *If* ***RatingAverage*** *is low and* ***UserParticipation*** *is sufficient then* ***recommendation*** *is high.*

3. *If* ***RatingAverage*** *is high and* ***UserParticipation*** *is insufficient then* ***recommendation*** *is low.*

4. *If* ***RatingAverage*** *is high and* ***UserParticipation*** *is sufficient then* ***recommendation*** *is high.*

Content-based uses cosine similarity to compare the binary vectors representing the profile of each item, thereby obtaining a numerical value that determines similarity, based on a threshold.

In other words, it makes a comparison of profiles of each item to determine the most similar to items the user has rated with highest score, context-aware recommender system proposed has a scale from 1 to 5. In the next step the outputs of every recommender algorithm is represented by a list of recommended items. Subsequently applies the context filter and context-aware recommender system gets the final con-

Table 5.5: Example of contextual ratings in the user profile.

User profile		
Item1	Rating1	Context1
Item2	Rating2	Context2
Item3	Rating3	Context3

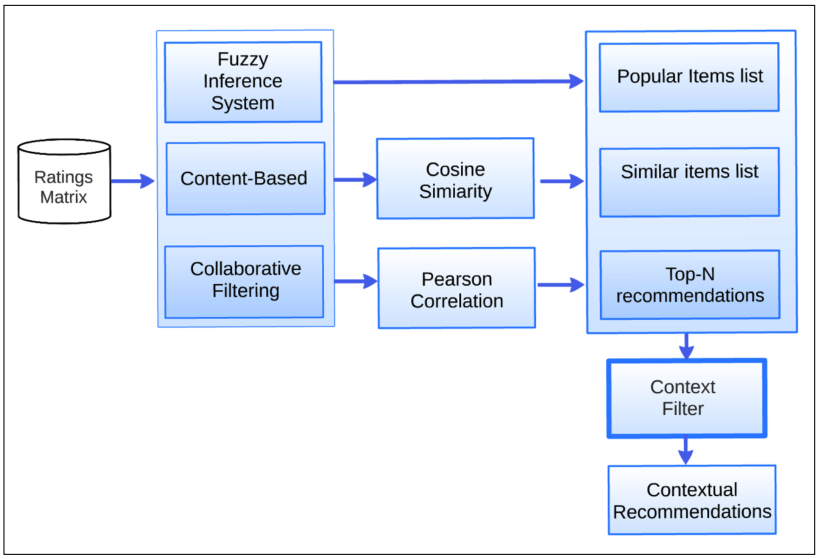


Figure 5.8: Recommender system architecture

textual recommendations.

Context-aware recommender system identifies contextual data of the user profile (see table 5.5), and compares recommended items to filter those items that are adjusted to the user context.

The context filtering is the last step before to get the recommended items. The schema of architecture for context-aware recommender system is depicted in figure 5.8. Two experiments were performed using TripAdvisor dataset, table 5.6 describes the data sets and the scarcity percentage of the specified data. Scarcity of 99% mean

Table 5.6: Datasets description.

Dataset	Users	Items	Ratings	Scarcity (percent)
TripAdvisor v1	1202	1890	4669	99.79
TripAdvisor v2	2731	2269	14175	99.77

Table 5.7: Comparison of RMSE.

Dataset	Algorithm	RMSE
TripAdvisor v2	FC + Post-filtering	0.504
	FC	0.994
	Pre-filtering + Relaxation	0.985

that there are problems to recommend items because the information is not enought to get good recommendations.

By other side, in table 5.7 the comparison shows that the algorithm has a acceptable performance, i.e., the error falls into the range of results obtained with others algorithms. Then, contextual recommendations were evaluated with the Root Mean Square Error in order to compare the results with context relaxation algorithm[48] that is evaluated with the same dataset.

The fundament of content-based is the cosine similarity; this means that if similarity value among items is high, the recommendations will improve the degree of user satisfaction. This is observed when calculating the similarity average in each dataset as shown in table 5.8.

FIS can provides a list of popular items for each dataset, recommendations through averages are obtained, and recommendations are conditioned to show it

Table 5.8: Level of similarity among items in datasets.

Dataset	Similarity	Avg.votes per user.
TripAdvisor v1	0.448	5
TripAdvisor v2	0.508	8

when the collaborative filtering and content- based are not delivering recommendations because of data scarcity. However, the majority of popular items of dataset were rated in contexts: romantic, family and bussines, that means that the dataset has biases.

In this experiment the context-aware recommender system proposed involves the paradigm of post-filtering for contextual recommendations. The structure of the datasets facilitated the evaluation of recommendations although the rating matrix has been scarce in both cases. Anyway, information of items and users was used to test the system and a good performance of the system was done.

With respect the performance, post-filtering allows select relevant items that are adjusted into the context, indeed, post-filtering and implementation of different recommendation techniques the system has suitable performance and the datasets help the processes performed.

Table 5.9: Contexts in InCarMusic dataset.

Context	Values
Driving style	elaxed, driving, sport driving.
Road type	city, highway, serpentine.
Landscape	coast line, country side, mountains/hills, urban.
Sleepiness	awake, sleepy.
Traffic conditions	free road, many cars, traffic jam.
Mood	active, happy, lazy, sad.
Weather	cloudy, snowing, sunny, rainy.
Natural phenomena	day time, morning, night, afternoon.

5.4 Datasets in matrix factorization

Filmtrust dataset

FilmTrust is a small dataset crawled from the entire FilmTrust website in June, 2011. Filmtrust contains a ratings matrix of 35498 ratings, 1504 users and 2071 movies. The dataset has a density of 1.14% and was used in [24] using the trust level such as context. The web page is *http://www.librec.net/datasets.html*.

InCarMusic dataset

InCarMusic dataset[10] has 8 contextual factors and the possible values for contextual conditions are explained in table 5.9. Music tracks were ten different genres. There is not unified music genre taxonomy, for this reason the recommender system uses the genres defined in [42]: classical, country, disco, hip hop, jazz, rock, blues, reggae, pop and metal, 50 music tracks and 42 users in dataset.

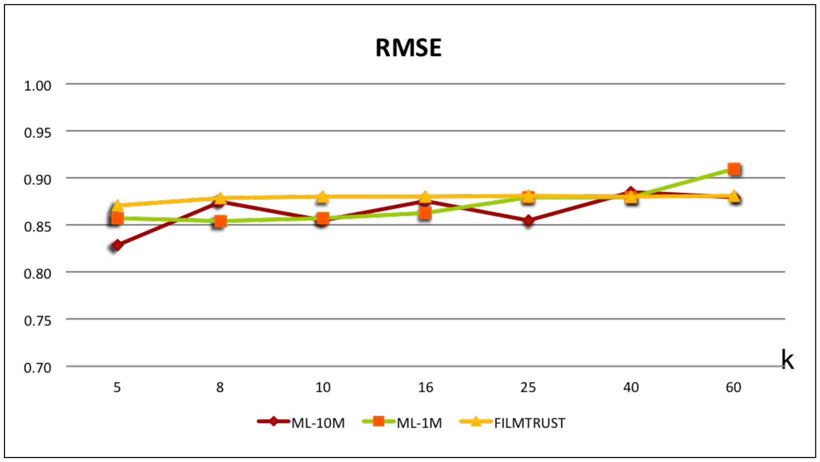


Figure 5.9: RMSE results of matrix factorization test.

5.4.1 Results

For experiments with matrix factorization technique the Graphlab toolbox was used. Both mentioned datasets and Movielens (1 millon and 10 millions) were used to test the algoritmh. The test involves K factors that are increasing for 50 iterations. previously, was done a test to identify what number of iterations are enough to get a good result with no overload of process in the algorithm. Results are depicted in the chart 5.9 where the axis (x, y) represent the K value and the error value, respectively. The observations deal to small differences among the datasets, in a range of 0.80-0.90, and the high variability is in MovieLens dataset 10 millions. The big dataset implies more unstable behaviour, while in a small dataset (Filmtrust) the error is less variable. A comparison among MovieLens 1 million and 10 millions shows that there's not a significant difference.

Table 5.10: RMSE of datasets using matrix factorization.

Dataset	Ratings	Cosine Sim.	RMSE
Tijuana Rest.	896	0.67	0.60
Mexico Rest.	1161	0.25	0.54
InCarMusic	4012	0.45	0.93
TripAdvisor	4669	0.17	0.85
MovieLens	10000	0.46	0.51
Movielens	100000	0.94	0.42

By other side, other datasets were used to test matrix factorization under the same parameters to calculate the RMSE for each one. Table 5.10 presents the total of ratings of each dataset, the cosine similarity, it means how similar are the items into the dataset, and the RMSE error obtained in the test with matrix factorization technique. The datasets contain less ratings than the presented in the chart 5.10, according the table 5.10 is not possible to assum that matrix factorization has a better performance with small datasets, because TripAdvisor and InCarMusic datasets obtain an error in the same range that the large datasets of the previuos chart.

Chapter 6

System evaluation

6.1 Metrics

To evaluate context-aware recommender system was used the **task success** and **time-on-task** metrics.

The **task success metric** is perhaps the most widely used performance metric. It measures how effectively users are able to complete a given set of tasks. The **time-on-task metric** is a common performance metric that measures how much time is required to complete a task[4].

Task success is something that almost anyone can do. If the users cant complete their tasks, then something is wrong. When the users fail to complete a simple task can be an evidence that something needs to be fixed in the recommender system.

The usability test consist of a list of simple tasks for users that they shall perform in the system to complete the test. Before to start, a minimal description about the system for user was explained. The tasks list are the following:

1. *Rated a restaurant without context.*
2. *Add context to the user profile.*
3. *Filter restaurants by favorite context.*
4. *Find information of a specific restaurant.*
5. *Find all the reviews of a specific restaurant.*
6. *Find section of my favorite restaurants.*
7. *Add a review of a restaurant.*
8. *Find the most popular restaurants.*
9. *Add a restaurant to your wishlist.*
10. *Get recommendations based on expert opinion.*
11. *Get the recommendations content-based.*
12. *Get the collaborative recommendations.*
13. *Get recommendations of the nearby restaurants.*

6.2 Enviromental set up

Each user did the task list, one by one, with previous instructions. It gives a brief explanation about the general features of system before to start. The time average for each user was around 10 minutes to finished all activities without disruptions. After, the results was depicted in a chart to observe the user behaviour for each task, in the figure 6.1 the axis (x, y) represent the task number and percent of success, respectively. The chart shows that only 3 tasks werent accomplished successfuly, the task 5, 6 and 7.

The issue with task 5 was that users can not found easily the reviews section in the interface, the issue in task 7 is derived of task 5 because the user couldnt find the manner to add a review. The task 6 correspond to the favorite restaurants, but the issue is that it was confused to chose favorite restaurants in place of wishlist section. In general, these results mean a possible redisign in the interfacte to facilitate the performance of these tasks. The time it takes a participant to perform a task says a lot about the usability of the application. In almost every situation, the faster a participant can complete a task, the better the experience. In fact, it would be pretty unusual for a user to complain that a task took less time than expected [4]. Then, task-on-time was applied to measure time that an user did the task. A resume of the time tasks for each user it is in table 6.1, *null* values mean that the user didn't

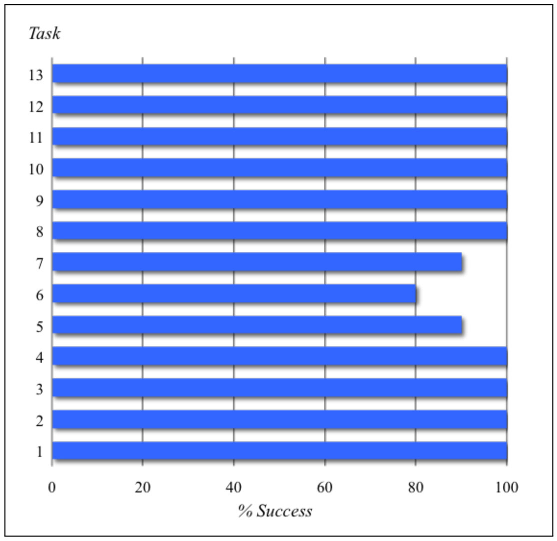


Figure 6.1: Representation of the percent of success for each task.

the task.

6.3 Results

To measure the efficiency of the metric it was chose an confidence interval. In this way, it is observed the time variability within the same task and also helps visualize the difference across tasks to determine whether there is a statistically significant difference between tasks. The obtained information is in table 6.2, the median was used to calculate the confidence interval. In the next step the USE *(Usefulnes, Satisfaction, and Ease of Use)* questionnaire [32] was applied in order to get the user's feedback and comments for to know about the difficults in the test. The USE questionnaire consists of 30 rating scales divided into 4 categories:

Table 6.1: Time on task data for 10 users and 13 tasks.

Task	Us1	Us2	Us3	Us4	Us5	Us6	Us7	Us8	Us9	Us10
1	12	28	24	30	19	33	23	16	5	7
2	3	4	17	5	17	134	9	16	12	11
3	123	69	159	53	69	113	44	41	70	98
4	20	4	86	40	13	4	17	3	20	3
5	50	10	63	50	7	11	10	5	20	Null
6	10	30	28	27	5	46	Null	7	Null	34
7	10	20	16	8	15	Null	9	24	16	28
8	18	24	10	10	5	3	27	4	5	6
9	5	6	31	4	45	9	12	5	3	8
10	15	17	15	11	10	19	13	10	20	20
11	30	15	20	16	20	22	15	13	18	20
12	12	14	19	14	40	10	17	17	15	15
13	25	15	15	14	10	10	11	10	10	25

Table 6.2: Confidence interval per task with a confidence level of 95%.

Task	Median	CI 95%	Upper bound	Lower bound
1	20	5.96	25.96	14.04
2	11.5	0.81	12.31	10.69
3	69.5	25.57	95.07	43.93
4	15	16.34	31.34	-1.34
5	15.5	14.84	30.34	0.66
6	27.5	11.57	39.07	15.93
7	16	5.19	21.19	10.81
8	8	5.80	13.80	2.20
9	7	9.43	16.43	-2.43
10	15	2.44	17.44	12.56
11	19	3.00	22.00	16.00
12	14.5	5.51	20.01	8.99
13	12.5	3.89	16.39	8.61

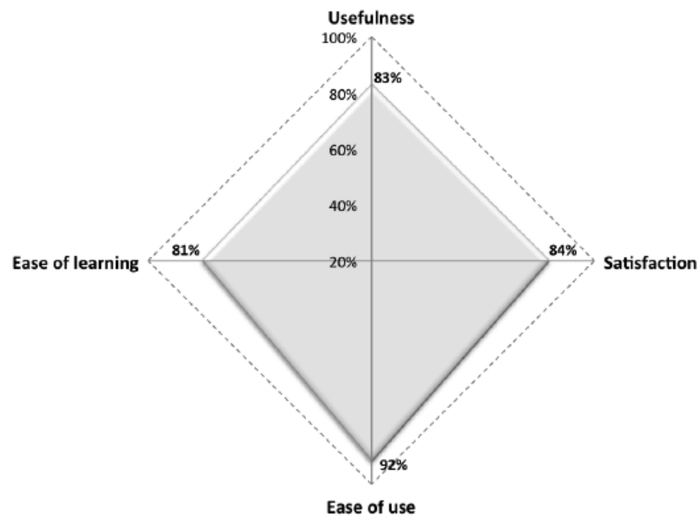


Figure 6.2: The radar chart that depicts the four axis evaluated in the questionnaire.

Usefulness, Satisfaction, Ease of Use, and Ease of Learning. Each is a positive statement to which the user rates level of agreement on a 7-point Likert scale. The USE questionnaire(see appendix B) allows to get values for Usefulness, Satisfaction, Ease of Use, and Ease of Learning, the visualizing the results is in the Fig.6.2 , where the four axis of the radar chart represent the values of percent which users rated positively this factors with respect to their interaction with the context-aware recommender system. The accurate values are *Usability 83%, Satisfaction 84%, Easy of use 92%, and Easy of Learning 81%.*

Chapter 7

Conclusions and future work

We observed the users behaviour to identify the most frecuently difficults and doubts about tasks. We did a brief interview with users after the test in order to understand their feelings or mood, their ideas about the experience, and overall, their opinion about the context-aware recommender system. The conclusions are based in user's comments, then the main errors in the system interface are summarized in three points:

1. Incomplete information for user, i.e., the system doesn't had enough and clear information to be a friendly interface, and therefore the user couldn't do easily a task.

2. Fails in design, because of unordered elements in the screen, in other words,

the elements are not in the correct site into the screen to be easily identified per users.

3. Fails in the language and confusion, because of the english language is not the native language of the users.

The three points mentioned are related to the null values in data table (see Table 6.1), some users didn't the task because they were confused, so they decided to omit the task. The null values weren't took in account when the median was calculated (see Table 6.2).

The USE questionnaire was useful to identify the weaknesses in the context-aware recommender system. The percent is upper of the acceptable (80%), the results allow to say that the system has a good performance.

For the future work we proposed to improve the problems found in the user interface, so the proposals are the following:

1. Redesign the user interface could helps to be more friendly for users. Due to the issues, the redesign involves:

   (a) Analyze the amount of information enough for a easy understanding, i.e., how much information the user needs seeing without overload it.

   (b) Modify the tasks descriptions in the most simple way to avoid confusion.

   (c) Add more language functionalities for to facilitate the tasks for users.

2. To apply the usability test again with the changes in the interface in order to observe the level of improves and to compare the results.

3. Apply an statistical test to analize the results.

4. Add collaborative filtering based on model (matrix factorization technique) within the context-aware recommender system in order to improve the level of user satisfaction in the context.

5. Add any contextual factors (such as companion, time of day, budget, etc.) in order to include more context information that could be relevant in the recommendations.

Publications

1. *Restaurant Recommendations based on a Domain Model and Fuzzy Rules. Xochilt Ramírez-García, Mario García-Valdéz. International Seminar on Computational Intelligence. Tijuana Institute of Technology. Tijuana Mexico. (2012).*

2. *Post-filtering for a Context-Aware Recommender System. Xochilt Ramírez-García, Mario García-Valdéz. Recent Advances on Hybrid Approaches for Designing Intelligent Systems . Springer International Publishing Switzerland. (2013).*

3. *Recomendaciones contextuales basadas en el enfoque de post-filtrado. Xochilt Ramírez-García, Mario García-Valdéz. Modelado computacional de Habilidades Linguisticas y Visuales. Vol.74. Research in Computer Sciences, IPN. 2014.*

4. *Context-aware Recommender System Based in Pre-filtering Approach and Fuzzy*

Rules. Xochilt Ramírez-García, Mario García-Valdéz. Recent Advances on Hybrid Approaches for Designing Intelligent Systems . Springer International Publishing Switzerland. (2014).

5. *Context-Aware Recommender System Using Collaborative Filtering, Content-Based Algorithm and Fuzzy Rules. Xochilt Ramírez-García, Mario García-Valdéz, 2016.*

6. *A Hybrid Context-aware Recommender System for Restaurants. Xochilt Ramírez-García, Mario García-Valdéz, 2016.*

Appendix A

Pseudocode

Algorithm 1 Get Cosine similarity values

Require: The list of itemProfilesUser and itemProfilesAll in binary format.
Ensure: The list of cosine similairty value for each item of the itemProfilesUser with each element of itemProfilesAll.

```
allProfiles ←[ ]
for itemu to size of itemProfilesUser do
    for itema to size of itemProfilesAll do
        if itemu = itema then
            jump next item
        else
            cosineSimilarityValue ← among itemu and itema
            itemProfiles ← itemu, itema, cosineSimilarityValue
        end if
    end for
end for
return allProfiles
```

Algorithm 2 Collaborative filtering algorithm

Require: The userId.
Ensure: The Top-N list of recommendations for the current user.

```
ratingMatrix ← allRatings
Call Recommendations ← getRecommendations() module
return Recommendations
```

Algorithm 3 Content-Based Algorithm

Require: The user id.
Ensure: The Top-N list of recommendations.

```
RV ← All items that user rated with 5
for item to size of RV do
   if item is not in RV then
      UV ← itemid
   end if
end for
allItems ← [ ]
getItemsProfilesUser ← Binary vectors of RV
allRatings ← Rating matrix
for item to size of allRatings do
   if itemid is not in allItems then
      allItems ← item
   end if
end for
getAllItemsProfiles ← Binary vectors of allItems
getCosineSim ← getItemsProfilesUser,getAllItemsProfiles
for item to size of highCosineSim do
   if itemsimilarity ≥ 0.8 then
      highCosineSim ← item
   end if
end for
Sort highCosineSim list
return itemProfiles
```

Algorithm 4 Get item profiles

Require: The UV vector, allItems vector and boolean value of userProfile.
Ensure: The list of temProfiles in binary vectors.
  if $userProfile$ **true then**
    $getItemsProfilesUser \leftarrow UV$
    for $itemp$ to size of UV **do**
      get binary vector of itemp
      $itemProfiles \leftarrow itemp$
    end for
  else
    $allItemProfiles \leftarrow allItems$
    for $itemp$ to size of $allItems$ **do**
      get binary vector of itemp
      $itemProfiles \leftarrow itemp$
    end for
  end if
  return $itemProfiles$

Algorithm 5 Calculate Cosine similarity

Require: The itemProfileUser and itemProfileAll, both vectors in binary format.
Ensure: The cosine similarity value.
  $sum \leftarrow 0$
  $normaItemUser \leftarrow 0$
  $normaItemAll \leftarrow 0$
  for $position$ to size of $itemProfileUser$ **do**
    $sumProduct \leftarrow sumProduct + (itemProfileUser[position] * itemProfileAll[position])$
  end for
  for $item$ to size of $itemProfileUser$ **do**
    $normaItemUser \leftarrow normaItemUser + itemProfileUser[item]^2$
  end for
  for $item$ to size of $itemProfileAll$ **do**
    $normaItemAll \leftarrow normaItemAll + itemProfileAll[item]^2$
  end for
  $squareRootUser \leftarrow squareroot(normaItemUser)$
  $squareRootAll \leftarrow squareroot(normaItemAll)$
  $cosineSimilarity \leftarrow sumProduct/(squareRootUser * squareRootAll)$
  return $cosineSimilarity$

Algorithm 6 Create a binary vector of item profile

Require: The tem profile content in r.
Ensure: The temProfile of r in a binary vector.

$price \leftarrow [4]$
$payment \leftarrow [2]$
$alcohol \leftarrow [2]$
$smokingarea \leftarrow [2]$
$dresscode \leftarrow [3]$
$parking \leftarrow [3]$
$installation \leftarrow [4]$
$atmosphere \leftarrow [5]$
$cuisine \leftarrow [30]$
$price[positionPriceId - 1] \leftarrow 1$
$payment[positionPriceId - 1] \leftarrow 1$
$alcohol[positionPriceId - 1] \leftarrow 1$
$smokingarea[positionPriceId - 1] \leftarrow 1$
$dresscode[positionPriceId - 1] \leftarrow 1$
$parking[positionPriceId - 1] \leftarrow 1$
$installation[positionPriceId - 1] \leftarrow 1$
$atmosphere[positionPriceId - 1] \leftarrow 1$
$cuisine[positionPriceId - 1] \leftarrow 1$
$itemProfile \leftarrow price + payment + alcohol + smookingarea + dresscode + parking + installation + atmosphere + cuisine$
return $itemProfile$

Algorithm 7 Get recommendations

Require: The currentUser and ratingMatrix.
Ensure: The Top-N list of recommendations for the current user.

```
Dictionaries totals ← {}, sumSimilarity ← {}
predictions ←[ ]
for otherUser to size of ratingMatrix do
    if otherUser = currentUser then
        jump next otherUser
    end if
    similarityValue ← get pearsonSimilarity
    if similarityValue ≤ 0 then
        jump next otherUser
    end if
    for item to size of profileOther do
        if item is not in profileUser then
            if profileUser[item] = 0 then
                Set in totals ← item
                totals[item] Add ratingMatrix[otherUser][item] * similarityValue
                Set in sunSimilarity ← item
                sumSimilarity Add similarityValue
            end if
        end if
    end for
end for
for each (item, total) in totals do
    predictions ← [(total/sumSimilarity[item], item)]
end for
Ranking of predictions
return predictions
```

Algorithm 8 Get Pearson correlation

Require: The currentUser, otherUser and preferences.
Ensure: The pearsonCorrelation score.

Dictionaries $itemsRatedMutually \leftarrow \{\}$
for each $item$ in preferences of $currentUser$ **do**
  if $item$ is in preferences of $currentUser$ **then**
    jump next $itemsRatedMutually[item] \leftarrow 1$
  end if
end for
$numberElements \leftarrow$ size of $itemsRatedMutually$
if $itemsRatedMutually = 0$ **then**
  return 0
end if
for $item$ to size of $itemsRatedManually$ to get all preferences **do**
  $sumCurrentUser \leftarrow preferences[currentUser][item]$
  $sumOtherUser \leftarrow preferences[otherUser][item]$
end for
for $item$ to size of $itemsRatedManually$ to get squares **do**
  $squareCurrentUser \leftarrow square(preferences[currentUser][item])^2$
  $squareOtherUser \leftarrow square(preferences[otherUser][item])^2$
end for
for $item$ to size of $itemsRatedManually$ to get sum of products **do**
  $sumProduct \leftarrow preferences[currentUser][item] * preferences[otherUser][item]$
end for
$pearsonNumerator \leftarrow sumProduct - ((sumCurrentUser * sumOtherUser)/numberElements)$
$pearsonDenominator \leftarrow square(squareCurrentUser - ((sumCurrentUser)^2/numberElements) * squareOtherUser - ((sumOtherUser)^2/numberElements))$
$pearsonCorrelation \leftarrow pearsonNumerator/pearsonDenominator$
return $pearsonCorrelation$ among two users

Algorithm 9 Matrix factorization

Require: R is a matrix to be factorized, dimension N * M, P an initial matrix of dimension N * K, Q an initial matrix of dimension M * K, K is the number of latent features, steps for the maximum number of steps to perform the optimization, alpha is the learning rate and beta is the regularization parameter.

Ensure: The factorized matrix P and Q.

$alpha \leftarrow 0.0001, beta \leftarrow 0.001$
$QMatrix \leftarrow QMatrix * T$
for $step$ to $rangeSteps$ **do**
  for i to size of $RMatrix$ **do**
    for j to size of $RMatrix[i]$ **do**
      if $RMatrix[i][j] > 0$ **then**
        $e_{i,j} \leftarrow RMatrix[i][j] - dotProduct(PMatrix[itoend], QMatrix[inittoj])$
      end if
      for k to range of $KFactors$ **do**
        $PMatrix[i][k] \leftarrow PMatrix[i][k] + alpha * (2 * e_{i,j} * QMatrix[k][j] - beta * PMatrix[i][k])$
        $QMatrix[k][j] \leftarrow QMatrix[k][j] + alpha * (2 * e_{i,j} * PMatrix[i][k] - beta * QMatrix[k][j])$
      end for
    end for
  end for
  $eR \leftarrow dotProduct(PMatrix * QMatrix)$
  for i to range of $RMatrix$ **do**
    for j to size of $RMatrix[i]$ **do**
      if $RMatrix[i][j] > 0$ **then**
        $e \leftarrow e + (beta/2) * PMatrix[i][k]^2 + QMatrix[i][j]^2$
      end if
    end for
  end for
  if $e < 0$ **then**
    break
  end if
end for
return $PMatrix, QMatrix * T$

Appendix B

USE Questionnaire

Usefulness

- It helps me be more effective.
- It helps me be more productive.
- It is useful.
- It gives me more control over the activities in my life.
- It makes the things I want to accomplish easier to get done.
- It saves me time when I use it.
- It meets my needs.
- It does everything I would expect it to do.

Ease of Use

- It is easy to use.
- It is simple to use.
- It is user friendly.
- It requires the fewest steps possible to accomplish what I want to do with it.
- It is flexible.
- Using it is effortless.
- I can use it without written instructions.
- I don't notice any inconsistencies as I use it.
- Both occasional and regular users would like it.
- I can recover from mistakes quickly and easily.
- I can use it successfully every time.

Ease of Learning

- I learned to use it quickly.
- I easily remember how to use it. It is easy to learn to use it.
- I quickly became skillful with it.

Satisfaction

- I am satisfied with it.
- I would recommend it to a friend.
- It is fun to use.
- It works the way I want it to work.
- It is wonderful.
- I feel I need to have it.
- It is pleasant to use.

Source: From the work of Lund (2001). Note: Users rate agreement with these statements on a 7-point Likert scale, ranging from strongly disagree to strongly agree. Statements in italics were found to weight less heavily than the others.

Bibliography

[1] Gregory D Abowd, Christopher G Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless networks*, 3(5):421–433, 1997.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[3] Wasfi Al-Khatib, Y Francis Day, Arif Ghafoor, and P Bruce Berra. Semantic modeling and knowledge representation in multimedia databases. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):64–80, 1999.

[4] William Albert and Thomas Tullis. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.

[5] Susan Auty. Consumer choice and segmentation in the restaurant industry. *Service Industries Journal*, 12(3):324–339, 1992.

[6] Robert Babuška. Fuzzy systems, modeling and identification. *Department of Electrical Engineering, Delft University of Technology*, 1996.

[7] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[8] Linas Baltrunas and Xavier Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS09)*, 2009.

[9] Linas Baltrunas and Francesco Ricci. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 245–248. ACM, 2009.

[10] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. Incarmusic: Context-aware music recommendations in a car. In *EC-Web*, volume 11, pages 89–100. Springer, 2011.

[11] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. Context-aware places of interest recommendations and explanations. In *Joint Proceedings of the Workshop on Decision Making and Recommendation Acceptance Issues in Recommender Systems (DEMRA 2011) and the 2nd Workshop on User Models for Motivational Systems: The Affective and the Rational Routes to Persuasion (UMMS 2011). CEUR Workshop Proceedings*, volume 740, pages 19–26, 2011.

[12] Ronald J Brachman, Hector J Levesque, and Raymond Reiter. *Knowledge representation*. MIT press, 1992.

[13] Rebecca Bulander, Michael Decker, Gunther Schiefer, and Bernhard Kölmel. Comparison of different approaches for mobile advertising. In *Mobile Commerce and Services, 2005. WMCS'05. The Second IEEE International Workshop on*, pages 174–182. IEEE, 2005.

[14] Robin Burke. Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce*, pages 69–72, 1999.

[15] Landro Castro and Silvana Aciar. Prototype of a tourism recommender system. In *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, pages 1–7. IEEE, 2012.

[16] Federica Cena, Luca Console, Cristina Gena, Anna Goy, Guido Levi, Sonia Modeo, and Ilaria Torre. Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Communications*, 19(4):369–384, 2006.

[17] Chung-Hua Chu and Se-Hsien Wu. A chinese restaurant recommendation system based on mobile context-aware services. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, volume 2, pages 116–118. IEEE, 2013.

[18] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and

Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer, 1999.

[19] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[20] Didier J Dubois. *Fuzzy sets and systems: theory and applications*, volume 144. Academic press, 1980.

[21] Brian Fling. *Mobile Design and Development: Practical concepts and techniques for creating mobile sites and web apps.* " O'Reilly Media, Inc.", 2009.

[22] Mario García-Valdez and Brunett Parra. A hybrid recommender system architecture for learning objects. In *Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control*, pages 205–211. Springer, 2009.

[23] Jennifer Golbeck, James Hendler, et al. Filmtrust: Movie recommendations using trust in web-based social networks. In *Proceedings of the IEEE Consumer communications and networking conference*, volume 96, pages 282–286. Citeseer, 2006.

[24] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2619–2625, 2013.

[25] Liang He and Faqing Wu. A time-context-based collaborative filtering algorithm. In *Granular Computing, 2009, GRC'09. IEEE International Conference on*, pages 209–213. IEEE, 2009.

[26] Jaksa Jack Kivela. Restaurant marketing: selection and segmentation in hong kong. *International Journal of Contemporary Hospitality Management*, 9(3):116–123, 1997.

[27] Amit Konar. *Computational intelligence: principles, techniques and applications.* Springer Science & Business Media, 2006.

[28] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[29] Brian Y Lim and Anind K Dey. Assessing demand for intelligibility in context-aware applications. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 195–204. ACM, 2009.

[30] Matteo Manca, Ludovico Boratto, and Salvatore Carta. Mining user behavior in a social bookmarking system-a delicious friend recommender system. In *DATA*, pages 331–338, 2014.

[31] Luis Martinez, Rosa M Rodriguez, and Macarena Espinilla. Reja: A georeferenced hybrid recommender system for restaurants. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*, pages 187–190. IEEE Computer Society, 2009.

[32] Margaret Morris and Arnie Lund. Experience modeling: How are they made and what do they offer. *LOOP: AIGA Journal of Interaction Design Education*, (7), 2001.

[33] José M Noguera, Manuel J Barranco, Rafael J Segura, and Luis Martínez. A mobile 3d-gis hybrid recommender system for tourism. *Information Sciences*, 215:37–52, 2012.

[34] Bing Pan and Daniel R Fesenmaier. Online information search: vacation planning process. *Annals of Tourism Research*, 33(3):809–832, 2006.

[35] Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[36] Sasank Reddy and Jeff Mascia. Lifetrak: music in tune with your life. In *Proceedings of the 1st ACM international workshop on Human-centered multimedia*, pages 25–34. ACM, 2006.

[37] Norma Saiph Savage, Maciej Baranski, Norma Elva Chavez, and Tobias Höllerer. *Im feeling loco: A location based context aware recommendation system*. Springer, 2012.

[38] Rossano Schifanella, André Panisson, Cristina Gena, and Giancarlo Ruffo. Mobhinter: epidemic collaborative filtering and self-organization in mobile ad-hoc networks. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 27–34. ACM, 2008.

[39] William Siler and James J Buckley. *Fuzzy expert systems and fuzzy reasoning.* John Wiley & Sons, 2005.

[40] Thomas Tran and Robin Cohen. Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press*, 2000.

[41] Gytis Tumas and Francesco Ricci. Personalized mobile city transport advisory system. *Information and Communication Technologies in Tourism 2009*, pages 173–183, 2009.

[42] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on*, 10(5):293–302, 2002.

[43] Ramírez-García Xochilt and Mario García-Valdez. Restaurant recommendations based on a domain model and fuzzy rules. In *Recent Advances on Hybrid Intelligent Systems*, pages 533–546. Springer, 2013.

[44] Jiawei Yao, Jiajun Yao, Rui Yang, and Zhenyu Chen. Product recommendation based on search keywords. In *Web Information Systems and Applications Conference (WISA), 2012 Ninth*, pages 67–70. IEEE, 2012.

[45] Chien-Chih Yu and Hsiao-Ping Chang. *Personalized location-based recommendation services for tour planning in mobile tourism applications.* Springer, 2009.

[46] LA Zedeh. Knowledge representation in fuzzy logic. *Knowledge and Data Engineering, IEEE Transactions on*, 1(1):89–100, 1989.

[47] Yong Zheng. Context-aware datasets. url: http://students.depaul.edu/ yzheng8/datasets.html. 2015. URL http://students.depaul.edu/~yzheng8/DataSets.html.

[48] Yong Zheng, Robin Burke, and Bamshad Mobasher. *Differential context relaxation for context-aware travel recommendation.* Springer, 2012.

[49] Yong Zheng, Bamshad Mobasher, and Robin Burke. Context recommendation using multi-label classification. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on*, volume 2, pages 288–295. IEEE, 2014.