# Acknowledgments

*This work is for everybody that trust in my effort and dedication to do everything that I propose.*

*I want to thank specially my sisters Rosy, Nely and Mary and of course my parents, they are the pillar that keeps me on the way.*

*I want to thank my friends: Maribel, Cinthya, Sam and Lorenzo, they are so special for me and they are always there such as my partners in my way until the end. Im lucky in friendship.*

*I want to acknowledge the effort and patience of Dr. Mario García Valdéz for four years. I appreciate the shared knowledge and the instructions about the research. Nowadays I got experience and tools to face up a new phase in my life. Thank you!.*

*Finally, I would like to express my gratitude to CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this thesis.*

# Resumen

Un sistema de recomendación sensible al contexto analiza las necesidades y preferencias de los usuarios con el propósito de recomendar items utilizando la información contextual que describe la situación actual del usuario. En esta tesis se propone un método para sistemas de recomendación sensible al contexto que puede ser aplicado en diferentes dominios para mejorar las recomendaciones utilizando la información contextual en un método de recomendación híbrido. El método involucra diferentes técnicas que trabajan simultáneamente para obtener las recomendaciones: filtrado colaborativo, basado en contenido y un Sistema de Inferencia Difuso para procesar reglas y atributos difusos. Posteriormente, las recomendaciones son filtradas por los factores de contexto que representan la situación actual del usuario. En esta tesis se analiza el trabajo relacionado y destaca las principales contribuciones del método, presentando resultados de un extensivo conjunto de experimentos que validan el método propuesto.

# Abstract

The context-aware recommender system analyzes the needs and preferences of users in order to recommend items using contextual information that describes the current situation of the user. In this thesis a method for context-aware recommender system was proposed, this method can be applied in different domains to improve the recommendations using contextual information in a hybrid recommender method. The proposed method involves several techniques that working simultaneously to get recommendations: collaborative filtering, content-based and Fuzzy Inference System to process rules and fuzzy attributes. Subsequently, recommendations are filtered by contextual factors that represent the current situation of the user. This thesis presents an analysis of related works and highlight the main contributions of the method, presenting results of an extensive set of experiments that validate the proposed method.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

The purpose of this research is to contribute in the knowledge of the use of context in recommender systems and to propose a method for recommendations. The method involves recommendation techniques and a fuzzy inference system, both techniques make recommendations. The goals were achieved by means of the collection of contextual information through questionnaires and several experiments to validate the method.

## 1.1 Motivation

Often people need take desitions, even although their experience is not enought to decide among the possible alternatives. Overall, persons trust in recommendations

of someone else, to taking desitions according their personal interest.

There is a large amount of information generated in a society, so, the lack of experience of persons highlights the importance to provide automatic methods to filter relevant information that helps to taking desitions.

Every day, a lot of information is generated in Internet and it is available for users, then the overload information problem arises, in a manner that users can't identify the relevant information of the irrelevant. The users need tools to facilitate tasks, as well as tools to provide a recommendation service that helps to use efficiently the available information. The overload of information and the lack of experience of users promotes the need of automatic search engines that contributes to taking desitions through recommendations of personalized products or services.

By other hand, in recent years the mobile computing rocketed its importance because of the impact of its use in daily life, any application can be applied in mobiles and anywhere it can be used, their limitations are every time less. For this reason, new technologies for use of context in mobile applications arise, intelligent systems can take advantage of the benefits that technology provides to manage the context that changing constantly. Mobile and ubiquitous computing[49] [23] are proposing a wide variety of applications that need recommendation engines using context for meet their purposes.

## 1.2 Context-awareness

Traditional recommender systems provides suggestions of useful items for a certain user. The suggestion relates to various decision-making processes, for instance, what items to buy, what music to listen to or what on-line news to read. *Item* is the general term to denote what product or service the system recommends for each user. A recommender system normally focuses only in a type of item [57]. The improvements of previous recommender systems are focused in the *integration of context* in its recommendation process. The idea of *context-aware computing* is to provide information or services for the user based in the user's situation [26]. In order to do that, the application needs to obtain situational data, process it and make use of it in a manner that benefits the user.

*Context* is a concept not easy to define, it is related with several disciplines that propose different definitions. For example, the authors Bazire et.al.[15] compare the context in different fields and conclude that is complicated makes a unifying definition of context because of the nature of the concept in the disciplines. In computer sciences Fischer G.[30] defines context as the interaction between humans and computers in socio-technical systems that takes place in a certain context referring to the *physical* and *social situation* in which computational devices and environments are embedded. Also identifies the important aspects to consider when the context

is used: how it is the contextual data obtained, how the context is represented and what goals and purposes the context has when is used in a particular application. The definition most used in the field of recommender systems to define *context* is proposed by Dey[26]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."* This definition makes it easier to define the contextual factors in a specific application. For instance, in a tourist guide application the entities can be companion(friends, family, couple), place of interest, season and weather, these could be considered as relevant contextual factors that help the recommender system to provide items adjusted the situational data of the user.

*Context-aware recommender systems* are gaining even more attention because of their performance and implementation for different domains, the way to improve personalized recommendations based in contextual factors is an important technique to increase the benefits in many domains. For instance, taking in account the *hour of the day*, or the *day of the week* when recommending restaurants could filter out restaurants that are currently closed or near closing time, when the user receives this information in real time, the user has the way for taking alternatives of restaurants that provide services. Nowadays, many companies are incorporating some type of context (as time, location or companion) in their recommendation engines,

the application can be found in fields such as e-commerce[60] [18], music[58] [12] [38], places of interest[14], movies[29], vacation packages[44] [45], travel guides[59], e-learning[50] and restaurants[24].

Plus, context can be used to improve the user satisfaction in recommender systems, thus the quality and accuracy of predictions is improved too.

The proposed method uses three recommendations techniques:

1. *Fuzzy Inference System*, this a rule based recommender defined by an expert in the domain, it considers the following variables: *ratings average: low,medium and high*, *price of restaurant: cheap, average and expensive*, and *number of ratings of item: few, several and many*, these variables are used to infer how relevant a restaurant is for the user. This recommendation is based on the popularity of each item in the user community.

2. *Content-based technique* utilizes the item profiles to compare how *similar* is an item with respect to another, i.e. restaurants that are *similar* (same cuisine, ambient, price range) to others that the user has rated high. The idea is to find items with similar features.

3. *Collaborative filtering technique* is based on the user profile to identify user's preferences and to find neighbors that have the same tastes. The recommendation consist in the suggestions of other users with similar tastes that rated

restaurants again in a similar way but where have not been rated by the current user. A Top-N list of restaurants is obtained to recommend for the user.

The results of the three techniques are a list of recommendations for the user, later, these recommendations are adjusted in context. This is the last step and is represented as *context filter* in the method, then the recommender system obtains a list of contextualized recommendations.

In the method, each technique works simultaneously to obtain recommendations, the hybrid method allows to generate suggestions even without user information, i.e., using content-based technique or the fuzzy inference system, so the system faces the cold-start or the overspecialization problem using these thechniques. These problems are described in a later section.

To test the performance were made several experiments that validate the method, the algorithms were tested using contextual datasets and the number of contextual factors used varies according the information provided in the dataset. The goals of the experiments was to observe the context role in the performance of the algorithms, what contextual factors matter for users in a specific situation, how recommendations are improved using context and, the accuracy in recommendations. Chapter 5 shows the results, discussion about results is explained in each section.

Another important metric is the *user satisfaction*, two metrics were used to measure it: *task-success* and *time-on-task*. These metrics allow to measure the user experi-

ence, this offers so much more than just simple observation.

Usability metrics can help reveal patterns that are hard or even impossible to see. Evaluating software with a small sample size usually reveals the most obvious usability problems[6].

Then, as a general rule of thumb, during the early stages of design, it needs fewer participants to identify the major usability issues. As the design gets closer to completion, the tests should include more participants to identify the remaining issues[6].

Following this precept, ten representative users were selected to test the system, subsequently, it was realized an analysis about the system performance and issues presented in the user interaction. The chapter 6 explains the process to evaluate the system and the results obtained.

## 1.3 Aims

The contribution is to propose a method of context-aware recommender systems using different techniques of recommendation, another aim is to provide a *useful knowledge* to utilize context-aware recommender systems using a hybrid method that is easily implemented in different domains such as e-learning, movies, music, tourism, etc. In this particular case, the restaurants domain is used as a case of

study to test the method.

Another important contribution is the use of fuzzy rules in the proposed method, this allows the use of linguistic information closer to the real context of the people, i.e., the method uses this technique to analyze the user preferences and get recommendations based in that information. For instance, the system provides a list of range of prices, this allows the user to select an specific range of price to get recommendations adjusted for the tag selected.

In order to support the achievement of contributions, the particular aims of this thesis are following:

- To elaborate an analysis about state of the art in the field of context-aware recommender systems through the revision of literature.

- To select the algorithms that represent alternatives of solution for the problem in order to test their perfomance in different domains.

- Realize experiments with the proposed algorithms.

- Based in previous experiments and results, to propose a hybrid method and apply it in a case of study.

- To define the fuzzy inference system that serves as recommender technique in the proposed method, as well as the variables and fuzzy rules involved.

- To develop a prototype of context-aware recommender system using the proposed method.
- To select the algorithms that represent alternatives of solution for the problem to evaluate their perfomance in different domains.
- To evaluate the performance of the algorithms using different datasets in order to observe the system behaviour for each particular case.
- To propose a hybrid method for context applied in a case of study.
- To develop a prototype of context-aware recommender system using the method in a specific domain.

## 1.4 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** describes an in-depth study of current background and related work is presented to give a general overview of recommender systems and their evolution in recent years. This study includes the traditional recommender systems, their methods and techniques to improve recommendations, as so the problems to face up this systems. Subsequently, the hybrid methods used in different applications, their strenghts and weaknesses for each hibridation and

the domains of application. Finally, context-aware recommender systems are mentioned, in the same way, we speak about the advantages and disadvantages of the use of context in recommender systems.

- **Chapter 3** describes the fundamental concepts required to understand the proposed method.

- **Chapter 4** presents a model of context-aware recommender system, the proposed method involves the paradigm of post-filtering in a restaurants domain. This chapter include the overall explanation of data models and the method functionality, as well as its components for this case of study.

- **Chapter 5**, the general results of different projects involved are presented along with the validation of every experimentation. The experiments were realized using different datasets and different algorithms in order to find an optimal manner to reduce the error level. This chapter also details the results for each experiment from a point of view of scientific results.

- **Chapter 6**, after the development of context-aware recommender system, it was evaluated the impact of context in recommendation process. This chapter describes the usability tests that were applied on-line in order to evaluate the satisfaction of users. Details of the environment and the characteristics of the tests are described, as well as the results of each one.

- **Chapter 7**, this chapter concludes with a summary of its contributions and its limitations. It discuss final conclusions and the proposals for the future work.

At the end, this thesis includes several appendices describing detailed technical aspects of the context-aware recommender system *(appendix C)*, the pseudocode of algorithms *(appendix A)*, interfaces of the prototype of context-aware recommender system *(appendix D)* and experiment study materials *(appendix B)*.

# Chapter 2

# State of the art

Recommender system is a technology used by many authors for personalization of information. The amount of information in Internet is rising quickly and the users every time has less capability to search the most relevant information among big databases. Many recommender system are used in several domains. The fundamental is the profit that provides in many of these domains that sometimes represents the key of success for the application. Some works utilize social information to recommend such as Manca et.al.[46] where the friend recommender system is applied in the social bookmarking domain, its goal was to infer the interest of users from content selecting the available information of the user behavior and analyzing the resources and the tags bookmarked for each user, therefore the recommendations are through mining user behavior in a tagging system, analyzing the bookmarks

tagged of the user and the frecuency for each used tag. J.Yao et.al.[68] proposes a new product recommendation approach for new users based on the implicit relationships between search keywords and products. The relationships between keywords and products are represented in a graph and relevance of keywords to products is derived from attributes of the graph. The relevance information is utilized to predict preferences of new users. J. Golbeck et.al.[33] presents FilmTrust, a website that integrates Semantic Web-based social networks, augmented with trust, to create prediction movie recommendations. Trust takes on the role of a recommender system forming the core of an algorithm to predict a rating for recommendations of movies. This is an example of how the Semantic Web, and Semantic trust networks in particular, can be exploited to refine the user experience.

Traditional recommender techniques has its pros and cons, for instance, the ability to handle data sparsity and cold-start problems or considerable ramp-up efforts for knowledge acquisition and engineering. Establish hybrid systems that combine the strengths of algorithms and models to overcome some of the shortcomings and problems has become the properly manner to improve the difficults for each algorithm. An example is presented by L.Castro et.al.[21] a hybrid recommender system for the province of San Juan, Argentina, to recommend tourist packages based on preferences and interest of each user, artificial intelligence techniques are used to filter and customize the information. The prototype of recommender system utilizes three

techniques to recommend: demographic, collaborative and content-based. The goal is to recommend tourist packages that matches with the user profile. L. Martinez et al.[47] presents REJA, a hybrid recommender system that involves collaborative filtering and knowledge-based model, that is able to provide recommendations in some situation for user; besides it provides georeferenced information about the recommended restaurants. Balabanovic et.al.[9] presents Fab, a hybrid recommender system for automatic recognition of emergent issues relevant to various groups of users. It also enables two scaling problems, pertaining to the rising number of users and documents, to be addressed. Claypool et.al.[25] presents P-Tango system that utilizes content-based and collaborative filtering techniques, it makes a prediction through the weighted average that includes content-based prediction and collaborative filtering prediction. The weights of predictions are determined on a per-user basis, allowing the system to determine the optimum mixture of content-based and collaborative recommendation for each user. Pazzani M.[53] presents Entree as a hybrid recommender system that it does not use numeric scores, but rather treats the output of each recommender (collaborative, content-based and demographic) as a set of votes, which are then combined in a consensus scheme. The recommender system includes information such as the content of the page, ratings of users and demographic data about users. Others works with hybrid recommender systems are ProfBuilder [5], PickAFlick[20] and [64], where are presented multiple recommenda-

tion techniques. Usually, recommendation requires ranking of items or selection of a single best recommendation, at this point some technique must be employed to recommend.

Traditional recommender systems such as above mentioned, tend to use simple user models. For example, user-based collaborative filtering generally models the user as a vector of item ratings. As additional observations are made about users preferences, the user models are extended, and the user preferences is used to generate recommendations. This approach, therefore, ignores the notion of any specific situation, the fact that users interact with the system within a particular context and that preferences of items might change in another context. Overall, the context is able to make the recommender system be powerful that is adaptable to the changing user's situation.

The context is defined in the domain of the application and the system has a context model that provides the information for the recommender system. For instance Ricci et.al. [12] uses the context in music domain using a model-based paradigm, in this context-aware recommender system the context was defined as a set of independent contextual factors(independent in order to get a mathematical model) such as *driving style, road type, landscape, sleepiness, traffic conditions, mood weather and natural phenomena* to specifies the relevant context for the music recommendation. In order to estimate the relevance of selected contextual factor, the users were re-

quested to evaluate music tracks in different contextual situations for each genre. The prediction takes in account this relevance to recommend music tracks prefered by the user according the genre and the contexts mentioned.In restaurant domain Chung-Hua et al.[24] presents a context-aware recommender system for mobiles using a post-filtering paradigm, the architecture involves a model client-server that works with a request of data in the client side for the server side. Subsequently, taking in account the contextual factors to filter the properly restaurants to recommend. The context-aware recommender system uses such as contextual factors *location and season*, also utilize the user preferences to personalize the recommendations in the user context. Baltrunas et.al.[13] presents ReRex for tourism, a context-aware recommender system based in a model-based paradigm, the system recommends and provides explanations about the why the places of interest(PoI) are recommended. The proposed model computes a personalized context dependent rating estimation. Subsequently, in order to generates the explanation of recommendation the system uses the factor that in the predictive model has the lasgest positive effect on the rating prediction for the point of interest. The set of contextual factors considered in ReRex are *distance*, temperature, weather, season, companion, time day, weekday, crowdedness, familiarity, mood, budget, travel length, transport and travel goal. The main issue in ReRex system is the low user satisfaction because of the explanations not able to be understood, however the users recognize that the explanation is

a very important component that it influence the system acceptance. Noguera et. al. [49] presents a context-aware recommender system for tourism based in REJA that utilizes the location through a 3D-GIS system, the application uses progressive downloading and rendering of 3D maps over mobiles networks. It is also in charge of tracking the users location and speed based on GPS and the requesting. The system utilizes pre-filtering and post-filtering paradigm. Pre-filtering is used to reduce the number of items considered for the recommendation according to the users location, and post-filtering is applied to re-rank the previous top-N list according to the physical distance from the user for each recommended restaurant. The disadvantage in this system is the lack of user reviews, because the recommendations are based only in the location point without consider the experience of other diners concerning the recommended restaurant. Cena et al.[22] presents a tourist guide for context in intelligent content adaptation. UbiquiTO system is a tourist guide that integrates different forms of context-related adaptation: for media device type, for user characteristics and preferences, for the physical context of the interaction. UbiquiTO uses a rule-based modeling approach to adapt the content of the provided recommendation, such as the amount, type of information and features associated with each recommendation. Bulander et.al[19] presents the MoMa-system that offers proactive recommendations using a post-filtering approach for matching order specifications with offers. When creating an order, the client application will automatically fill in

the appropriate physical context and profile parameters, for example, *location* and *weather*, then, for example, the facility should not be too far away from the current location of the user and beer should not be recommended if it is raining. On the other side, advertisers suppliers put offers into the MoMa-system. These offers are also formulated according to the catalogue. When the system detects a pair of context matching order and offer, the end user is notified, in the preferred manner (for example, SMS, email). At this point, the user must decide whether to contact the advertiser to accept the offer. Finally, Schifanella et al.[61] develops Mob-Hinter, a *context-dependent* distributed model,where a user device can directly connect to other mobile devices that are in *physical proximity* through ad-hoc connections, hence relying on a very limited portion of the users community and just on a subset of all available data (pre-filtering). The relationships between users are modeled with a similarity graph. MobHinter allows a mobile device to identify the affinity network neighbors from random ad-hoc communications. The collected information is then used to incrementally refine locally calculated predictions, with no need of interacting with a remote server or accessing the Internet. The Recommendations are computed using the availables rating of the user neighbors. Abowd et. al. presents Cyberguide project [1], which encompassed several tour guide prototypes for different handheld platforms. Cyberguide provided tour guide services to mobile users, exploiting the contextual knowledge of the users current and past locations in the

recommendation process. The PECITAS system [65] presented by Thumas offers location-aware recommendations for personalized point-to-point paths. The paths are illustrated by listing the various connections that the user must take to reach the destination using public transportation and walking. An interesting aspect of PECITAS is that, although an optimal shortest path facility is incorporated, users may be recommended alternative routes that pass through several attractions, given that their specified constraints (e.g. latest arrival time) and travel-related preferences (maximum walking time, maximal number of transport transfers, sightseeing preferences, etc) are satisfied. Yu and Chang presents LARS [69] which supports personalized tour planning using a rule-based recommendation process. This system packages where to stay and where to eat features together with typical tourist recommendations for sightseeing and activities. For instance, recommended restaurants (selected based on their location, menu, prices, customer rating score, etc) are integral part of the tour and the time spent for lunch/dinner is taken into account to schedule visits to attractions or to plan other activities. Savage et. al. presents "I'm feeling LoCo" system [59] that proposes a ubiquitous location based recommendation algorithm that focuses on user experience by considering user preferences, time, location and similarity measures automatically, having Foursquare as a dataset. We also focus on user experience and aim that user input is minimal. The information om the user's social network, form of transportation and phone's sensors is inferred

to provide recommendation of places om the dataset. Reddy et.al[56] presents Life-Track system that incorporates sensor information into song selection. The songs are represented in terms of tags that the user assigns in order to link the songs to the appropriate contexts in which they should be played. User feedback is incorporated to make a song more or less likely to play in a given context. Context considered relevant to song selection includes location, time of operation, velocity of the user, weather, traffic and sound. User locations and velocity are determined by GPS. Location information includes tags based on zip code and whether the user is inside or outside (inferred by the presence or absence of a GPS signal). The times of the day are divided out into configurable parts of the day (morning, evening, etc). The velocity is abstracted into one of four states: static, walking, running and driving. Use of accelerometers are planned to enable indoor velocity information. If the user is driving, an RSS feed on traffic information is used to typify the state as calm, moderate or chaotic. If the user is not driving, a microphone reading is used for the same purpose. Additionally, an RSS feed provides a meteorological condition (frigid, cold, temperate, warm or hot).

The table 2.1 describes examples of contextual factors in different domains of application, specifies the contextual factors considered such as part of the context, the methodology for each application and kind of devices.

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

## 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 [17] [41]:

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 [17]:

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 [41]. 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) [63]. Depending on the form of the consequent, two main types of fuzzy production systems are distinguished [8]:

- **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 [8] 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[8]:

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 [28]:

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 mem-

bership 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

Persons transmit ideas to each other and reacting appropriately. This is due to factors as: the richness of the language shared, the common understanding of how the world works, and an implicit understanding of situations in daily life. When persons talk, are able to use implicit situational information(contextual information), to increase the conversational bandwidth. Unfortunately, this ability to transmit ideas does not transfer well to persons interacting with computers. In traditional interactive computing, users have an poor mechanism for providing input to computers. Consequently, computers are not currently enabled to take full advantage of the context of the human-computer dialogue. By improving the computer's access to context, we increase the richness of communication in human-computer interaction and make it possible to produce useful computational services.

In order to use context effectively, we must understand what context is and how it can be used. An understanding of context will enable application designers to choose what context to use in their applications. An understanding of how context can be used will help application designers to determine what context-aware behaviours to use in the applications[26].

To stablish a specific definition of *context* that can be used in the *context-aware* computing field, is necessary to review how researchers define the context in their own work. Schilit and Theimer[2] refer to context as *location*, identities of nearby people and objects, and changes to those objects.

These types of definitions that define context by example are difficult to apply when it want to determine whether a type of information not listed in the definition is context or not, it is not clear how it can use the definition.

Schilit et al.[62] affirm that the important aspects of context are: where you are, who you are with, and what resources are nearby. Pascoe defines context to be the subset of physical and conceptual states of interest to a particular entity.

These definitions are too specific, context is all about the whole situation relevant to an application and its set of users. It is complicated enumerate which aspects of all situations are important, as this will change from situation to situation. For this reason and for purposes of this thesis, is adopted the definition of context proposed by Dey[26] (see section 1.2).

However, another important aspect is to stablish a meaningful classification that covers the characteristics that describe the contextual factor. Dourish[27] has distinguished between two different views of context: the *representational view* and the *interactional view*. The *representational view* makes four key assumptions: context is a *form of information*, it is *delineable*, it is *stable*, and it is *independent* from the underlying activity. In this view, context can be described using a set of observable attributes that are known a priori. Furthermore, the structure of these contextual attributes does not change over time. The *interactional view*, takes a different stance on the key assumptions made by the representational view. In the interactional view, the scope of contextual features is defined dynamically, and it is occasioned rather than static. Rather than assuming that context acts as a set of conditions under which an activity occurs, this view assumes a cyclical relationship between context and activity, where the activity gives rise to context and the context influences activity.

Context should include information to allow systems to use contextual information about users and their situation, then the system providing users the personalised and contextual services. The importance of context lies in the assumption of the influence of *contextual factors* that matter for users when they decide choose or discard an item.

In the real world, the context in a situation is involved in the *environment* of the

person, the *entities* belong at the *situational information*, but an entity becomes in a *contextual factor* when its information *affects* the recommendation process, therefore, the entity and its values of domain will be involved in the process such as a contextual factor.

The domain values of a contextual factor changes over time, in the real life the situation occurs when we decide that, for instance, we like a kind of clothes and the next day, for a any reason we don't like. As for the *"time of change"* representation, a data model of *time* should be specified in a manner that the system *interprets* the time in a data structure (for instance weeks, days, hours, minutes, seconds, etc.).

Assuming the existence of certain contextual factors such as *time*, *location* and *purchasing purpose* that are identified in the context of recommendations, Adomavicious[4] proposes two important aspects that highligh when different kinds of context are defined: *what a recommender system may knows about these contextual factors* and, *how contextual factors change over time.*

**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), Adomavicious categorizes the knowledge of a recommender system

about the context as following:

- **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 instance, *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 ShoppingCcompanion, but not their structure. Note that there can possibly be different levels of *"partial observability"*.

- **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.

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

- **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 instance, new categories can be added to the *PurchasingPurpose* contextual factor over time.

By other hand, Fling[31] generalizes four types of contexts that can be used in 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 user's goals, mood, experience, and cognitive capabilities.

Subsequently the revision of the literature of context, it is important to mention a formal definition that describes what features it has a context-aware system, this definition is proposed by Dey[26]: *"a system is context-aware if it uses context to provide relevant information and/or services for the user, where relevancy depends on the users task."*

This definition is closer to the reality about behaviour of *context-aware recommender system* when incorporates contextual information.

Based in this definition, A.K. Dey proposes some characteristics that a context-aware application should be support:

- **Presentation of information** and services to a user.

- **Automatic execution** of a service for a user.

- **Tagging of context** to information to support later retrieval.

An example to explain the context in a context-aware application, for instance, it can be 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 user's situation.

## 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[40]. 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 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 collaborative filtering 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 that only needs the rating matrix 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:

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

Where the * symbol is the dot product of vectors and $|a|$ 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 off-line, 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*.

### Data sparsity and cold-start problem

In real-world applications, the ratings matrix tend to be *very sparse*(sparcity problem), as customers typically provide ratings for (or have bought) only a small fraction of the catalog items. In general, the challenge in that context is thus to compute good predictions when there are relatively few ratings available. One straightforward option for dealing with this problem is to exploit additional information about the users, such as gender, age, education, interests, or other available information that can help to classify the user. The set of similar users (neighbors) is thus based not only on the analysis of the explicit and implicit ratings, but also on information external to the ratings matrix. These hybrid systems [54], however, no longer *purely* collaborative, and new questions of how to acquire the additional information and how to combine the different classifiers arise. Still, to reach the critical mass of users needed in a collaborative approach, such techniques might be helpful in the *ramp-up phase* of a newly installed recommendation service.

The *cold-start problem* can be viewed as a special case of the sparsity [37]. The questions here are (a)*how to make recommendations to new users that have not rated any item yet* and (b)*how to deal with items that have not been rated or bought yet.*

Both problems can be addressed with the help of hybrid approaches [3]. To face the *new-users problem*, one option could be to ask the user for a minimum number of ratings before the service can be used. In this situation the system could intelligently ask for ratings for items that, from the view point of information theory, carry the most information[55]. A similar strategy of asking the user for a gauge set of ratings is used for the Eigentaste algorithm presented in [34].

### 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, i.e., 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 [40].

### 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 [40].

### 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 [40].

### Overspecialization and cold-start problem

Learning-based methods quickly tend to propose more of the same, that is, such recommenders can propose only items that are somehow similar to the ones the current user has already (positively) rated. This can lead to the undesirable effect that obvious recommendations are made and the system, for instance, recommends items that are *too similar to those the user already knows*.

A typical example is a news filtering recommender that proposes a newspaper article that covers the same story that the user has already seen *in another context*. The system in [16] defines a threshold to filter out not only items that are *too different* from the profile but also those that are *too similar*. A general goal to face the *overspecialization*, therefore is to increase the serendipity of the recommendation lists that includes unexpected items in which the user might be interested, because expected items are of little value for the user.

The *cold-start problem*, which we discussed for collaborative systems, also exists in a slightly different form for content-based recommendation methods. Although content-based techniques do not require a large user community, they require at least an initial set of ratings from the user, typically a set of explicit like and dislike statements.

In all described filtering techniques, recommendation accuracy improves with the

number of ratings; significant performance increases for the learning algorithms were reported in [52] when the number of ratings was between twenty and fifty. However, in many domains, users might not be willing to rate many items before the recommender service can be used. In the initial phase, it could be an option to ask the user to provide a list of keywords, either by selecting from a list of topics or by entering free-text input.

### 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 [40].

### 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 system's task is to predict that user's rating or that item, reflecting the degree of user's preference for that item[40].

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 [43].

### 3.3.5 Paradigms for using of contextual information

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 [4] defines three paradigms for the context-aware recommendation process that is based on contextual user preference:

- **Contextual pre-filtering (or contextualization of recommendation input).** The approach uses contextual information to select the most relevant 2D (Users x Items) data for generating recommendations. One major advantage of this approach is that it allows deployment of any of the numerous traditional recommendation techniques previously proposed in the literature [3]. In particular, when using this approach, context $c$ essentially serves as a query (or a filter) for selecting relevant rating data. An example of a contextual data filter for a movie recommender system would be: if a person wants to see a movie on Saturday, only the Saturday rating data is used to recommend movies. Note that this example represents an exact pre-filter because the data

was filtered using exactly the specified context (figure 3.1.a).

- **Contextual post-filtering (or contextualization of recommendation output).** In this approach ignores context information in the input data when generating recommendations, that is, when generating the ranked list of all candidate items from which any number of *top-N* recommendations can be made. Instead, the contextual post-filtering approach uses contextual information to adjust the obtained recommendation list for each user. The recommendation list adjustments can be made by: (1) filtering out recommendations that are irrelevant in a given context, or (2) adjusting the ranking of recommendations in the list. For example, in a movie recommendation application, if a person wants to see a movie on a weekend, and on weekends he or she only watches comedies, the system can filter out all noncomedies from the recommended list (figure 3.1.b).

- **Contextual modeling (or contextualization of recommendation function).** This approach uses contextual information directly in the recommendation function as an explicit predictor of a user's rating for an item and, thus, gives rise to truly multidimensional recommendation functions representing either predictive models (such as decision trees, regressions, and so on) or heuristic calculations that incorporate contextual information in addition to

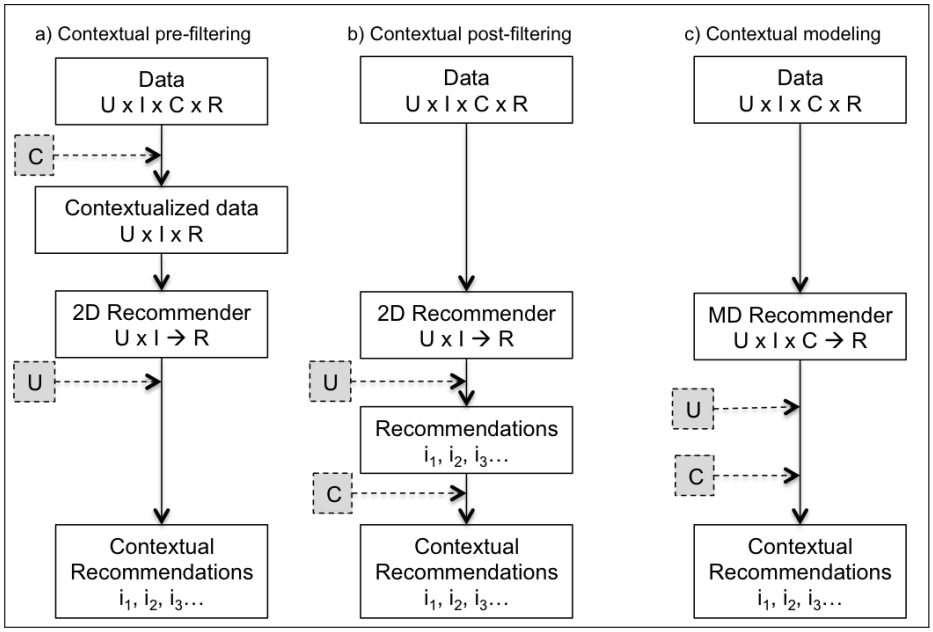



Figure 3.1: Paradigms for incorporating context in recommender systems[4].

the user and item data (figure 3.1.c).

# Chapter 4

# Proposed method

## 4.1 Data models

The data models was implemented in PostgreSQL database, the information in context-aware recommender system was managed in a scheme of a relational database. Technical support about installations of dependencies and the application are mentioned in appendix **??**.

### 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 consumer's preferences[4]. Pan and Fesenmaier[51] 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 consumer's desitions making[7]. 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[39] identifies four types of restaurants: 1) *fine dining/gourmet*, 2) *theme/atmosphere*, 3) *family/popular*, and 4) *convenience/fast-food*; and Auty[7] identifies four 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*,

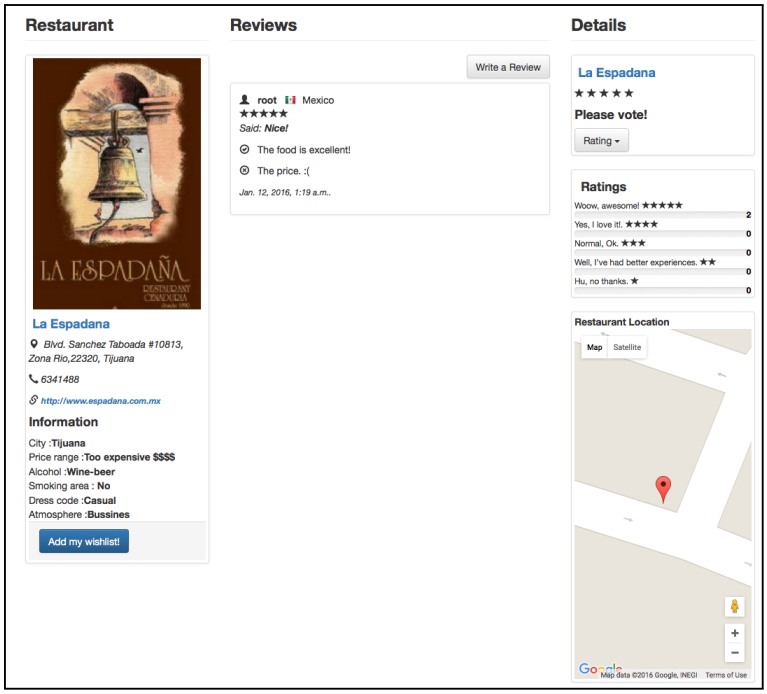



Figure 4.1: User interface of the restaurant model.

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*.

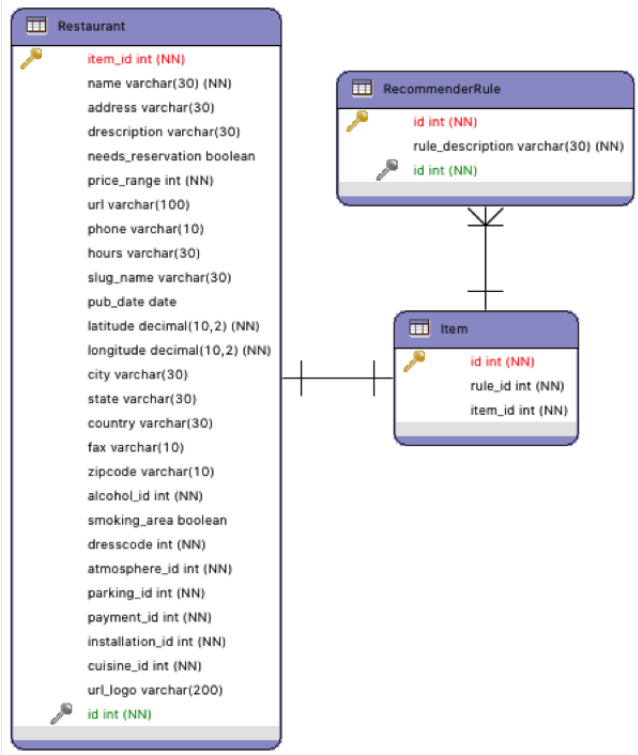



Figure 4.2: The data model of restaurant.

### 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:

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

- **Payment:** *credit/debit card, cash.*
- **Alcohol:** *no alcohol, wine-beer.*
- **Smoking area:** *yes, no.*
- **Dresscode:** *casual, informal, formal.*
- **Installations:** *garden, terrace, indoor, outdoor.*
- **Atmosphere:** *relax, familiar, friends, bussines, romantic.*
- **Parking:** *no parking, free parking, valet parking.*
- **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.*

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

The smoking area is an attribute with boolean value, it defines if a restaurant has a smoking area in its installation.

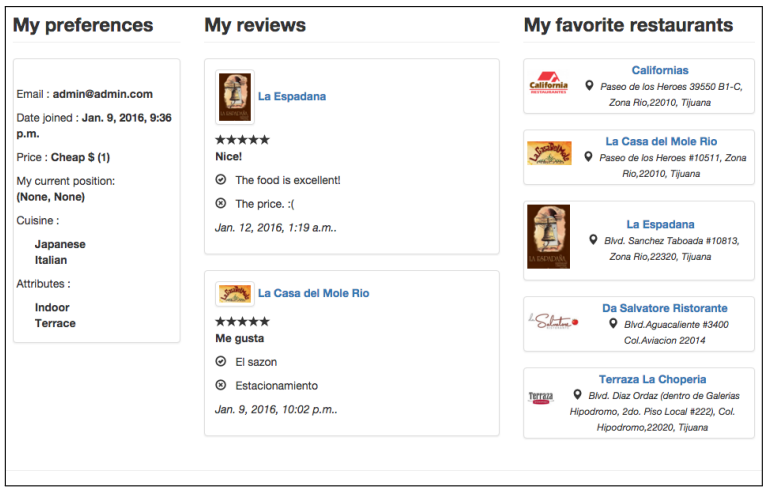



Figure 4.3: Example of user interface for user profile.

### 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: 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 characteristics) could be contained in user profile, this information is used such as contextual information also. The domain values are following:

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

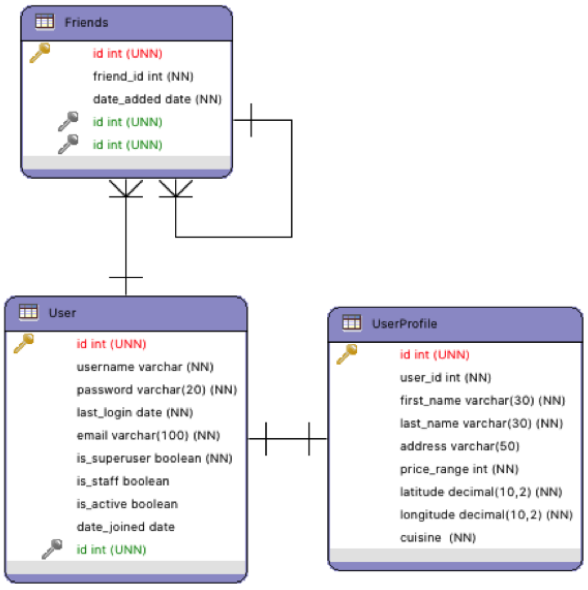



Figure 4.4: The data model of user profile.

- **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.

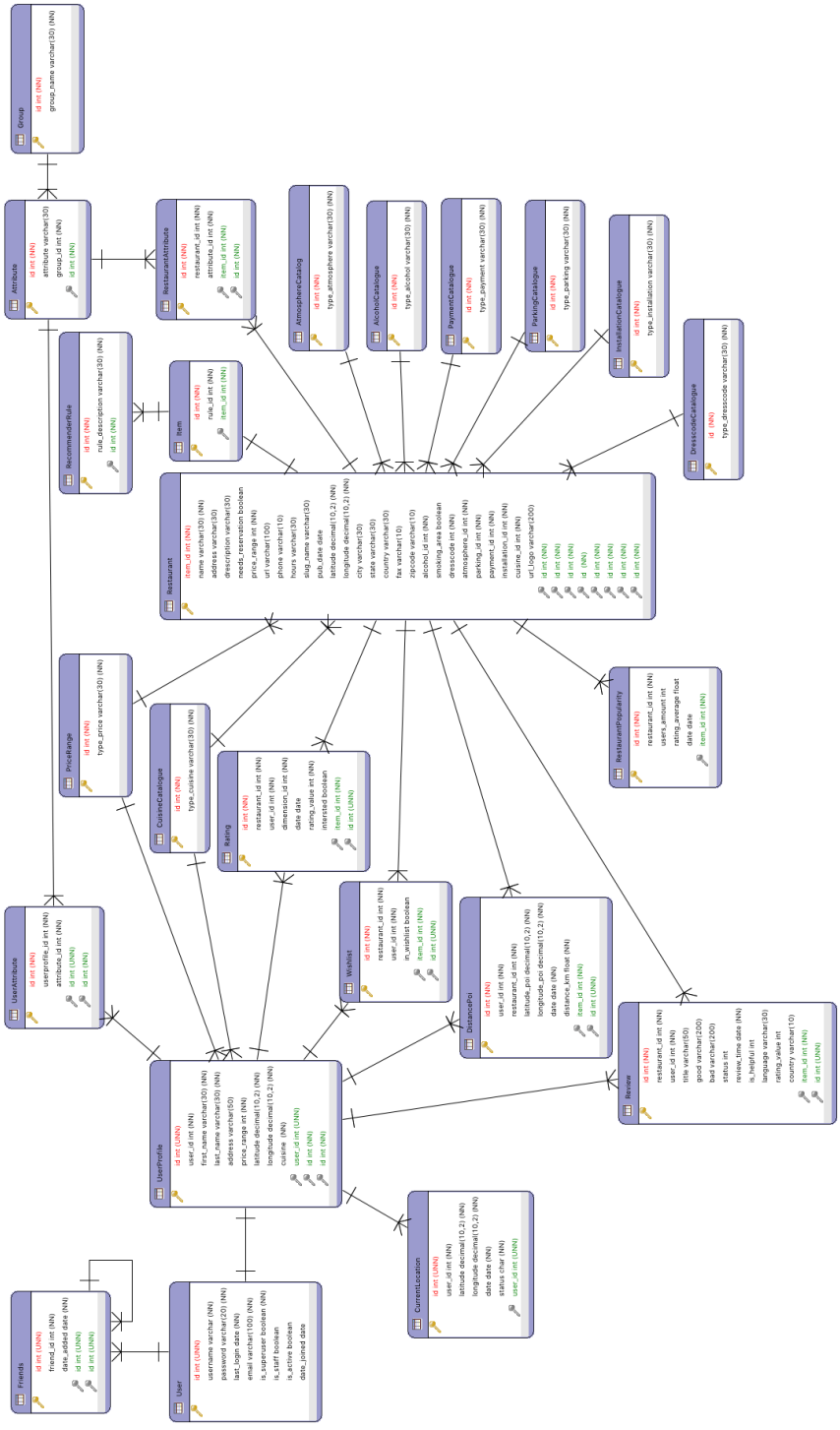



Figure 4.5: The relational database of context-aware recommender system.

## 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 [70].

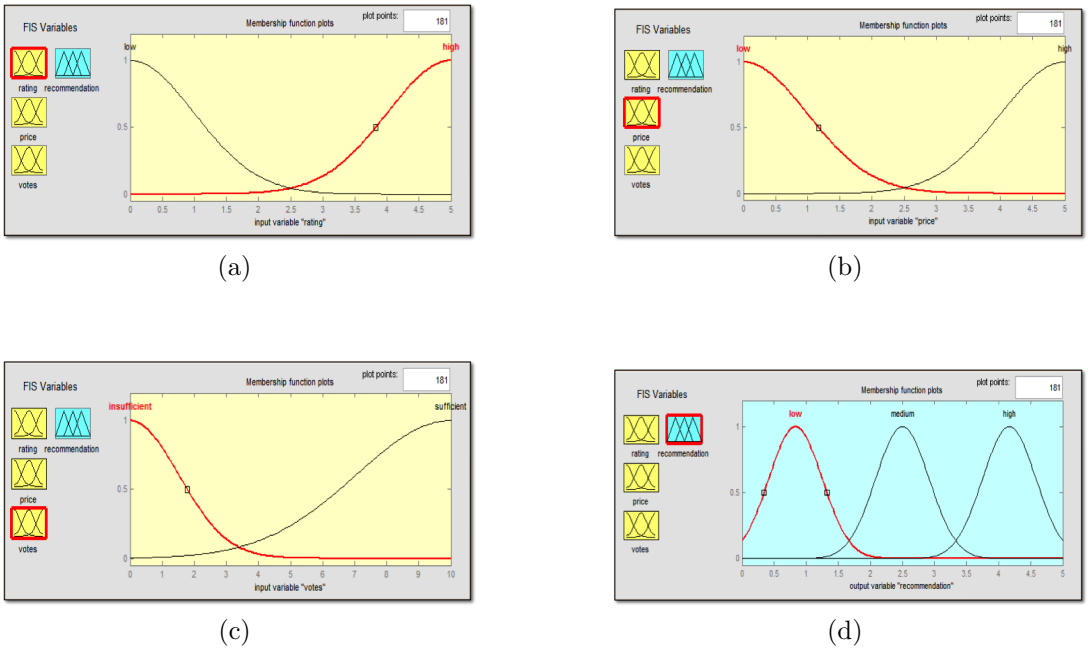
In this work, fuzzy logic is used to model fuzzy variables that highligh in the popularity of a restaurant. The context-aware recommender system has implemented a Fuzzy Inference System that represents the expert recommendation.

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[32]): 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 D.2a), 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 D.2b), 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

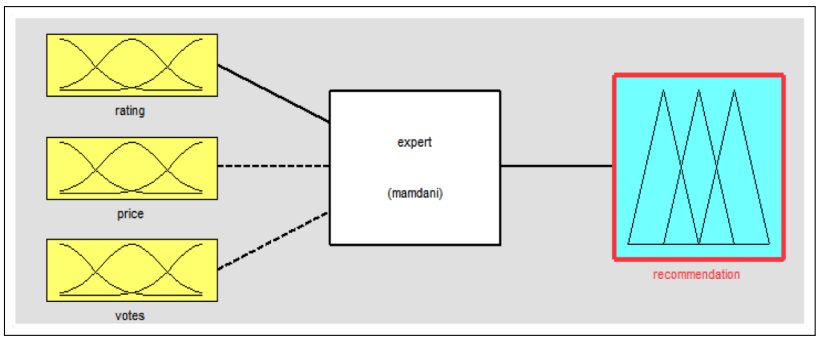



Figure 4.7: Fuzzy Inference System of expert.

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

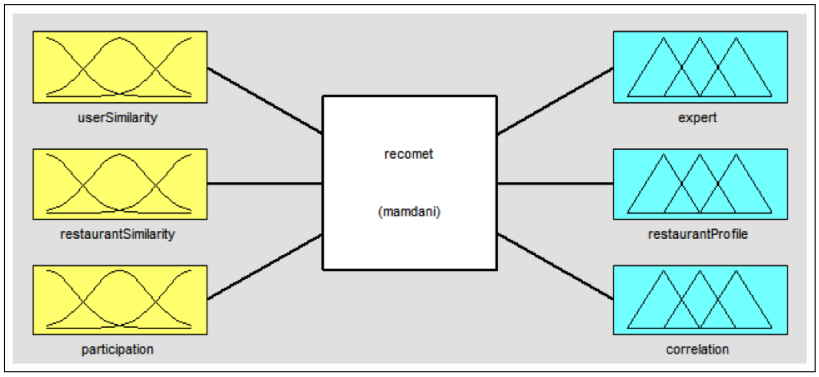



Figure 4.8: Fuzzy Inference System to assign weights.

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

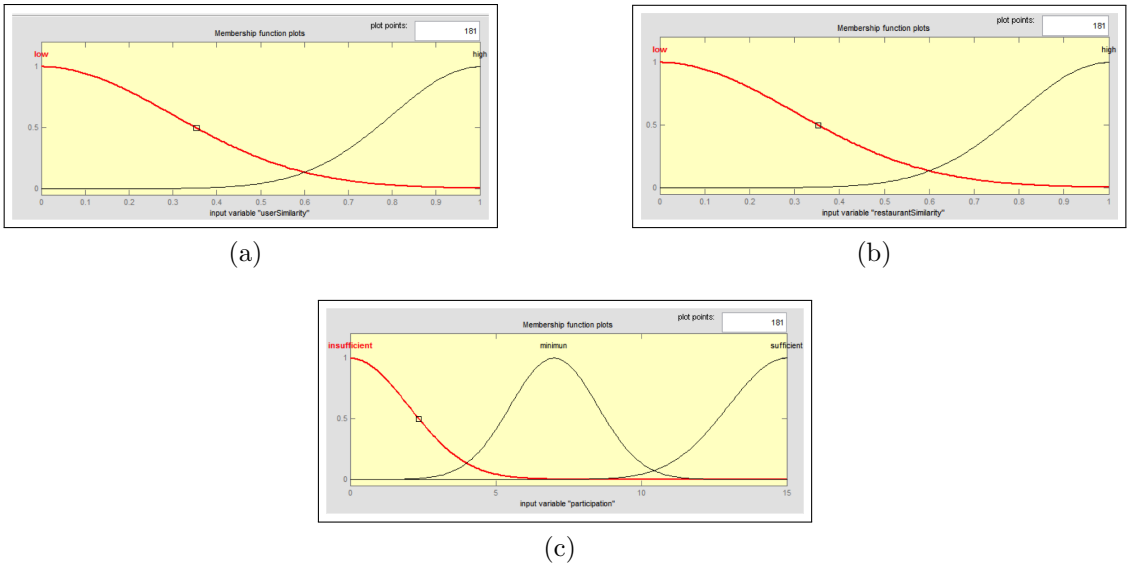



Figure 4.9: The Gaussian membership functions of input variables.

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,* ***correla-***

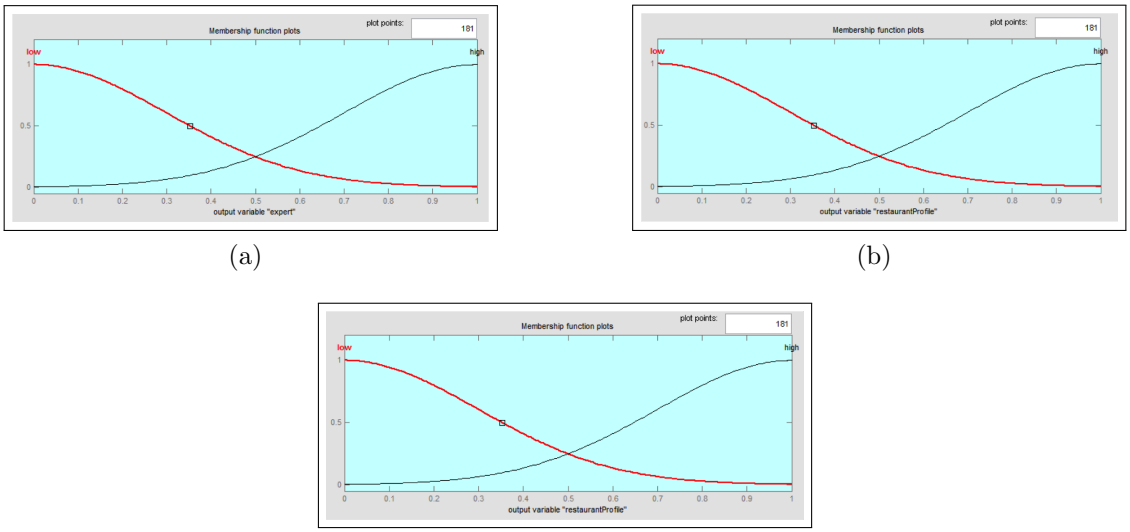



Figure 4.10: The Gaussian membership functions of output variables.

***tion*** *is low.*

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* ***participation*** *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* ***participation*** *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 post-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

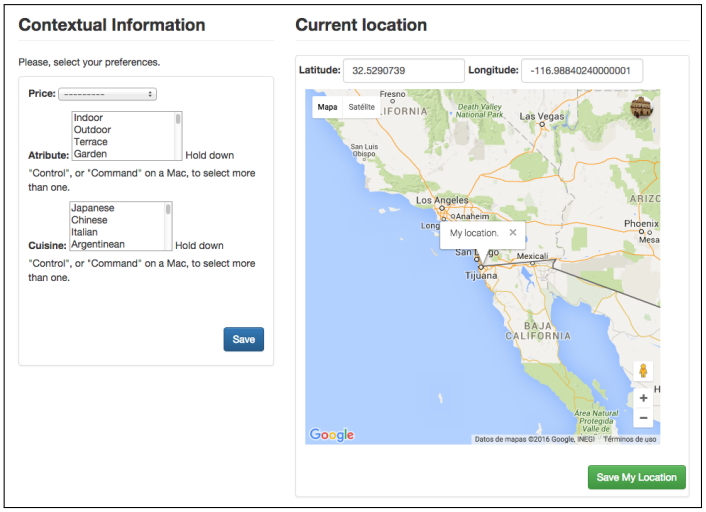



Figure 4.11: System interface to collect contextual information.

favorite cuisine, it can be one or more cuisines such as in attributes also.

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 four 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 services.

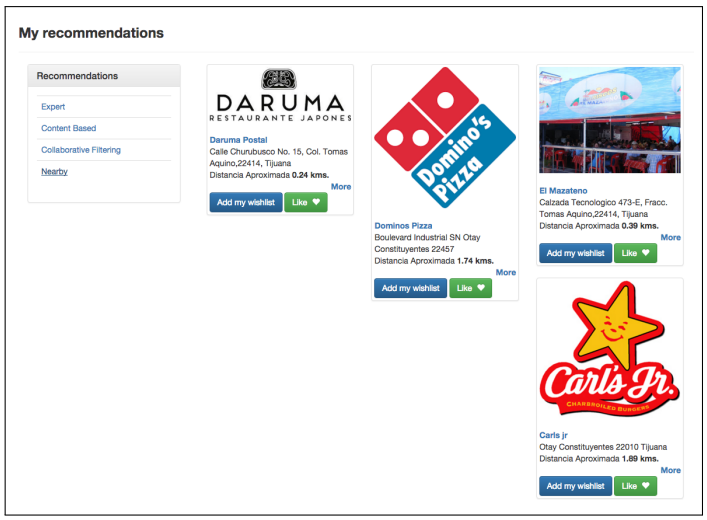



Figure 4.12: System inferface of recommendations for the user.

## 4.5 Methodology

The scheme of poposed method is depicted in the figure 4.13. In the first part, the three techniques of recommendations are suplied by the rating matrix. Ratings matrix makes that **fuzzy inference system** can obtain the inputs values to calculate the output value. **Content-based** utilizes the rating matrix and user profiles to compare the similarity among the restaurants through cosine similarity measure. **Collaborative filtering** is based in user profiles content in ratings matrix, using Pearson correlation calculates the similarity among the users, subsequently, a list of neighbors is obtained to use their preferences to calculate predictions.

The second part shows the recommendation lists for the user. Later, the recommendation lists are reduced when filter of context is applied, i.e., the recommendations

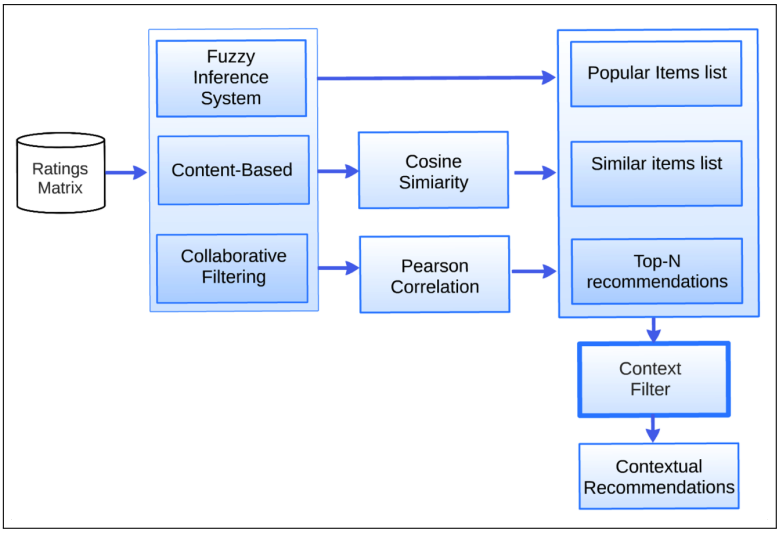



Figure 4.13: Context-aware recommender system methodology.

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 are explained the experiments performed in order to validate the proposed method, each experiment was tested in different context domain.

## 5.1 Tijuana Restaurants dataset

The context-aware recommender system uses collaborative filtering to find restaurants for the user[67]. The ratings of user profiles are used to determine the similarity among users using Pearson correlation.

The similarity between the user and neighbors is used to calculate a weighted average of ratings for each particular item; the top-N ratings are used as a list of recommended restaurants for the active user. The output of the collaborative filter-

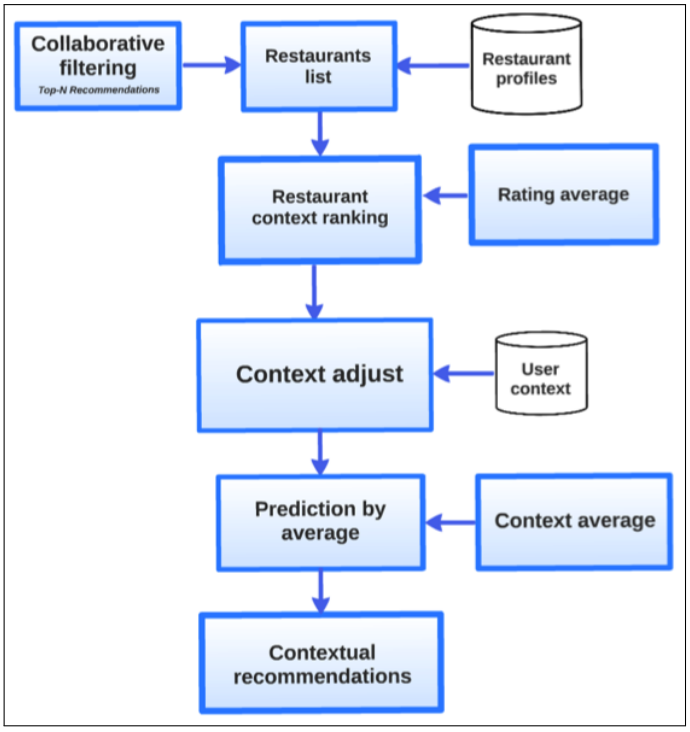



Figure 5.1: The Post-Filtering architecture for Tijuana restaurants.

ing algorithm (top-N list) is supplied to the next step of the post-filtering process. The restaurants are adjusted to the context in the next step in order to make ranking of restaurants in the current context. Post- filtering is based on the average of ratings in a specific context, so prediction is made with: 1) the average post-filtering a restaurant has in the current context (that is the mean of user ratings) and 2) the rating predicted by the collaborative filtering algorithm. The top-N list contains the restaurants with highest predictions, so each restaurant is adjusted for the users context and listed in contextual recommendations; the process is depicted in figure 5.1.

Table 5.1: Questionnaire applied to collect contextual dataset.

| Question | Answers |
|---|---|
| 1.What is your occupation? | 1. Student 2.Employee |
| 2.According your priority, order by importance the features you consider when you choose to visit a restaurant. | 1.Installation/decoration 2.Prices 3.Service 4.Dishes 5.Atmosphere 6.Location |
| 3.What are the devices that you used utilizes? | 1.Smartphone 2.Tablet 3.Laptop 4.PC |
| 4.What Operating System do you used? | 1.Android 2.Windows 3.iOS 4.Symbian 5.Blackberry 6.Other |
| 5.Did you use an application to search restaurants in Tijuana? | 1.Yes 2.No 3.Which one? |
| 6.Would you like to use an application of recommender systems of Tijuana? | 1.Yes 2.No |
| 7.Please, rates your favorites restaurants(without context). | Restaurant list |
| 8.Please, rates your favorites restaurants in contextual situations. | Restaurant list |

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 5.1.

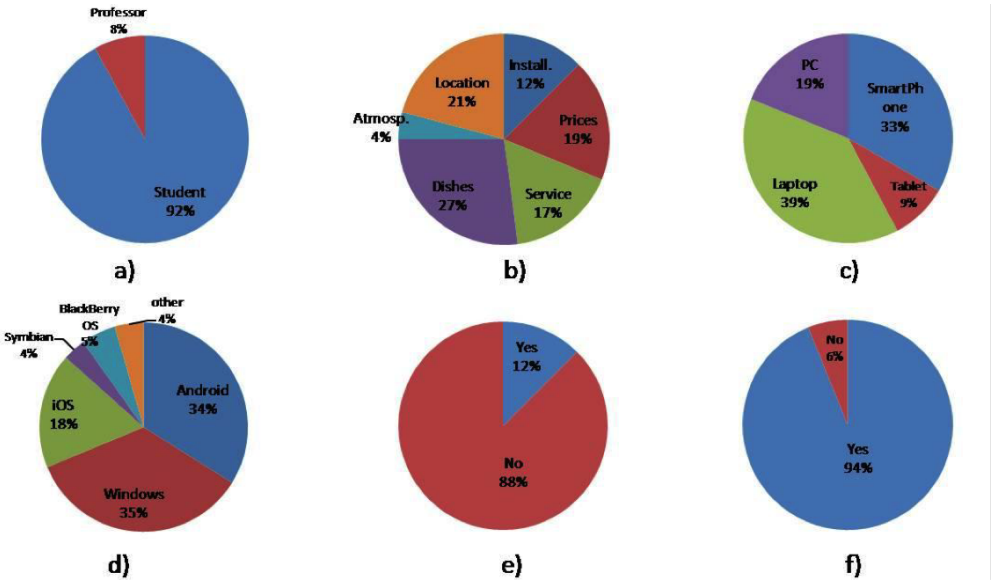



Figure 5.2: The chart shows the users preferences for questions from 1 to 6.

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),

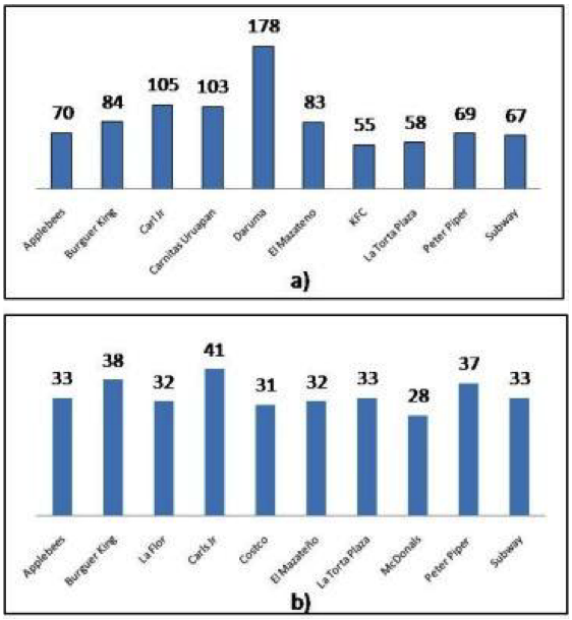



Figure 5.3: The chart shows the users preferences for questions 7 and 8.

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 to combinations of contextual factors shown in table 5.2. The dataset was explicitly collected from **50 users** whom answered questionnaire (see table 5.1). 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

Table 5.2: 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 |

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 [11], [10], [42], and [36], results vary according the techniques that were done.

In [36] 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 [42] 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 [11] 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 [10] the user profile is segmented into micro-profiles corresponding to a particular 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 infor-

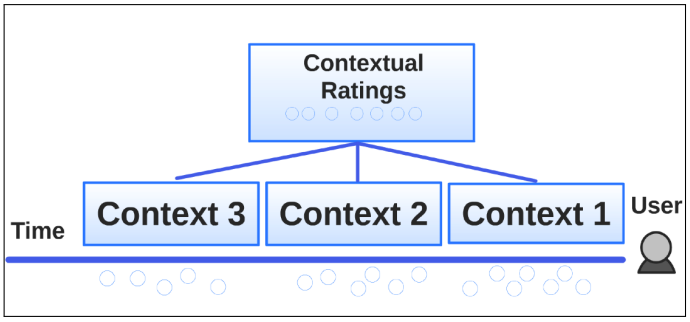



Figure 5.4: Time segmentation of contexts based on current user context.

mation 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 [67]. 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 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

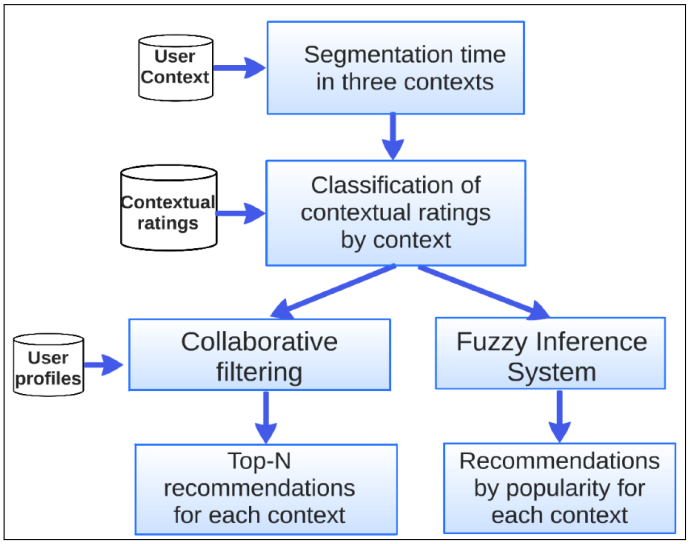



Figure 5.5: Pre-filtering process for context-aware recommender system.

Table 5.3: Results of comparison by contexts in MovieLens dataset.

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

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 5.3 shows the error in three contexts. The error increase in context 3, in this context the ratings matrix is a 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 [71], this datasets was used in [73], [72] to evaluate the performance of context-aware recommender systems.

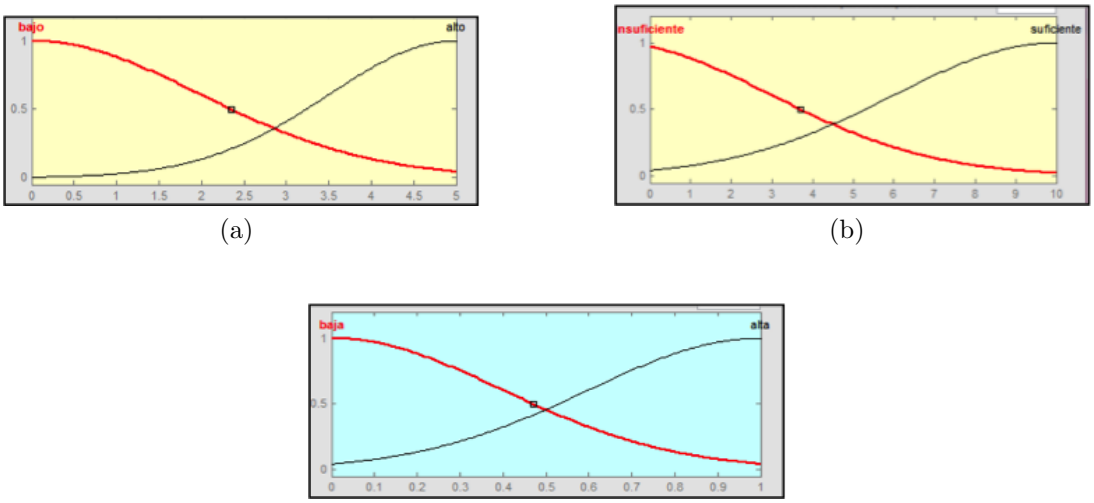
The first dataset contains 4669 contextual ratings, 1202 users and 1890 hotels; the second dataset contains 14175 contextual ratings, 2731 users and 2269 hotels. Data were collected of reviews online in tripadvisor.com. There is only one context: type of trip (family, friends, bussines, romantic and 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[4] 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.*

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.*

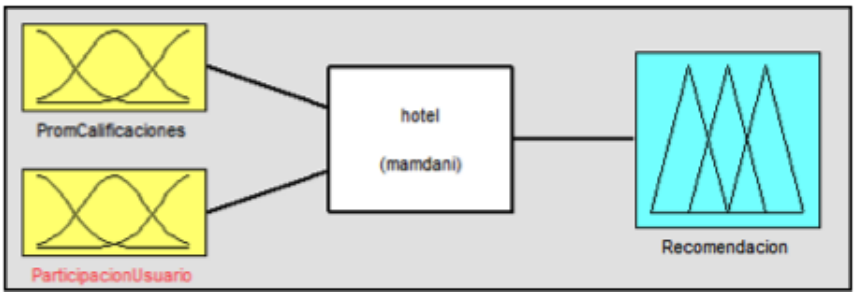



Figure 5.7: Fuzzy Inference System.

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 rec-

Table 5.4: Example of contextual ratings in the user profile.

| User profile | | |
|---|---|---|
| Item1 | Rating1 | Context1 |
| Item2 | Rating2 | Context2 |
| Item3 | Rating3 | Context3 |

ommender algorithm is represented by a list of recommended items. Subsequently applies the context filter and context-aware recommender system gets the final contextual recommendations.

Context-aware recommender system identifies contextual data of the user profile (see table 5.4), and compares recommended items to filter those items that are adjusted

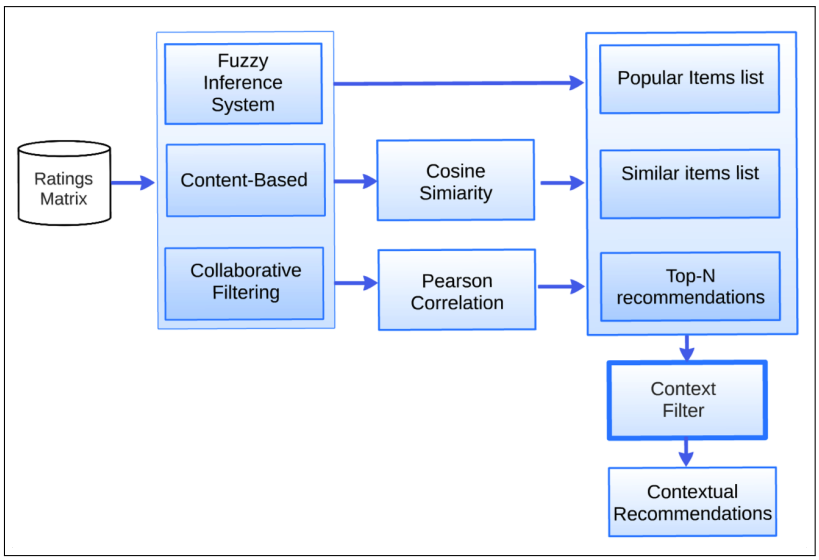



Figure 5.8: Recommender system architecture

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 5.5 describes the data sets and the scarcity percentage of the specified data. Scarcity of 99% mean that there are problems to recommend items because the information is not enought to get good recommendations.

By other side, in table 5.6 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[72]

Table 5.5: Datasets description.

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

Table 5.6: Comparison of RMSE.

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

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 5.7.

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

Table 5.7: Level of similarity among items in datasets.

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

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.

## 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 [35] using the trust level such as context. The web page is *http://www.librec.net/datasets.html*.

Table 5.8: 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. |

**InCarMusic dataset**

InCarMusic dataset[12] has 8 contextual factors and the possible values for contextual conditions are explained in table 5.8. Music tracks were ten different genres. There is not unified music genre taxonomy, for this reason the recommender system uses the genres defined in [66]: classical, country, disco, hip hop, jazz, rock, blues, reggae, pop and metal, 50 music tracks and 42 users in dataset.

### 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

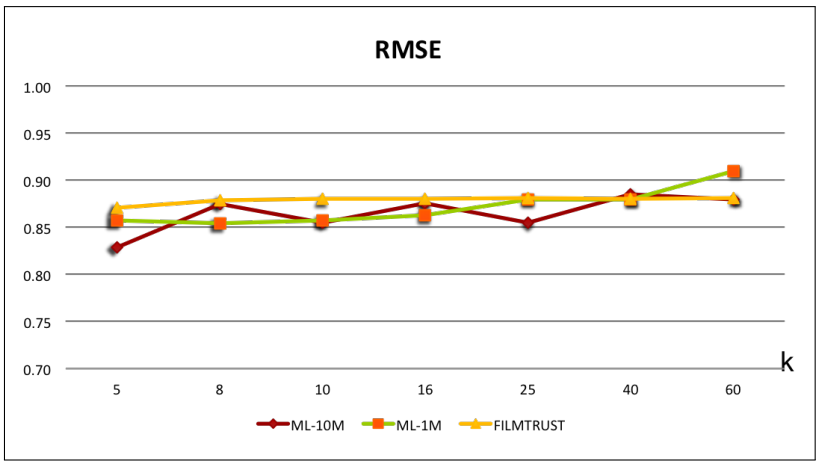



Figure 5.9: RMSE results of matrix factorization test.

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.

By other side, other datasets were used to test matrix factorization under the same parameters to calculate the RMSE for each one. Table 5.9 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 5.9, according the table 5.9 is not possible to assum that matrix factorization has a better

Table 5.9: 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 |

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[6].

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 [6]. 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

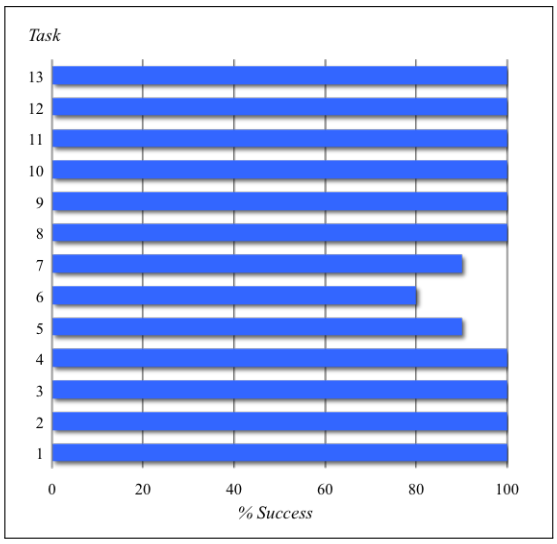



Figure 6.1: Representation of the percent of success for each task.

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 [48] 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 |

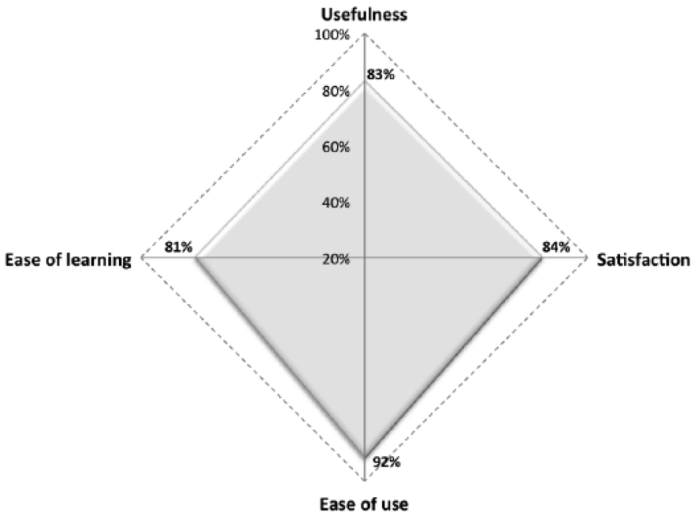



Figure 6.2: The radar chart that depicts the four axis evaluated in the questionnaire.

*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 **??**) 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.
  $ratingMatrix \leftarrow allRatings$
  Call $Recommendations \leftarrow getRecommendations()$ module
  **return** *Recommendations*

**Algorithm 3** Content-Based Algorithm

**Require:** The user id.
**Ensure:** The Top-N list of recommendations.
  $RV \leftarrow$ All items that user rated with 5
  **for** *item* to size of $RV$ **do**
    **if** *item* is not in $RV$ **then**
      $UV \leftarrow itemid$
    **end if**
  **end for**
  $allItems \leftarrow [\,]$
  $getItemsProfilesUser \leftarrow$ Binary vectors of RV
  $allRatings \leftarrow$ Rating matrix
  **for** *item* to size of $allRatings$ **do**
    **if** *itemid* is not in $allItems$ **then**
      $allItems \leftarrow item$
    **end if**
  **end for**
  $getAllItemsProfiles \leftarrow$ Binary vectors of allItems
  $getCosineSim \leftarrow$ getItemsProfilesUser,getAllItemsProfiles
  **for** *item* to size of $highCosineSim$ **do**
    **if** $itemsimilarity \geq 0.8$ **then**
      $highCosineSim \leftarrow item$
    **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 ← UV
    for itemp to size of UV do
        get binary vector of itemp
        itemProfiles ← itemp
    end for
else
    allItemProfiles ← allItems
    for itemp to size of allItems do
        get binary vector of itemp
        itemProfiles ← 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 ← 0
normaItemUser ← 0
normaItemAll ← 0
for position to size of itemProfileUser do
    sumProduct ← sumProduct + (itemProfileUser[position] * itemProfileAll[position])
end for
for item to size of itemProfileUser do
    normaItemUser ← normaItemUser + itemProfileUser[item]^2
end for
for item to size of itemProfileAll do
    normaItemAll ← normaItemAll + itemProfileAll[item]^2
end for
squareRootUser ← squareroot(normaItemUser)
squareRootAll ← squareroot(normaItemAll)
cosineSimilarity ← 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.

# Appendix C

# Technical support of installation

### Dependencies of the application

- Django framework 1.7. Url: *https://www.djangoproject.com/download/*
- Django-registration library. Url: *https://pypi.python.org/pypi/django-registration*
- Django-countries library. Url: *https://pypi.python.org/pypi/django-countries*
- Django-geoposition library. Url: *https://pypi.python.org/pypi/django-geoposition*
- Python-dateutil library. Url: *https://pypi.python.org/pypi/python-dateutil/2.4.1*
- Pyproj library. Url: *https://pypi.python.org/pypi/pyproj?*
- Numpy library. Url: *https://pypi.python.org/pypi/numpy*
- PostgreSQL database. Url: *http://www.postgresql.org/*
- Psycopg2 connection to database. Url: *http://initd.org/psycopg/docs/install.html*

### Tijuana Restaurants dataset

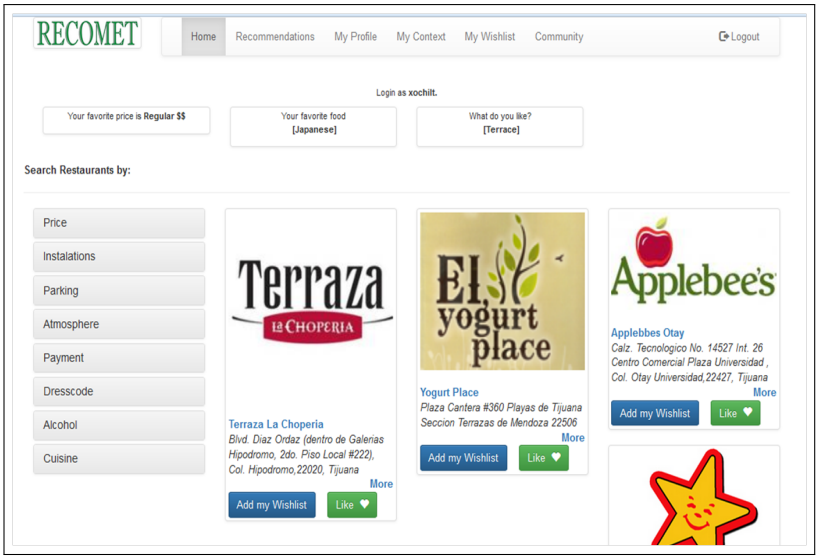
Table C.1: Sample of Tijuana dataset and contextual factors.

| Id | Restaurant | Price | Latitude | Longitude | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Californias | 3 | 32.52837 | -117.02001 | 2 | 0 | 1 | 2 | 2 | 1 | 1 | 20 |
| 2 | Yogurt Place | 2 | 32.53404 | -117.12053 | 1 | 0 | 1 | 2 | 2 | 1 | 1 | 19 |
| 3 | Los Arcos | 1 | 32.51582 | -117.01032 | 2 | 0 | 1 | 2 | 2 | 1 | 1 | 24 |
| 4 | La Casa del Mole | 2 | 32.51995 | -117.01001 | 2 | 0 | 1 | 2 | 2 | 1 | 1 | 20 |
| 5 | Cafe de la Flor | 2 | 32.53198 | -116.94801 | 2 | 0 | 1 | 3 | 2 | 1 | 1 | 16 |
| 6 | Casa Plascencia | 3 | 32.5137 | -117.00712 | 2 | 0 | 1 | 4 | 2 | 1 | 1 | 22 |
| 7 | La Lena | 3 | 32.51238 | -117.00417 | 2 | 0 | 1 | 4 | 2 | 1 | 1 | 17 |
| 8 | La Querencia | 3 | 32.51678 | -117.00961 | 2 | 0 | 1 | 4 | 2 | 1 | 1 | 22 |
| 9 | Big Boy | 2 | 32.51957 | -117.01942 | 1 | 0 | 2 | 2 | 2 | 1 | 1 | 26 |
| 10 | Burger King | 1 | 32.53387 | -116.95086 | 1 | 0 | 2 | 2 | 2 | 1 | 1 | 26 |
| 11 | Cheripan Otay | 4 | 32.51987 | -117.01253 | 2 | 0 | 1 | 2 | 2 | 1 | 1 | 4 |
| 12 | Costco | 1 | 32.50826 | -116.96436 | 1 | 0 | 2 | 2 | 2 | 1 | 1 | 26 |
| 13 | Daruma Postal | 2 | 32.52874 | -116.98579 | 1 | 0 | 2 | 3 | 1 | 1 | 2 | 1 |
| 14 | Dominos Pizza | 1 | 32.53481 | -116.97108 | 1 | 0 | 2 | 2 | 2 | 1 | 1 | 27 |
| 15 | El Mazateno | 1 | 32.52836 | -116.99242 | 2 | 0 | 2 | 3 | 1 | 1 | 2 | 24 |
| 16 | El Porton | 2 | 32.47732 | -117.02924 | 2 | 0 | 1 | 4 | 2 | 1 | 1 | 20 |
| 17 | El Rodeo | 1 | 32.54961 | -116.90429 | 2 | 0 | 1 | 2 | 2 | 1 | 1 | 17 |
| 18 | Giuseppis Rio | 4 | 32.53135 | -116.94833 | 2 | 0 | 1 | 4 | 2 | 1 | 1 | 3 |
| 19 | KFC | 1 | 32.52821 | -117.02397 | 1 | 0 | 2 | 2 | 2 | 1 | 1 | 26 |
| 20 | La Torta Plaza | 1 | 32.53434 | -117.01821 | 1 | 0 | 2 | 3 | 2 | 1 | 1 | 26 |
| 21 | Landini Ristorante | 3 | 32.51483 | -117.01069 | 2 | 0 | 1 | 4 | 3 | 1 | 1 | 3 |
| 22 | Carls jr | 1 | 32.52973 | -116.9682 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 26 |
| 23 | Carnitas Uruapan | 1 | 32.50907 | -116.98759 | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 20 |
| 24 | Fonda Argentina | 3 | 32.51339 | -117.00743 | 2 | 1 | 1 | 3 | 2 | 1 | 1 | 4 |
| 25 | La Espadana | 4 | 32.51786 | -117.00954 | 2 | 1 | 1 | 4 | 3 | 1 | 1 | 20 |

Table C.1 shows a sample of dataset where the domain of contextual factors is depicted as numeric values, from column 5 to 12 are the contextual factors used in the system and each column contains the domain values. The column names are: *payment type*, *alcohol type*, *smoking area*, *atmosphere type*, *dress code*, *installations type*,*parking type*, and *cuisine type*. The complete dataset is available to download in the url: `http://www.xochilt.wix.com/ittresearch`.

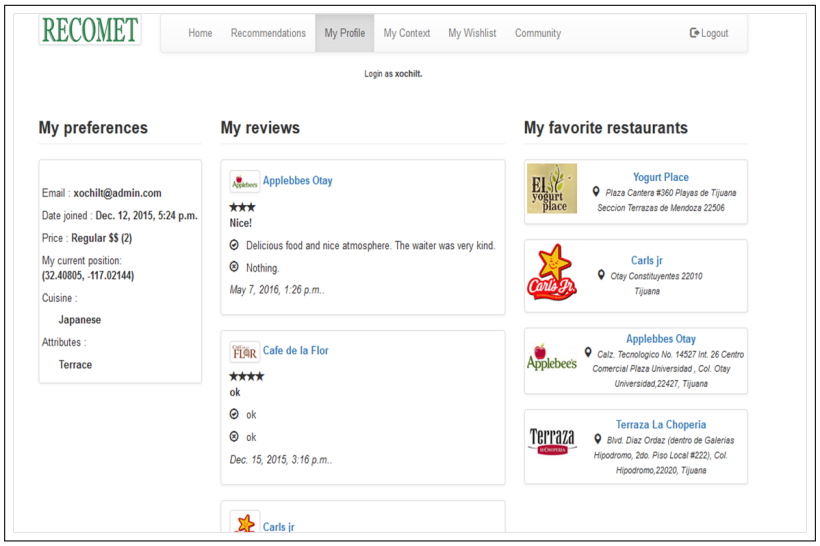
# Appendix D

# System interfaces

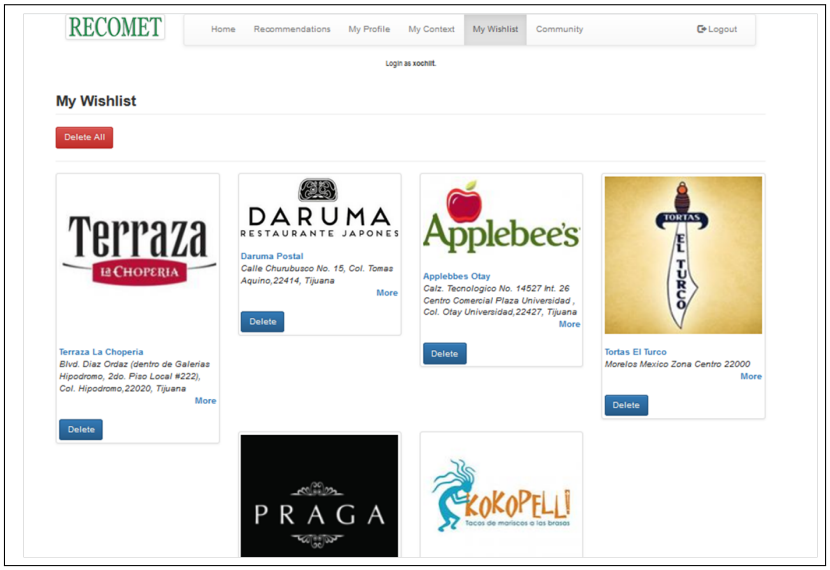



(a)

Figure D.1: Home interface of the system.




(a)




(b)

Figure D.2: a)*My profile* and b)*My wishlist* interfaces of the system.

# 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] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

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

[5] 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.

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

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

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

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

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

[11] 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.

[12] 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.

[13] 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.

[14] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, 16(5):507–526, 2012.

[15] Mary Bazire and Patrick Brézillon. Understanding context before using it. In *Modeling and using context*, pages 29–40. Springer, 2005.

[16] Daniel Billsus and Michael J Pazzani. A personal news agent that talks, learns and explains. In *Proceedings of the third annual conference on Autonomous Agents*, pages 268–275. ACM, 1999.

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

[18] Rebecca Bulander, Michael Decker, B Kolmel, and G Schiefer. Enabling personalized and context sensitive mobile advertising while guaranteeing data protection. *Proceedings of EURO mGOV 2005*, page 445C454, 2005.

[19] 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.

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

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

[22] 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.

[23] Chuang-Kai Chiou, Judy CR Tseng, Gwo-Jen Hwang, and Shelly Heller. An adaptive navigation support system for conducting context-aware ubiquitous learning in museums. *Computers & Education*, 55(2):834–845, 2010.

[24] 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.

[25] 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.

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

[27] Paul Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1):19–30, 2004.

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

[29] Eyrun A Eyjolfsdottir, Gaurangi Tilak, and Nan Li. Moviegen: A movie recommendation system. *UC Santa Barbara: Technical Report*, 2010.

[30] Gerhard Fischer. Context-aware systems-the right information, at the right time, in the right place, in the right way, to the right person. *AVI 12, Capri Island, Italy*, 2012.

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

[32] 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.

[33] 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.

[34] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste:

A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2): 133–151, 2001.

[35] 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.

[36] 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.

[37] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):116–142, 2004.

[38] Arefin Huq, Juan Pablo Bello, and Robert Rowe. Automated music emotion recognition: A systematic evaluation. *Journal of New Music Research*, 39(3): 227–244, 2010.

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

[40] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

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

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

[43] 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.

[44] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. Personalized travel package recommendation. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 407–416. IEEE, 2011.

[45] Qi Liu, Enhong Chen, Hui Xiong, Yong Ge, Zhongmou Li, and Xiang Wu. A cocktail approach for travel package recommendation. *Knowledge and Data Engineering, IEEE Transactions on*, 26(2):278–293, 2014.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] Alvaro Ortigosa, Javier Bravo, Rosa M Carro, and Estefanía Martín. Entornos de aprendizaje móviles adaptativos y evaluación: Comole y geses (adaptive mobile learning environments and evaluation: Comole and geses). *Revista Iberoamericana de Educación a Distancia*, 13(2):167, 2010.

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

[52] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.

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

[54] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[55] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M McNee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134. ACM, 2002.

[56] 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.

[57] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[58] Francesco Ricci. Context-aware music recommender systems: workshop keynote abstract. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 865–866. ACM, 2012.

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

[60] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[61] 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.

[62] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE, 1994.

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

[64] 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.

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

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

[67] 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.

[68] 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.

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

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

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

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

[73] 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.