Acknowledgments

This work is for everybody that trusted my effort and dedication to do everything that I proposed.

I specially want to thank my sisters Rosy, Nely and Mary and of course my parents, as they are the pillars 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, as partners, with me until the end. Im lucky to have their friendship.

I want to acknowledge the effort and patience of my advisor for four years Dr. Mario García Valdéz. I appreciate the knowledge shared and the instructions about conducting the research. Nowadays I have the 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 sensibles al contexto el cual puede ser aplicado en diferentes dominios, para mejorar las recomendaciones utilizando información contextual en un método de recomendación híbrido. El método involucra diferentes técnicas que trabajan simultáneamente para obtener las recomendaciones: filtrado colaborativo y basado en contenido además de un sistema de inferencia para procesar tanto reglas como 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 se destacan las principales contribuciones del método, presentando los resultados de un extenso conjunto de experimentos que validan el método propuesto.

Abstract

A 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 is proposed. This method can be applied in different domains to improve recommendations using contextual information as an hybrid recommender system. The proposed method involves several techniques that work simultaneously: collaborative and content-based filtering, and a Fuzzy inference system to process rules and fuzzy attributes. Subsequently, recommendations are filtered by using contextual factors that represent the current situation of the user. An analysis of the related work is presented, and the main contributions of the method highlighted showing ther esults 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 make a contribution in the application of contextual information in recommender systems by proposing a methodology for the development of context aware recommender systems. The method follows an hybrid approach by integrating several recommendation techniques through a fuzzy inference system. The proposed method was validated by the execution of several experiments, and the implemantion of a case study. The results presented are favorable, indicating the strenghts of the hybrid method.

1.1 Motivation

From it is arising Internet the human life suffer great changes that improves the quality of life style. A lot of apps come to to stablish a different way to realize the daily tasks of people. But Internet is not perfect, with it also comes big amount of problems, since details as a training for persons to use correctly of the network, until the advanced learning to develop sophisticated apps. In a middle level it is the information available to manage systems. From a poin of view of bussines, all bussines of services want to provide information of their services for all users, in the on-line stores is necessary to provide information about items to promote the sales among the customers and new customers, and so, a lot of bussines and users are uploading information with not know that they are contributing to the overload information problem, the reason is that the capability of Internet has not limits. In our daily social interaction we need to take decisions, for instance, to take a bus, to choose a restaurant for dinner, to select a movie in the cinema, etc. This desitions reach us to do anything, despite our unexperience or ignorance respect to discern among the possible alternatives. This lack of experience, time and resources highlight the need of automatic methods to filter relevant information. To obtain recommendations usually the people seek help asking to social friends or persons with a certain level of affinity to any recommendation.

Currently, information filters are an integrated component of the Web, where for instance automatic search engines help users to find or make decisions through personalized recommendations of products and services.

On the other hand, nowadays mobile computing has drasticaly incremented because of the impact of their use, if an application is available in a smartphone it can be used all most anywhere, and thanks to sensors and GPS technology location aware applications are common actually.

Recent technologies also consider information about the user's current situation in mobile applications, as intelligent systems can take advantage of the benefits that this technology provides, to manage the constantly changing context.

The development in mobile and ubiquitous computing [**?**] [**?**], are proposing a wide variety of applications that will benefit from recommendation engines using context to meet their purpose. But still there is a need to model the information available in the system in order to provide context awareness.

Information regarding context can also be described in natural language in a fuzzy manner. For instance when describing the current situation an user could say: *"Is early in the morning and I am currently near my work and I need to find a coffee shop that is not to expensive, is open and not very far"*. This sentence uses many words to describe the situation that are not crisp, but instead, the description uses fuzzy variables to describe the situation.

1.2 Context of use

Context is an important concept in everyday life. People often provide context when writing postcards, referring to the weather or the holiday atmosphere. A knowledge of context can also help to explain why a certain object is produced or designed the way it is. When a product (or system) is developed, it will be used within a particular context. It will be used by a user population with certain characteristics. The user will have certain goals and has the intension to perform certain tasks. The product will also be used within a certain range of technical, physical and social or organizational environments [**?**] that may affect its use.

We can refer to these environments as the *context of use* (see Figure **??**), this concept has been formally defined by ISO standard 9241-11 [**?**] as *"users, tasks and equipment (hardware, software and materials), and the physical and social environments in which a product is used"*.

In daily life we often find products that are difficult to use or understand. This type of difficulties are *usability problems* that arise from diverse issues that have not been addressed in the design of the product's *user-artifact interaction*. The *user-artifact interaction* refers the way that the user interacts with a product and vice versa, this term has been studied in Human Computer Interaction.

As emerging technologies constantly change the way people interact with products

and their physical environment, recent studies have started looking at *human experience* as a source to generate products or systems that *engage* the user.

The *user experience*, *context of use* and *product usability* have been associated in computer sciences field. The *Usability* is became a well-established concept in the IT world to represent the user-friendliness of a system. However, there was a need to establish the concept more clearly and to determine how to measure it. Probably the best known definition of usability is by Nielsen [**?**]: *"usability is about learnability, efficiency, memorability, errors, and satisfaction".* However, the definition of usability from ISO 9241-11 [**?**]: *"the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"*, is becoming the main reference of usability.

Thus, taking in account these definitions we can say that designing for *usability* involves establishing user requirements for a new *system* or *product*, developing design solutions, prototyping the system and the user interface, and testing it with representative users.

In the study of Sato [**?**] that brings *context issues* into design practice, addressed the concept of *context* as a critical component of the design information in order to enhance the *human-centred design practice.* After a literature's revision for definitions of *context* in diverse fields, Sato explains that there are external and internal conditions into the definitions and suggest that it has four characteristics:

1. Aspects of context are based on the nature of actions and conditions.

2. Descriptions depends on the focus of the viewpoints.

3. Contextual changes are triggered from differents elements of the domain.

4. Context evolves over the time, some aspects change fast and others change slow.

From this, Sato defined *context* as a *mental model* or a *pattern of one's memory* triggered by *elements in the situation*, where situation is a collective condition at the scene of interaction composed of relations among *variables of conditions*. Sato employed this concept to describe the *influence of contexts* in *people's interactions* and *system performance* and vice versa.

1.2.1 Logical model of context

The *logical model* explains the relationship between *context*, *contextual information* and *contextual factors*. Figure **??** gives a graphical representation of these relations. The goal is to facilitate the use and implementation of fuzzy context in recommender systems for different domains.

In the Figure **??**, the first box shows different real life situations of the user (contexts), the user can change from situation to situation in a little time or in a long

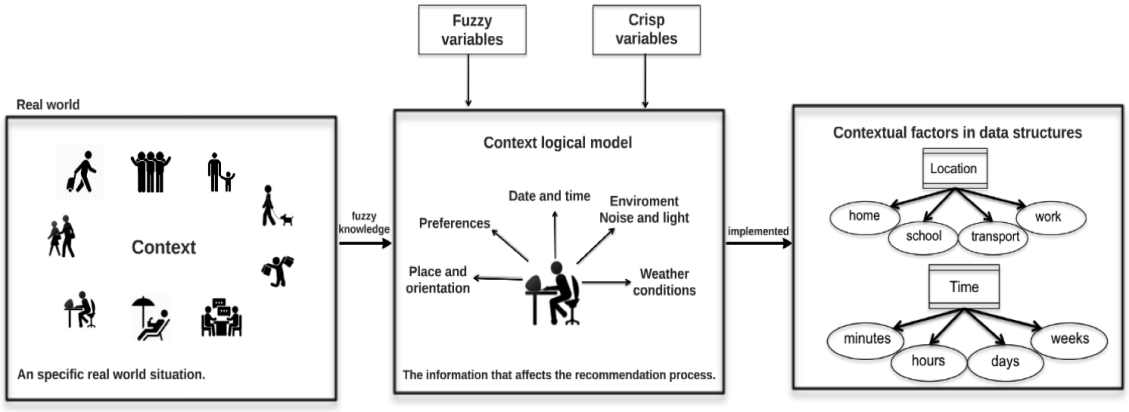


Figure 1.1: Logical data model of context.

time, this situational or contextual information in the real world provides the abstract knowledge that the system will use to determine the *"contextual information"* that will be used. From the real world, specific data will be extracted by different means as sensors, user interaction or even manual input. In this work we call this data *"contextual factors"*(for instance, place and orientation, preferences, date and time, etc.) and represent the information that affects the recommendation process. The information could be represented as fuzzy variables or crisp variables depending of its domain. Later, contextual factors are implemented as data structures that define the domain for each one.

1.3 Context-awareness

Traditional recommender systems provide 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 on a single type of item [**?**]. Some improvement efforts for recommender systems, have been mainly focused on the *integration of context* in the recommendation process. The idea behind *context-aware computing* is to provide information or services for the user based in the user's situation [**?**]. 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 that is not easy to define, as it is related with several disciplines that propose different definitions. For example, the authors Bazire et.al. [**?**] compare the notion of context in different fields and conclude that is complicated to make a unifying definition of context because of the nature of the concept in many disciplines. In computer science Fischer [**?**] 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"*. Fischer 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.

Probably the definition most used in the field of recommender systems to define *context* is proposed by Dey [**?**]: *"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 [**?**] [**?**], music [**?**] [**?**] [**?**], places of interest [**?**], movies [**?**], vacation packages [**?**] [**?**], travel guides [**?**], e-learning [**?**] and restaurants [**?**].

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

The method proposed in this thesis uses three recommendations techniques, applied as a case study in a restarant domian:

1. *Expert's Fuzzy Inference System*, this is a rule based recommender defined by an expert in the domain, in the case of a restaurant recommender, 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's 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 for the current context. This is the last step and is represented as a *context filter* in the method, as a result a list of contextualized recommendations is obtained.

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 section **??** and **??**, respectively.

To evaluate the performance of the proposed recommender several experiments were maded, the algorithms were tested using contextual datasets and the number of contextual factors used varied according to the information provided by the dataset. The goal of the experiments were to observe the role of contextual data in the performance of the algorithms and to have a better understanding of what contextual factors are more important for users in a specific situation, how recommendations

are improved using context and, the accuracy in recommendations. Chapter 5 shows the results while discussion about results are explained in each section.

As a user-cenetered system another important aspect to consider and measure is the *user satisfaction*, for this two metrics were used: *task-success* and *time-on-task*. These usability metrics allow designers to measure the user experience.

Usability metrics can help reveal patterns that are hard or even impossible to see. Evaluating software applications with a small sample size (between five and eight tests) usually reveals the most obvious usability problems [**?**].

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 [**?**].

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.

1.4 Aims

Recommender systems has been used to obtain relevant items for current user through simple models (user-item), the technique is limited because of ignore in-

formation about behaviour of users and others factors that are involved in the user environment. To implement new tecnologies to adapt the recommender algorithms for the user needs is the novel approach in recommender systems. The purpose of them is to improve the suggestions in order to increase the user satisfaction as well as to facilitate the interaction user-system. The improves proposed by researchers in this field involve the context implementation, through contextual factors defined according the domain proposed, and considering experiences and recommendations of other authors. The goal of this thesis highliths the implementation of context in a method that includes fuzzy logic techniques and pre-filtering paradigm that uses traditional recommender techniques improved to make recommendations. To achieve this, uses of fuzzy rules to treat linguistic terms that include fuzzy variables, this allows the use of aproximated information to the real context of the user, and it allows an improved performace of the recommender system because analyzes the user preferences and obtain recommendations based in contextual factors. For instance, the system provides a list of range of prices, this allows the users to select a specific range of price to get recommendations adjusted for the tag selected, this tag represents a common word of their native language.

The contribution of this thesis is the novel method that includes a hybrid recommender system (content-based and collaborative filtering techniques). It provides a *useful knowledge* to utilize in the hybrid recommender system, provides techniques

to maximize the use of context by fuzzy variables that represent a part of context. The context complementation is represented by external factors such as web services. Plus, the method has features that allow to adapt it to be implemented in different domains such as e-learning, movies, music, tourism, etc. For a case of study, the restaurants domain was used to test the method.

Several particular aims were done in order to support the achievement of contributions:

- Elaborate an analysis about the state of the art in the field of context-aware recommender systems through the revision of the relevant literature. State of the art helps to define the tendency of currents recommender systems, the recent improves and helps to understand what is the relevance of the proposed method as well as the factibility and how we can make an important contribution to the recommender systems field.

- Selection of algorithms that are representative of the alternatives for this problem domain in order to compare their perfomance. An analisys of the features and performs of the algorithms was done, it was extracted of papers, books, and toolbox results to choose the suitable algorithm for the method. There are enough papers to make a consense about what is that we want from each one.

- Design and conduct several experiments with the proposed algorithms. The experiments were based in previous results of similar methods, the algoritms were adjusted to improve their performance and testing with datasets for several applications, such as travels, hotels, restaurants and movies. The metric implemented to measure the experiments results was root mean square error beause of is the most recommended by literature.

- Propose a hybrid method and apply it in a case of study. Subsequently the results obtained, the proper algorithms and paradigm was included in the proposed method. As well as the fuzzy inference system and the fuzzy variables to represent the context used by the recommender system.

- Develop a prototype of a context-aware recommender system using the proposed method. Using recent technologies to develop a proper interface for restaurant recommender system as case of study, the development was done to apply the proposed method. The context is represented by fuzzy variables and database models related the user model and characteristics that describe the restaurant model.

- To evaluate the proposed method the usability metrics were proposed. We consider the nature of the recommender system and the results that we can get of usability test. The test was prepared to be realized by real users that

never have been interacting with recommender systems. Results are explained in a posterior chapter.

1.5 Outline

The rest of this thesis is organized as follows:

- **Chapter 2** describes an in-depth study and background of current and related work, presenting a general overview of recommender systems and their evolution in recent years. This study includes traditional recommender systems, their methods and techniques to improve recommendations, aswall as the challenges of these systems. Subsequently, hybrid methods used in different applications, their limitations and advantages for each hibridation and the domains of application. Finally, context-aware recommender systems are disscussed, in the same way, with an analysis of the advantages and disadvantages of the use of context in recommender systems.

- **Chapter 3** describes the fundamental concepts that form the basis of 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 includes the overall explanation of data models and the 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, the impact of context in recommendation process was evaluated. 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 limitations. Final conclusions are drawn and also proposals for future work are presented.

At the end, this thesis includes appendices that describe detailed technical aspects about installation and libraries of python that uses the context-aware recommender system *(appendix* **??***)*, the pseudocode of algorithms proposed in the method are available in *(appendix* **??***)*, and experiment study materials used to obtain the test

results are in *(appendix* **??***)*.

Chapter 2

State of the art

In this section the state of the art in conventional and context-aware recommender systems is presented. As a technology recommender systems have been applied in many domains, and sometimes they represent the key technology for the success of web and mobile applications.

2.1 Traditional recommender systems

Some works utilize social information to recommend such as Manca et.al. [**?**] where the Friend recommender system is applied for social bookmarking, its goal was to infer the interest of users from content, selecting the information available of the user behavior, and analyzing the resources and the tags bookmarked for each user.

Recommendations are generated through the mining of user behavior in a tagging system, analyzing the bookmarks tagged by each user, and the frecuency for each tag used. J.Yao et.al. [**?**] 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. [**?**] presents FilmTrust, a website that integrates semantic web-based social networks, augmented with trust, to create prediction of movie recommendations. FilmTrust takes on the role of a recommender system, forming the core of an algorithm to predict a rating for movie recommendations. 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 have benefits and problems, 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 of each algorithm. A popular example of an hybrid system is called Fab, which was developed by Balabanovic et.al. [**?**]. Fab is a system for the 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. [**?**] presents the 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. [**?**] 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. L. Martinez et al. [**?**] presents REJA, a hybrid recommender system that involves collaborative filtering and a knowledge-based model, that is able to provide recommendations in some situation for a user; besides, it provides georeferenced information about the recommended restaurants. L.Castro et.al. [**?**] proposes 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 a recommender system utilizes three techniques to recommend: demographic, collaborative and content-based. The goal is to recommend tourist packages that matches

the user's profile. Others works with hybrid recommender systems are ProfBuilder [**?**], PickAFlick [**?**] and [**?**], where multiple recommendation techniques are presented. Usually, a recommendation requires the ranking of items or the selection of a single best, at this point some technique must be employed to achieve this.

Traditional recommender systems (single or hybrid) 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 users' preferences are 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.

2.2 Context-aware recommender systems

Overall, Context factors are able to make a recommender system adaptive to the ever 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. Some examples of this types of system are explained next.

Abowd et. al. [**?**] proposed the Cyberguide project, which encompass several tour

guide prototypes for different handheld platforms. Cyberguide provided tour guide services to mobile users, leveraging in the recommendation process both *contextual knowledge* about current users and their *locations* they have benn in the past. Cena et al. [**?**] presents a tourist guide for particular contexts in *intelligent content adaptation*. The UbiquiTO system is a tourist guide that integrates different forms of context-related adaptation for: *media device type*, *user characteristics and preferences*, and 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 [**?**] 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 both the appropriate *physical context* and *profile parameters*, for example, for a certain *location* and *weather*, a facility should not be selected if is too far away from the user's current location and a cold beer should not be recommended if it is raining. Another feature is related with advertisement and offers from suppliers presented in 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. Schifanella

et al. [**?**] developed 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*. Mob-Hinter 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. Recommendations are computed using the *available ratings* of the user neighbors. The PECITAS system [**?**], 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 present LARS [**?**] 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 the "I'm feeling LoCo" system [**?**] that proposes a *ubiquitous location* based recommender algorithm that focuses in the *user experience* considering *user preferences*, *time*, *location* and *similarity measures* automatically, using the Foursquare dataset. The information of the *user's social network*, *transportation* and *phone's sensors* is inferred to provide a recommendation of places. Reddy et.al [**?**] presents the LifeTrack system which incorporates sensor information for selecting tracks in a music recomender system. The songs are represented by tags of terms, that the user assigns in order to link songs with the *appropriate contexts* in which they should be played. *User feedback* is incorporated to make a song more or less likely to be played in a given context. Contextual data considered as relevant to song selection includes *location*, *time of operation*, *velocity of the user*, *weather*, *traffic* and *sound*. *User locations* and *velocity* are determined through GPS. Location information includes tags as *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). Ricci et.al. [**?**] uses context in a music domain, using a model-based paradigm, in this context-aware recommender system 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 factors, users were requested to evaluate music tracks in different contextual situations for each genre. The prediction takes in to account this *relevance* to recommend music tracks prefered by the user according the *genre* and contexts mentioned. In the domain of restaurants Chung-Hua et al. [**?**] proposed a context-aware recommender system for mobiles using a post-filtering paradigm, the architecture involves a client-server model that works with a request of data from the client side to the server. Subsequently, taking in account the contextual factors to filter the appropiate restaurants to recommend. That context-aware recommender system uses contextual factors such as *location and season*, and also uses *user preferences* to personalize the recommendations to a particular context. Baltrunas et.al. [**?**] presents ReRex for tourism, a context-aware

recommender system, based on a *model-based paradigm*, the system recommends and provides explanations about why certain Places of Interest(PoI) were recommended. The proposed model computes a personalized context-dependent rating estimation. Subsequently, in order to generate the explanation, the system uses the factor that in the predictive model has the largest positive effect on the rating prediction. 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 the explanations are diffucult to understand, however, users recognize that the explanation is a very important component, that influences the system's acceptance. Finally, Noguera et. al. [**?**] 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 mobile networks. It is also in charge of tracking the *user's location* and *speed* based on GPS technology. The system utilizes *pre-filtering* and *post-filtering* paradigms. Pre-filtering is used to reduce the number of items considered for the recommendation according to the user's 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

considering the experience of other diners concerning the recommended restaurant. To sum up the contextual recommender systems mentioned, Table **??** describes the main contextual factors, the domain of application, the method used for each application, and type 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

In this chapter, the fundamental concepts related to this work are presented, the formal definitions referring to the fuzzy logic, fuzzy systems and linguistic terms are explained, the concept of context and its classification, and contextual factors, as well as the concepts of recommender system techniques and their main problems are explained.

3.1 Production systems and fuzzy models

A central aspect of the proposed method is the use of both fuzzy logic and fuzzy inference systems, in this section formal definitions of these models of knowledge representation are presented.

3.1.1 Traditional Production Systems

Production Systems represent knowledge in the form of rules, which specify actions that will be executed when certain conditions are met. In these systems experts in a 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 of mainly these three components [**?**] [**?**]:

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 [**?**]:

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 which case the work is done from the conclusion to the facts, to chain-backward, goals in the WM are matched 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 [**?**]. In the next section

a review of the extension of production systems with fuzzy logic is presented.

3.1.2 Fuzzy Production Rules

Fuzzy production rules use fuzzy logic sets to characterize the variables and terms used in propositions. 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) [**?**]. Depending on the form of the consequent, two main types of fuzzy production systems are distinguished [**?**]:

- **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 [**?**] 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 FISs are used mainly in control systems, and are basically composed of five modules [**?**]:

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 [**?**]:

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

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

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

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

Three main inference systems can be described:

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

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

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

3.2 Context

People transmit ideas to each in a complex way. This is due to many factors such 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 people talk,

they 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 poor mechanisms for providing inputs. 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 a human-computer interaction enabling the development of more useful computational services.

In order to use context effectively, we must define what context is and how it can be used. An understanding of *how context can be used* will help application designers to determine what context-aware behaviours to use in applications [**?**].

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 [**?**] refer to context as *location*, *identities of nearby people and objects*, and *changes to those objects*.

This type of definitions that define context by example are difficult to apply when developers try to determine whether a type of information not listed in the definition is part of the context or not, as it is not clear how it can be used by the definition. Schilit et al. [**?**] affirms that the most 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 general, context is all about the whole situation relevant to an application and its set of users. It is complicated to enumerate which aspects of all situations are important, as this will change from situation to situation. For this reason and for the purpose of this thesis, the definition of context proposed by Dey [**?**] has been adopted (see section **??**).

However, another important aspect is to stablish a meaningful classification that covers the characteristics that describe the contextual factors.

Dourish [**?**] 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 activities.

Context should include information to allow systems to use contextual information about users and their situation, enabling the system to provide users 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 of a situation is involved in the *environment* of the people, 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 change over time, in 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 it anymore. As for the representation of the *"change of time"*, a data model of *time* should be specified in a way that the system *interprets* time as 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 [**?**] proposes two important aspects that highligh when different kinds of context are defined: *what a recommender system may know about these contextual factors*

and, *how contextual factors change over time.*

A recommender system can have different types of knowledge about the contextual factors, which may include the exact list of all the relevant factors, their structure, and their values. 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 the 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 that only the *Time*, *PurchasingPurpose*, and *ShoppingCompanion* factors matter in this application. Further more, the recommender system may know the structure of all 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 been made*, *with whom*, and *for whom*.

- **Partially observable**: Only some of the information about the contextual factors described above, is explicitly known. For example, the recommender system may know all the contextual factors, such as Time, PurchasingPurpose,

and ShoppingCompanion, but not their structure. Note that there can possibly be different levels of *"partial observability"*.

- **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 remain the same (stable) over time. For example, when recommending the purchase of a certain product, such as a shirt, we can include the contextual factors of Time, PurchasingPurpose and ShoppingCompanion without change 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.

On the other hand, Fling [**?**] 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 [**?**]: *"a system is context-aware if it uses context to provide relevant information and/or services for the user, where relevancy depends on the user's task."*

This definition is closer to the reality about behaviour of *context-aware recommender system* when incorporates contextual information. Based in this definition, 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 for indoor activities. Therefore, it is not context. The presence of other people, however, can be used to characterize the users situation. If any 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

Recommender systems are software tools and techniques providing suggestions for items to be of use to a user [**?**]. The suggestions relate to various decision-making processes, such as *what items to buy*, *what music to listen to*, or *what online news to read*. *"Item"* is the general term used to denote what the system recommends to users. A recommender system normally focuses on a specific type of item (for instance CDs or news) and accordingly its design, its graphical user interface, and the core recommendation technique used to generate the recommendations are all customized to provide useful and effective suggestions for that specific type of item. Recommender systems are primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a Web site, for example, may offer [**?**]. A case in point is a book recommender system that assists users to select a book to read. In the popular Web site, Amazon.com, the site employs a recommender system to personalize the online store for each customer [**?**].

In their simplest form, personalized recommendations are offered as ranked lists of items. In performing this ranking, recommender systems try to predict what the most suitable products or services are, based on the users preferences and constraints. In order to complete such a computational task, recommender systems

collect from users their preferences, which are either explicitly expressed (for instance as ratings for products) or are inferred by interpreting user actions [**?**].

3.3.1 Collaborative Filtering

The idea behind collaborative recommendation approaches is to exploit information about past behavior or opinions of an exisiting user community for predicting which items certain user of the system will most probably like or be interested in [**?**]. Recommender systems are useful in several types of applications, however, their biggest impact has been mainly in ecommerce web sites in order to personalize the information for a particular user as the system can help to promote several items of his or her interest, thus increasing the sales of the on-line store. In traditional implementations a collaborative filtering algorithm takes as input a given *user-item* sparse matrix of ratings to generate a prediction for each user-item 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. Different 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 an approach that only uses the rating matrix

to obtain recommendations. The neighborhood selection consists in taking the k nearest (similar) neighbors into account usind the k threshold to define the size of the neighborhood. A small neigborhood can not make accurate predictions, and on the other hand if the neighborhood is too large the information about certain nighbours could not be significant.

To obtain the similarity value between a user and his neighbors, the Pearson correlation 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 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 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 as the *User-based* recommendation, the difference is that this approach tries to find similar items instead of similar

users to make a prediction using again only the rating matrix as input. Then, in an *item-based* recommendation is to compute predictions using the similarity between items and not the similarity between users. To find similar items a cosine similarity measure is often used, this metric measures the similarity between two *n-dimensional* vectors based on the cosine of 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 by Equation **??**:

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

Where the dot symbol $(\cdot)$ 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 is first processed offline, as described for *item-based* filtering or also using some dimensionality reduction technique. At run time, only the learned model is required to make predictions. Although a *memory-based approach* is theoretically more precise because the full data is available for generating recommendations, such systems face problems of scalability when for instance a database 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 information available 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 [**?**], however, are no longer *purely* collaborative, and also bring new questions on how to acquire the additional information and how to combine the different classifiers. 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 sparsity [**?**]. 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 [**?**].

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 [**?**]. A similar strategy of asking the user for a gauge set of ratings is used for instance in the Eigentaste algorithm presented in [**?**].

3.3.2 Content-based algorithm

In a content-based recommendation, the task consists of determining those items that best match the active users preferences. 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 some-

thing are not always related to certain product characteristics and may be based on a subjective impression of the items exterior design [**?**].

Content representation

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

Vector space model

Content-based systems have historically been developed to filter and recommend text-based items such as e-mail messages or news. The standard approach in content-based recommendation is, therefore, not to maintain a list of *meta-data 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 [**?**].

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 [**?**] 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 is to avoid the *overspecialization*, therefore increasing the serendipity of the recommendation lists that now includes unexpected items in which the user might be interested, because sometimes 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.

In all described filtering techniques, recommendation accuracy improves along with the number of ratings; significant performance increases for the learning algorithms were reported in [**?**] when the number of ratings was between twenty and fifty. However, in many domains, users might not be willing (or is not feasible) to rate that 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 a free-text input.

3.3.3 Hybrid recommender systems

Each recommender system technique has its advantages and disadvantages, 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 are able to fully exploit all of these.

Consequently, building hybrid systems that combine the strengths of different algorithms and models to overcome some of the afore mentioned shortcomings and problems has become the target of recent research.

Hybrid recommender systems are technical approaches that combine several algorithms or recommendation components [**?**].

3.3.4 Context-aware recommender systems

Traditionally, the recommendation problem has been viewed as a prediction problem in which, given a user profile and a target item, the recommender system's task is to predict that user's rating or that item, reflecting the degree of user's preference for that item [**?**]. Specifically, a recommender system tries to estimate a rating function: $R : Users * Items \rightarrow Ratings$, 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*: $R : Users * Items * Contexts \rightarrow Ratings$, where *Contexts* represent 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 [**?**].

A multidimensional model as those found in data warehousing systems [**?**] is used to depict the context dimensions, in Figure **??** the time dimension belongs to the

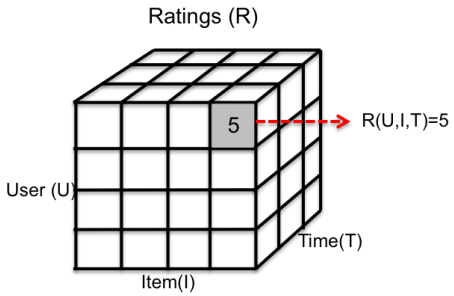


Figure 3.1: Multidimensional model of context.

set of contextual factors and, is described as a *set of attributes*, for instance it may consist of attributes such as *morning*, *evening*, *nigth*, etc., as it was mentioned in section **??**.

3.3.5 Paradigms for using of contextual information

When recommender system uses contextual information, it starts with the data having the form *Users * Items * Contexts * Ratings*, where *Contexts* represent an additional contextual dimension and end up with a list of contextual recommendations: $item_1$,$item_2$,$item_3$...$item_n$ for each user.

However, when the recommendation process does not take into account contextual information, is posible to apply the information about the current (or desired) context c in various stages of the recommendation process.

Adomavicious [**?**] defines three paradigms for the context-aware recommendation process that is based on contextual user preference, the Figure **??** is a graphic

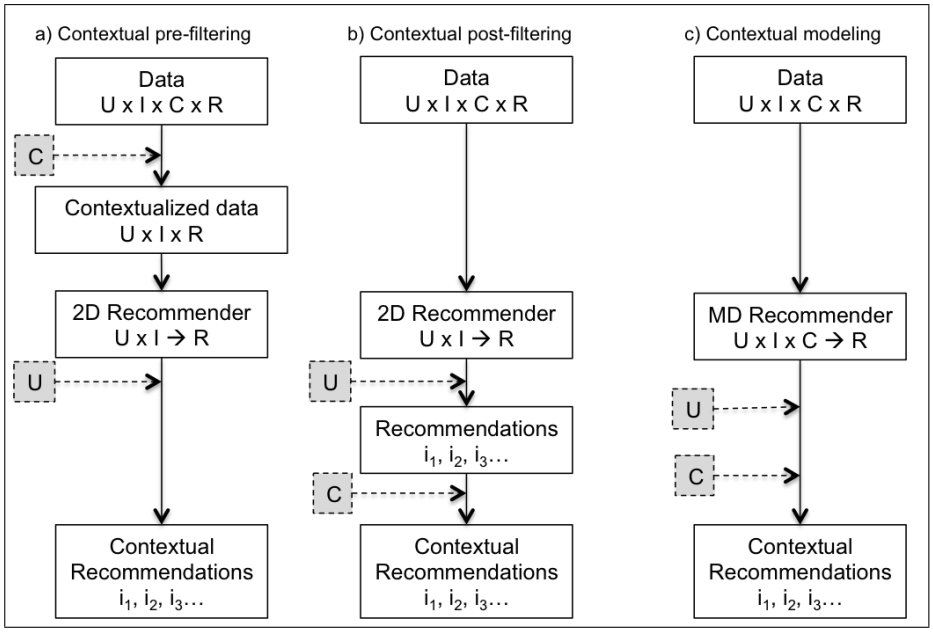


Figure 3.2: Paradigms for incorporating context in recommender systems [**?**].

representation of paradigms and Adomavicious describes them as follows:

- **Contextual pre-filtering (or contextualization of recommendation input).** The approach uses contextual information to select the most relevant 2D data *(Users* x *Items)* for generating recommendations. The major advantage of this approach is that it allows deployment of any of the numerous traditional recommendation techniques previously proposed in the literature [**?**]. 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 a crisp pre-filter because the data was filtered using exactly the specified context (Figure **??**.a).

- **Contextual post-filtering (or contextualization of recommendation output).** In this approach context information in the input data is ignored 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 if on weekends he or she only watches comedies, the system can filter out all noncomedies from the recommended list (Figure **??**.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, regression, and so on)

or heuristic approaches that incorporate contextual information in addition to the user and item data (Figure **??**.c).

Chapter 4

Proposed method

The proposed method is based in a previous architecture presented in [**?**] used in an *e-learning plataform (Protoboard)* where the context involves characteristics referring the student enviroment as well as their performance in previous learning activities.

After that, another recommender system called *Recomet* [**?**] implements the same method using only traditional recommender systems, i.e., it does not use the context in recommendation process to suggest items to users.

Chapter **??** describes some the applications of both pre-filtering and post-filtering paradigms, each one is used in the appropiate domain and taking in account the system goals (see Table **??**). The propose method follows a *post-filtering* approach, to contextualize the recommendations list (see Figure **??**) previously generated using

a hybrid of three recommendation techniques, working simultaneously. Once a user defines the current context, the post-filtering approach adjusts the recommendations for user's context, this process is called "contextualization".

Each recommendation tecnique uses the collaborative filtering rating-matrix to obtain predictions of items, considering variables such as popularity and similarity among users and items. In this section each technique used and components of the method are explained in detail.

Overall, the post-filtering paradigm tries to cover the user's needs in a wide range of situations and it is based on the user's profile information. The chapter describes each component of the method, as well as its functionality and thier interaction with another components.

In order to explain the method, a case of study in a restaurant domain will be used. Then, an brief description of recommendation techniques are also explained:

1. **Expert's fuzzy inference system**, this is a rule based recommender, these rules are defined by an expert in the domain. The fuzzy inference system has the following input 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 represent fuzzy values and 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. For other domains, fuzzy

variables must be identified by a domian expert and expressed as a fuzzy inference system.

2. **Content-based technique** utilizes the item profiles (descriptions) to compare how *similar* is an item with respect to another, i.e. restaurants that are *similar* have equal atributes (same cuisine, atmosphere, price range, etc.) to others that have been rated high by the user. The idea is to find items with similar features to the items that the user prefers or it has high rating in the user profile.

3. **Collaborative filtering technique** is based in the user's preferences to find neighbors that have similars tastes and preferences. Similarity among users is calculated using Pearson Correlation, a threshold upon of 0.8 is used to determine the size of the neighbour. Then, recommendation process considers the opinions of other users that rated restaurants with a similar rating, but only recommends when the item have not been rated by the current user. A threshold of 3.5 (in rating scale) determines the items that will be added in the Top-N list that will be shows for the user.

The last step is the *contextualization*, represented as a *context filter* to obtain the list of contextualized recommendations.

As each technique works simultaneously to obtain recommendations, the hybrid

method allows to generate suggestions even without ratings of user, it means that using content-based technique or the fuzzy inference system, the system faces the cold-start and overspecialization problems using these thechniques, these problems were previously described in section **??** and section **??**, respectively.

For the case of study, the proposed method was tested in the restaurants domain for the city of Tijuana, a context-aware recommender system was developed in order to verify its performance with real users interacting with the system and to test its functionality.

The application was developed using technologies as *Django Framework*, *Python language*, *Bootstrap 3.0*, *JavaScript*, *JQuery* and *PostgreSQL database*. The complete description of the system will be show in a posterior section.

To evaluate the performance of the proposed method in the case of study several experiments were conducted, the algorithms were tested using contextual datasets and the number of contextual factors used varied according the information provided in the datasets.

The goal of the experiments were to observe the role of contextual data in the performance of the algorithms and to identify what contextual factors are more relevant for users in a specific situation, how recommendations are improved using context and, how improves the accuracy in recommendations.

4.1 Data Models

The data model defines an abstraction of the context. An effective on- line recommender system must be based upon an understanding of consumer's preferences and successfully mapping potential products into these preferences [**?**].

Pan and Fesenmaier [**?**] argued that this can be achieved through the understanding of how consumers describe in their own language a *product*, *a place*, and the overall *experience* when they are consuming the product or visiting the place. It is observed that the language of the consumers is often imprecise, with attributes values often expressed in fuzzy terms.

4.1.1 Restaurant Data Model

Following the case of study mentioned in previous section, the restaurant model will be explained.

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 desition making [**?**].

While food quality and food type have been perceived as the most important vari-

ables for consumers' restaurant selection, situational and contextual factors have been found to be important also. Due to this Kivela [**?**] identifies four types of restaurants: 1) *fine dining/gourmet*, 2) *theme/atmosphere*, 3) *family/popular*, and 4) *convenience/fast-food*; on the other hand Auty [**?**] identifies four types of dining out occasions: 1) *celebration*, 2) *social occasion*, 3) *convenience/quick meal*, and 4) *business meal*.

Taking in account the context, the model proposed was definded with 55 attributes about restaurants.

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*, 8) *atmosphere type*, and 9) *cuisine type*.

An example of a restaurant model is depicted in Figure **??** 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 other information such as *users's reviews*, *ratings average*, and *geographycal location.*

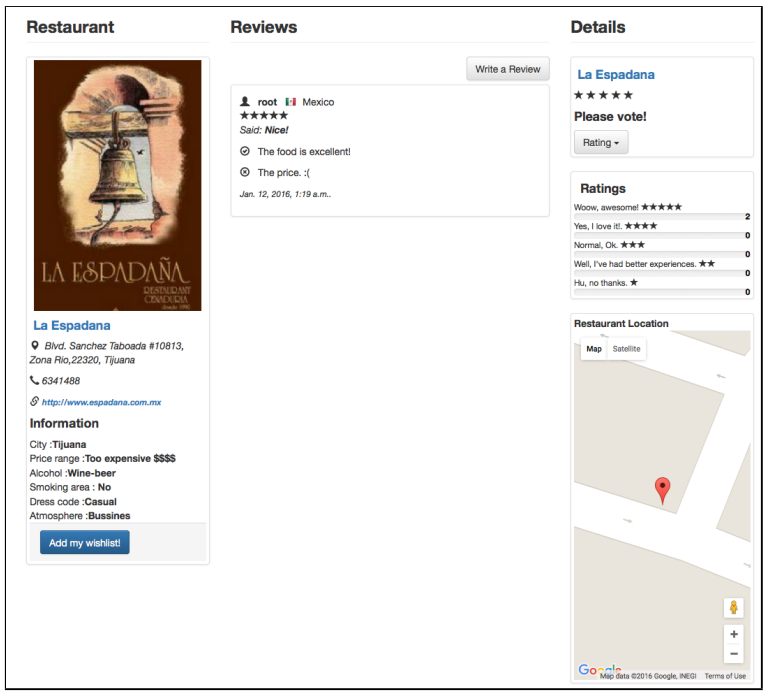


Figure 4.1: User interface of the restaurant model.

Contextual Factors Data Model

The data model was physically implemented in a PostgreSQL database model, then the information in the context-aware recommender system was managed in a schema of a relational database.

Technical support about installations of dependencies and the open source application are described in appendix **??**. The restaurant data model is depicted in the Figure **??**, 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 restric-

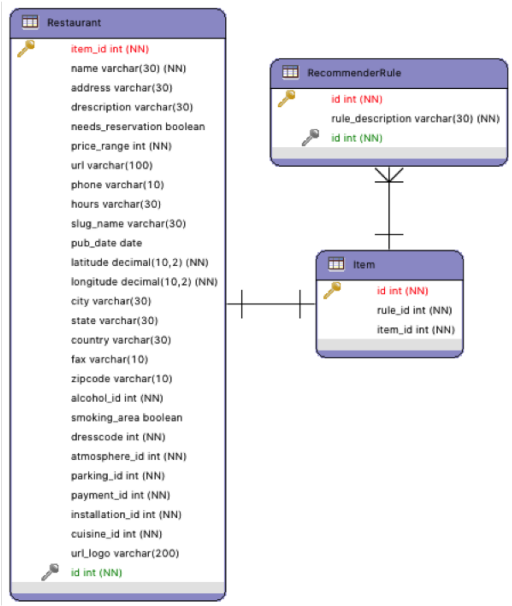


Figure 4.2: The restaurant data model.

tions for item recommendations. A large view of all the entities related is depicted in the whole scheme refered in Figure **??**.

The entities related correspond to the proposed contextual factors, are defined as follows:

- **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 *cuisine types* 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 that defines if a restaurant has a smoking area into its installations. These are the fuzzy contextual factors proposed in this case study.

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 $\neq$.

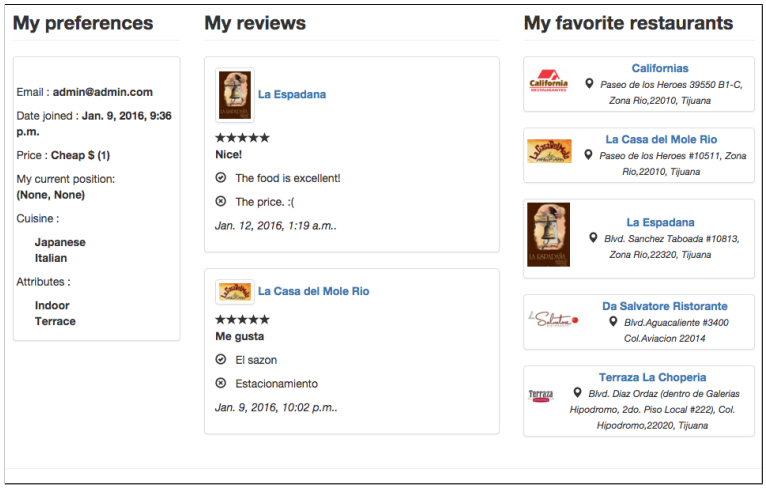


Figure 4.3: Example of user interface for user profile.

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.

User Data Model Implementation

¡¡¡¡¡¡¡ HEAD The user data model in postgreSQL is represented in the Figure **??**, 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.

In the application, user profile has contextual information such as: 1) price range, 2) current location, 3) cuisine types, 4) attributes or characteristics of restaurants that the user want, ¡¡¡¡¡¡¡ HEAD 5) list of favorite restaurants.

The user profile is stored in database and it is available for queries upon request, and it can be changed by users many times in a session.

The information used to make recommendations is the most recent. The user interface is represented in Figure **??**. 5) list of favorite restaurants. The user profile is stored in database and it is available for queries upon request, and it can be changed by users many times in a session. The information used to make recommendations is the most recent. The user interface is represented in figure **??**. ¿¿¿¿¿¿¿ origin/-master The *Friends entity* represents the social aspect in 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, ¡¡¡¡¡¡¡ HEAD attribute groups corresponds to restaurant model mentioned (section **??**). A total of *55 attributes* (or characteristics) could be contained in *userProfile*. The domain values for price, cuisine and attribute groups are following:

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

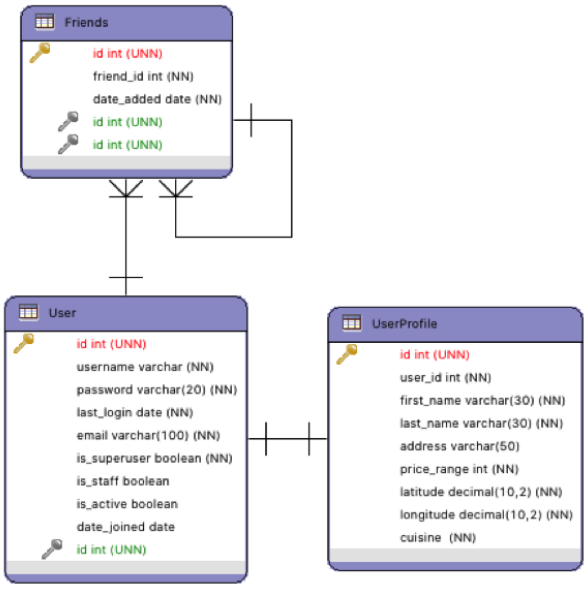


Figure 4.4: The user data model.

- **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 **??**. This model involves the whole relational database for context-aware recommender system, as well as the entities and relations among them.

Contextual information is also stored in the following entities: *CurrentLocation, DistancePoi and Ratings.*

In the case of *Reviews entity* only describes the user's opinion about visited restaurants, this information contributes to have additional information about recent preferences of diners, however, this process is not implemented in the system still.

CurrentLocation entity stores the geographical position of user to get a *"nearby recommendation"*, the system locates restaurants around two kilometers from the user position, for instance the system can takes a user position as *latitude:32.529865* and *longitude: -116.986605*, this information is stored to calculate distances from this point.

The geographical position is changed frequently, in this manner, it allows the adaptation for each ¡¡¡¡¡¡¡ HEAD particular user situation.

Distance PoI entity stores the distances (kilometers) between the user and restaurants, this information is used to calculate a *"nearby recommendation"*, where each recommended restaurant must not be over the threshold defined, then in this case,

the system considers only nearby items, into two kilometers around the user position.

Finally, *Rating entity* represents the user's preferences in a vector of ratings that could be increased in time and the user's preferences patterns thatcould be changed in time also.

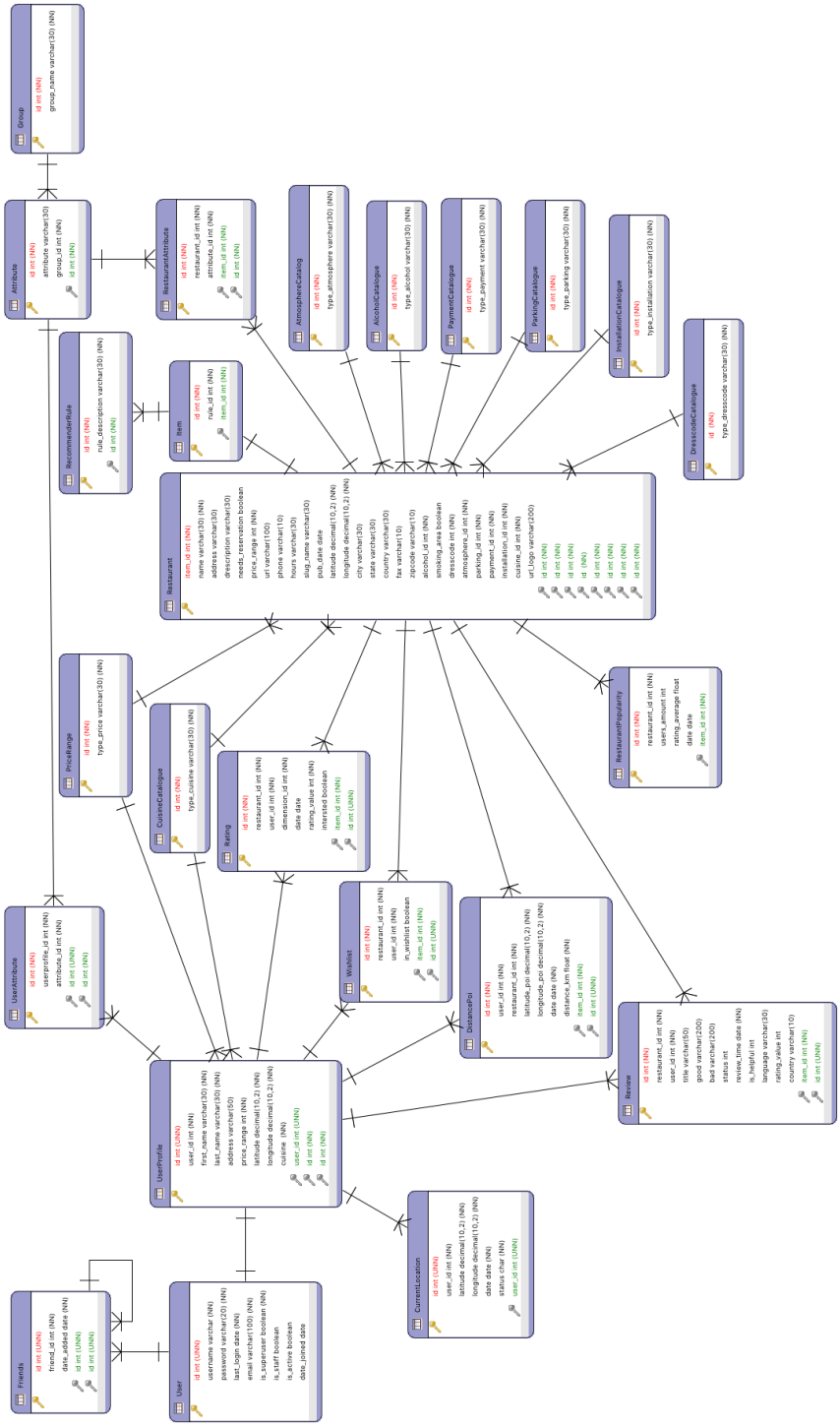


Figure 4.5: The data model of context-aware recommender system.

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.

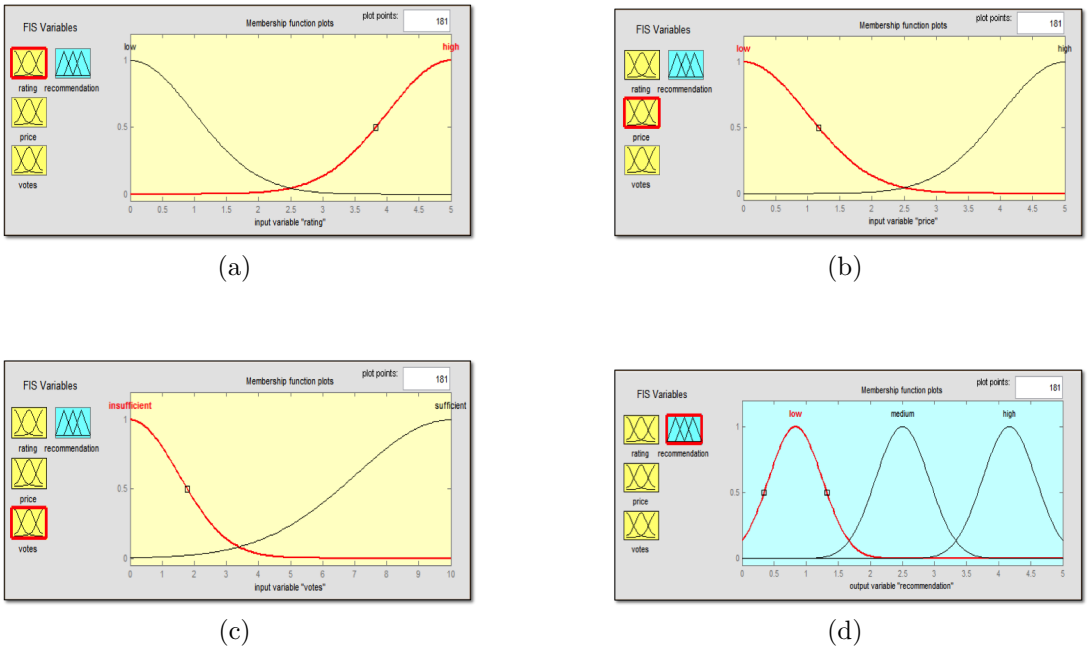
Fuzzy logic is a computational intelligence technique that allows to use information with a high degree of inaccuracy; this is the difference with the conventional logic that only uses concrete and accurately information [**?**].

In this work, fuzzy logic is used to model fuzzy variables that highligh in the popularity of a restaurant.

The Expert module (fuzzy inference system) generates recommendations when the other recommendation techniques (collaborative filtering, content-based) are not returning results because of the cold start problem.

The fuzzy inference system proposed has three **input variables** (as in the previous work [**?**]): 1) *rating* is an average of ratings that has a particular restaurant inside the user community; the domain of variable is from 0 to 5 and contains two membership functions labeled as *low* and *high*(Figure **??**), 2) *price* represents the kind of price that has a particular restaurant; the domain of variable is from 0 to 5 and contains two membership functions labeled as *low* and *high* (Figure **??**), 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 **??**), the domain of variable is from 0 to 10 and contains two 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 from 0 to 5 and contains three membership functions labeled as *low*, *medium* and *high* (Figure **??**). The proposed fuzzy inference system (Figure **??**) represents



(a) (b)

(c) (d)

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

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

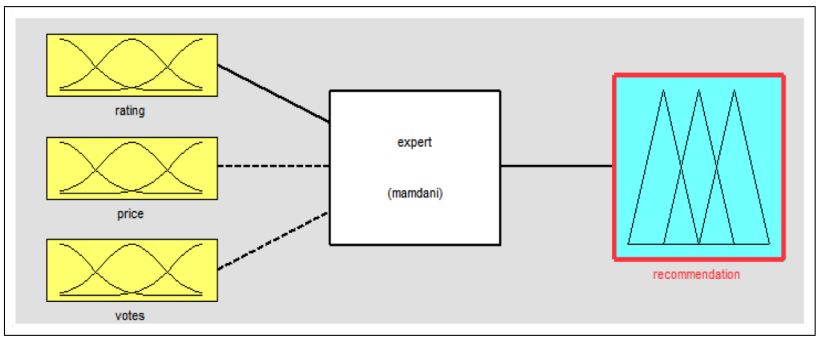


Figure 4.7: Fuzzy inference system of the expert.

weights that influence in the final recommendations. The fuzzy inference system uses five 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 Weighted Average

The main goal of this fuzzy inference system is to assign weights for each recommendation list.

The recommendation technique is based in the amount of available information stored, because each technique uses this information to provide a list of restaurants as well as a weight for each one, therefore, these are 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 **??**. There are three membership functions for inputs and three for outputs.

The input variables are: *userSimilarity, restaurantSimilarity and Participation* and are depicted in Figure **??**. The Figure **??**.a and Figure **??**.b are in a range from 0 to 1 to define the similarity average among users and restaurants, respectively. The Figure **??**.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 **??**). On the other side, the output variables are: *Expert*, *RestaurantProfile* and *Correlation*, these are depicted in Figure **??**.

The Figure **??**.a represents the weight for expert recommendation list, Figure **??**.b represents the weight of the content-based list and Figure **??**.c represents the weight

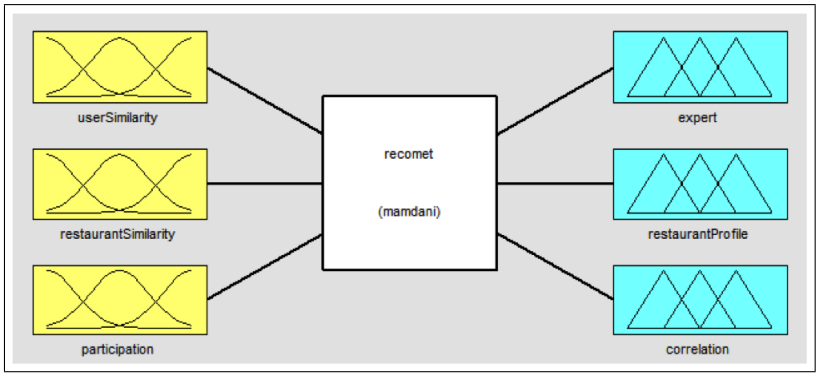
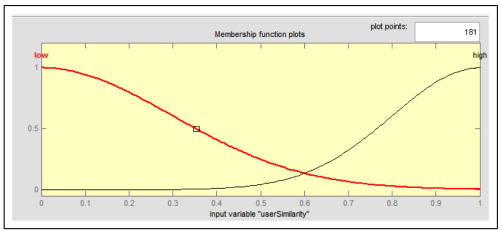
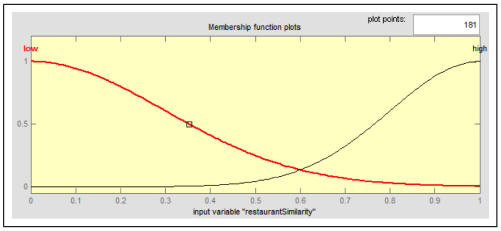


Figure 4.8: Fuzzy Inference System to assign weights.

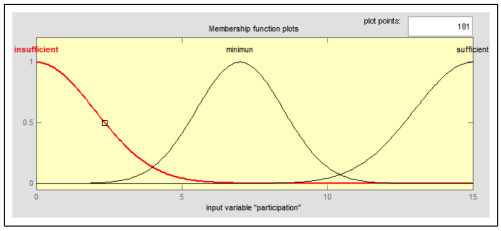
of collaborative recommendation list, their membership functions are in a range from 0 to 1 to get the value. Taking in to account the input variables, the rules utilized



(a)



(b)



(c)

Figure 4.9: The Gaussian membership functions of input variables.

to infer the output values are following:

1. *If* ***userSimilarity*** *is low and* ***restaurantSimilarity*** *is low and* ***participa-***

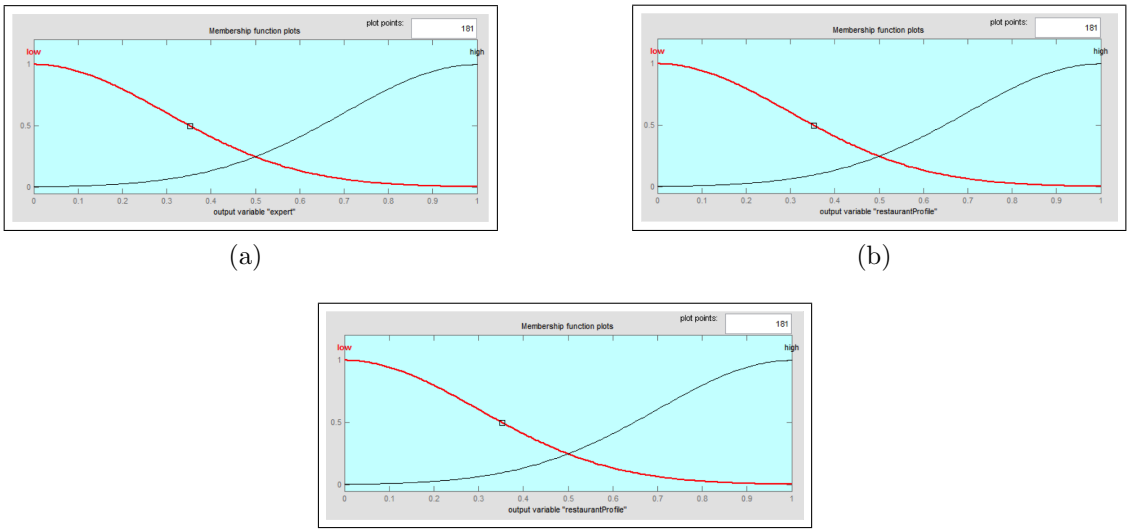


Figure 4.10: The Gaussian membership functions of output variables.

tion *is insufficient then* ***expert*** *is high,* ***restaurantProfile*** *is low,* ***correlation*** *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,* ***correla-***

tion *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,* ***corre-***

lation *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.*

At the end, a *weighted average* allows to get predictions for each restaurant in the list of recommendations. In this way, for instance if the **expert** has a weight of **0.5**, the **restaurantProfile** is **0.8** and the **correlation** is **0.6**, the system uses these weights to calculate the final prediction for a particular restaurant using the formula of the weighted average (Equation **??**):

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} = \frac{x_1 w_1 + x_2 w_2 + x_3 w_3 + ... + x_n w_n}{w_1 + w_2 + w_3 + ... + w_n} \tag{4.1}$$

Where the set X=$\{\ x_1, x_2, x_3, ..., x_n\ \}$ represents the predictions and the set $W = \{\ w_1, w_2, w_3, ..., w_n\ \}$ represents the weights given for the fuzzy inference system for each recommendation technique. Then, applying the formula, the system can

process it in the follow manner (Equation **??**):

$$\bar{x} = \frac{(0.5 * 4.0) + (0.8 * 5) + (0.6 * 4.5)}{(0.5 + 0.8 + 0.6)} \tag{4.2}$$

The prediction corresponds the final value of recommendation for the item, and is used to include it or exclude it of the list of recommendation if is not over the threshold.

Then, for this case, the prediction is **4.57**, it means that this restaurant will be in the recommendation list of the user, subsequently, the recommendations list will be contextualized.

4.4 Contextual recommendation

The interface of the system (Figure **??**) allows the collection of 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 as a filter for the final recommendations list. For example, geographical location is used to get restaurants around two kilometers of distance, next, the list of nearby restaurants is displayed for the user.

If the context-aware recommender system considers other attributes, as type of price or the type of cuisine preferred by the user, the system gets restaurants matched

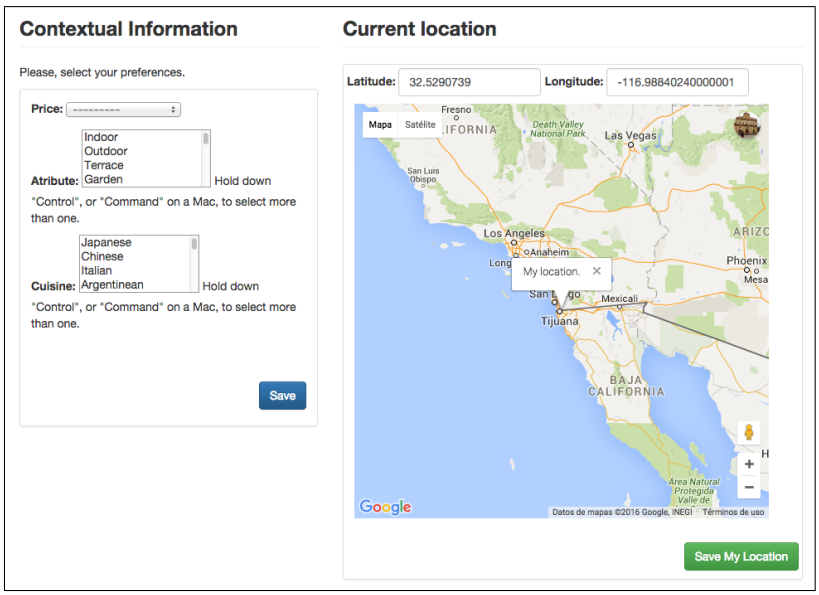


Figure 4.11: System interface to collect contextual information.

in the context especified by the user at this time. In the attributes box, the user can choose any preference about what things are importants to select a restaurant. The features are collected from the dataset of Tijuana restaurants.

In the cuisine box, the user choosen his/her favorite cuisine, it can be one or more cuisines such as in attributes also. 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 **??** shows recommendations: *1) Expert, 2) Content-based, 3) Collaborative filtering and 4) Nearby.* Each one was explained above, except the nearby recommendations.

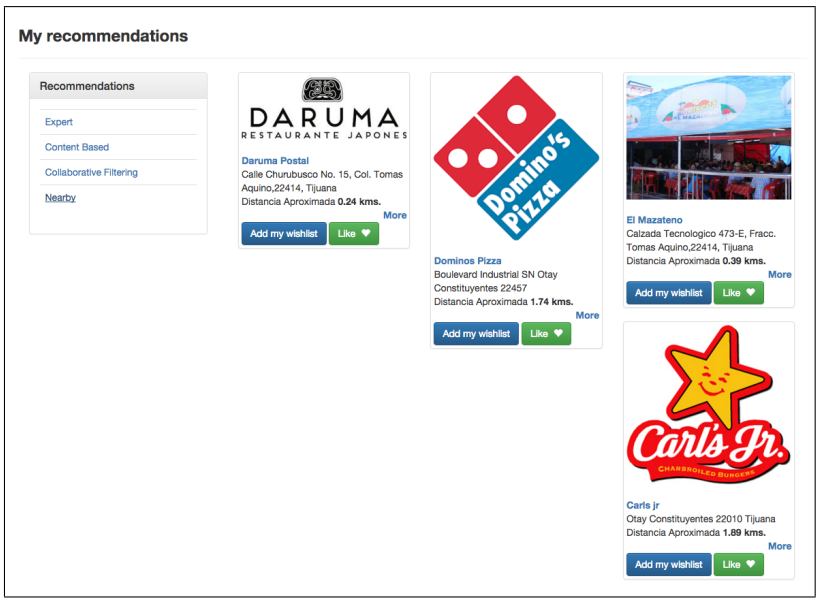


Figure 4.12: System interface of recommendations for the user.

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 two kilometers around the user position to determine what restaurants will be recommended. The geographical position is obtained through the Google Maps API.

4.5 Method scheme

The general scheme of the proposed method is depicted in the Figure **??**. In the first part, the three techniques of recommendations are suplied by the rating matrix:

1) **Fuzzy inference system of the expert** obtains the input values through rating

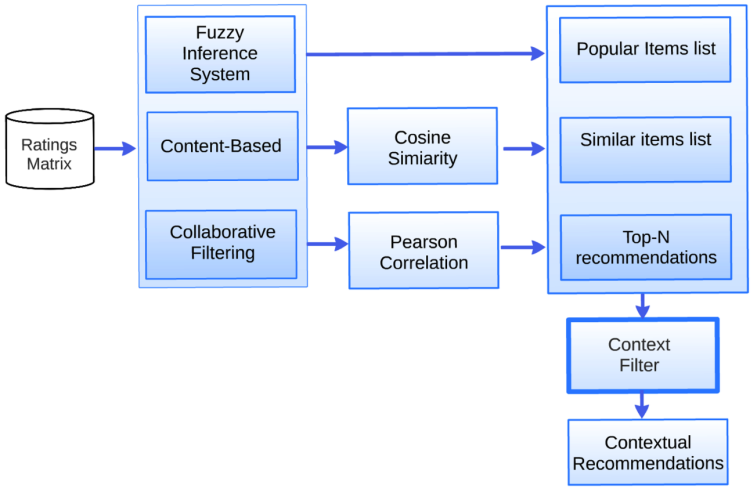


Figure 4.13: Scheme of the proposed method.

matrix taking users ratings, so it can infer the output value. 2) **Content-based** uses the high ratings in the user profile and item profiles to calculate similarity between them and do that using Cosine similarity measure. 3) **Collaborative filtering** is based in user profiles content in ratings matrix, it calculates the similarity among users using Pearson correlation, subsequently, a list of neighbors is obtained to use their preferences to calculate predictions.

The second part shows three lists obtained, the fuzzy inference system is used to assign weights for each one in order to determine the prediction, these weights are important to calculate a prediction because is obtained through the weighted average.

Later, the recommendation lists are reduced when the context filter is applied, i.e., the contextualization process is applied to get recommendations adjusted in the

user context. Finally, the contextual recommendations list is displayed in the user interface (see Figure **??**).

Chapter 5

Case of study

In this chapter we present the relevant experiments that were done to test the proposed method. Subsequently, it shows the implementation of the method in a restaurant recommender system used as case of study, the context was integrated into the recommender system when we consider the contextual factors. A wide explanation of the system interfaces and its functionality is explained also.

5.1 Experimental settings

In this particular experimental scenario, the basic guidelines proposed in Adomavicius [**?**] are followed, and they are briefly explained next.

- **Hypothesis:** before running the experiment we must form an *hypothesis*. It

is important to be concise and restrictive about this hypothesis, and design an experiment that tests the hypothesis. For example, an hypothesis can be that *algorithm A* predicts better user ratings than *algorithm B*. In that case, the experiment should test the prediction accuracy, and not other factors.

- **Controlling variables:** when comparing a few candidate algorithms on a certain hypothesis, it is important that all *variables* that are not tested will stay fixed. For example, suppose that we wish to compare the prediction accuracy of movie ratings of *algorithm A* and *algorithm B*, that both use different collaborative filtering models.

- **Generalization power:** when drawing conclusions from experiments, we may desire that our conclusions generalize beyond the immediate context of the experiments. When choosing an algorithm for a real application, we may want our conclusions to hold on the deployed system, and generalize beyond our experimental data set. Similarly, when developing new algorithms, we want our conclusions to hold beyond the scope of the specific application or data set that we experimented with. It is important to understand the properties of the various data sets that are used. Generally speaking, *the more diverse the data used, the more it can generalize the results.*

5.1.1 Off-line experiments

An off-line experiment is performed using a pre-collected dataset of users' selections or ratings. By using this dataset we try to simulate the behavior of users that will interact with the recommender system.

In doing so, it is assumed that the user behavior when the data was collected will be similar enough to the users' behavior when the recommender system is deployed, so that we can make reliable decisions based on the simulation. Off-line experiments are attractive because they *require no interaction with real users*, and thus it allows to compare a wide range of candidate algorithms at a low cost.

The downside is that it can answer a very narrow set of questions, typically questions about the prediction of an algorithm. The goal of the off-line experiments is to filter out inappropriate approaches, leaving a relatively small set of alternatives algorithms for subsequently to be tested for the more costly user studies or on-line experiments [**?**].

The next sections show a comprehensive description of the experimental setup for experiments, as well as results obtained in the experiments. Each method was tested using contextual datasets in the domain. This chapter ends with the description of the functionality of the prototype used to validate the method utilized in the experiments.

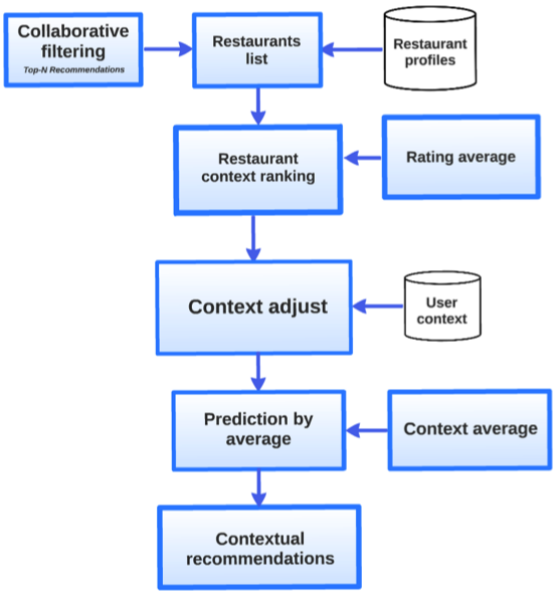


Figure 5.1: The post-filtering approach for Tijuana restaurants.

5.2 Restaurants recommendations

In earlier implementations of the context-aware recommender system several experiments were done to test the behaviour of algorithms and the performance in the proposed method. For the experiment the proposed hypothesis is *The accuracy of recommendations is improved using context factors in the recommendation process.* To validate the hypothesis the next test was realized. A first experiment described in [**?**], presents a contextual recommender system using the post-filtering approach and collaborative filtering technique in a restaurant domain. Collaborative filtering works to get a Top-N list utilized to adjust it in the context.

Later, the Top-N list is obtained and the restaurants are adjusted to make ranking

Table 5.1: Questionnaire applied to collect contextual dataset.

Question	Answers
1.What is your occupation?	1. Student 2.Faculty
2.According to 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 devices do you use	1.Smartphone 2.Tablet 3.Laptop 4.PC
4.What Operating System do you use?	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

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* that a restaurant has in the current context (that is the mean of user ratings) and 2) *the rating* predicted by the collaborative filtering algorithm.

Top-N list contains the restaurants with highest predictions, so each restaurant is adjusted for the user's context and listed in contextual recommendations; the process is depicted in Figure **??**.

In order to validate the proposed approach, data about restaurant preferences of users in different contexts was used. The study subjects were students with a

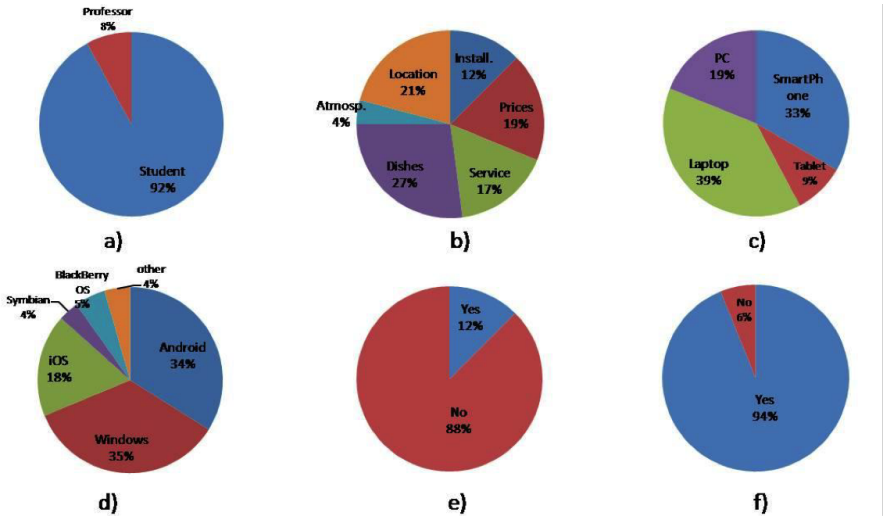


Figure 5.2: The chart cakes show the users preferences for questions from one to six.

major in engineering and a graduate program and faculty 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 most often used by them. The *questionnaire* consisted of *eigth questions* and also they were asked to rate any number of restaurants from a list of 40 restaurants. Each of the restaurants chosen, was rated six times one per proposed context, a matrix rating with *1,422 ratings* was collected. The questions are shown in Table **??**.

The reason for allowing users to chose what restaurants to rate it to give them the same liberty that they have when visiting a web or mobile application. The users' answers from question one to question six are represented in the Figure **??**. Figure

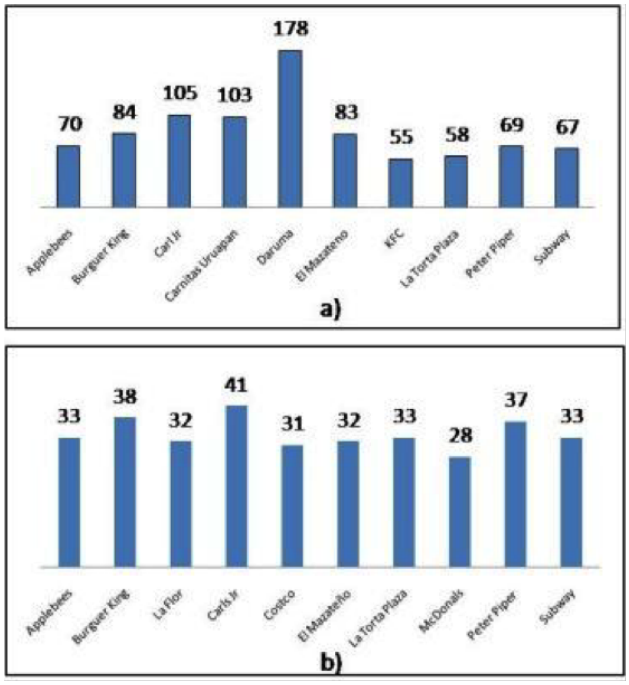


Figure 5.3: The chart of users preferences for questions seven and eigth.

??*a* represents the percentage of surveyed students and teachers; Figure **??***b* the percentage of the element that users consider the most important to visit a restaurant; Figure **??***c* represents the preferences of devices when are using Internet for restaurant recommendations; Figure **??***d* represents the percentage of operating system that often used, Figure **??***e* shows the percentage of users that use the Internet to search restaurants in Tijuana; and Figure **??***f*, shows the percentage of users that would like using a restaurant recommender system of Tijuana.

For questions seven and eigth only the top-ten restaurants are shown, without/with the contextual situation. In Figure **??***a*, the favorite restaurant is **Daruma**(178

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

votes), whereas in Figure **??***b*, **Daruma** does not appear in the top-ten.

When considering the context *midweek*, the favorite restaurant was **Carl's 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 six contexts mentioned correspond to combinations of contextual factors shown in Table **??**. Mean absolute error obtained was **0.5859** in contextual recommendations. The observation for this result is that using a small dataset the performance of the method proposed is limited, the cold-start problem affects the accuracy because of the data scarcity.

5.3 Hotels recommendations

A second experiment using TripAdvisor's dataset was cunducted. For this case, the proposed method consisted of three algorithms to recommend: *fuzzy inference sys-*

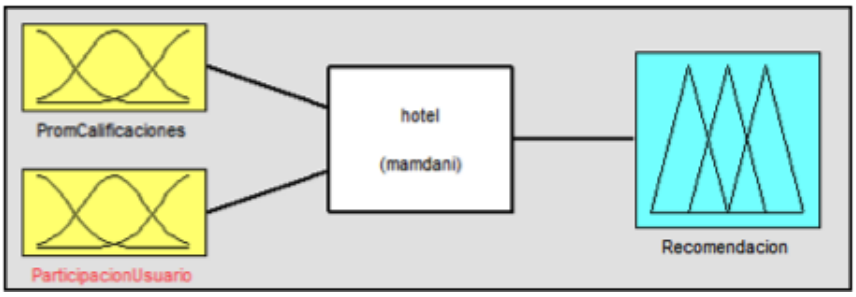


Figure 5.4: Fuzzy inference system.

tem, *collaborative filtering* and *content-based*. Each one uses the ratings matrix to get recommendations.

The contextual recommender system uses *post-filtering* approach [**?**] for adjust recommendations in context such as restaurants. The recommendation by popularity is through the fuzzy inference system depicted in Figure **??**, 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.

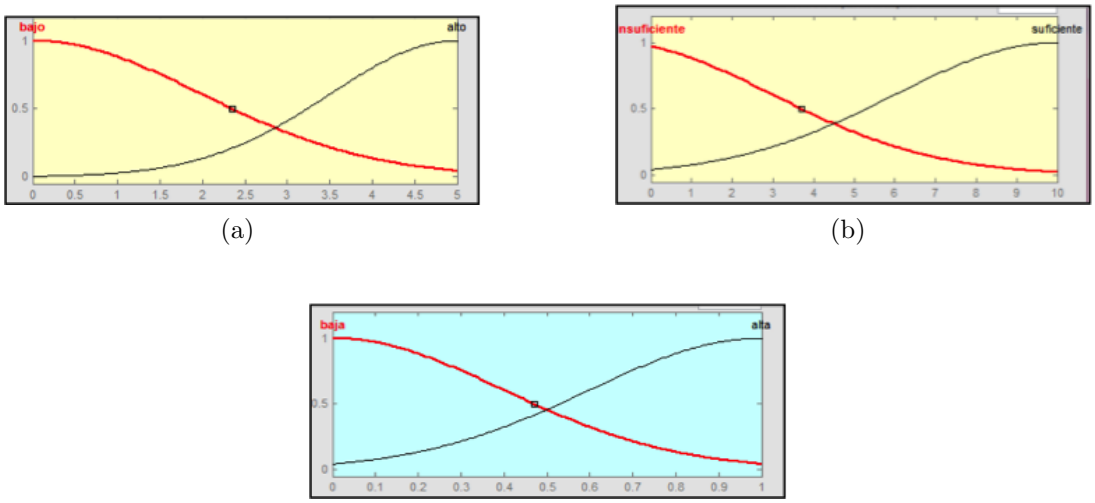
The output represents how matter each item into the users community, i.e. if it is a popular item between users.

The dataset used to evaluate the algorithm was TripAdvisor in two versions downloaded [**?**], this datasets was used in [**?**] and [**?**] 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 FIS has Gaussians membership functions and are depicted in Figure **??**. The



(a) (b) (c)

Figure 5.5: 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 crisp 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.*

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.

Next, the outputs of every recommender technique 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 such as in Table **??**, and compares recommended items to filter those items that are adjusted to the user context.

Table 5.3: Example of contextual ratings in the user profile.

User profile		
Item	Rating	Context
La Casa del Mole	5.0	Midweek
Daruma	4.0	Weekend
Daruma	5.0	Midweek
Carl's Jr.	3.0	Weekend

The context filtering is the last step before to get the recommended items.

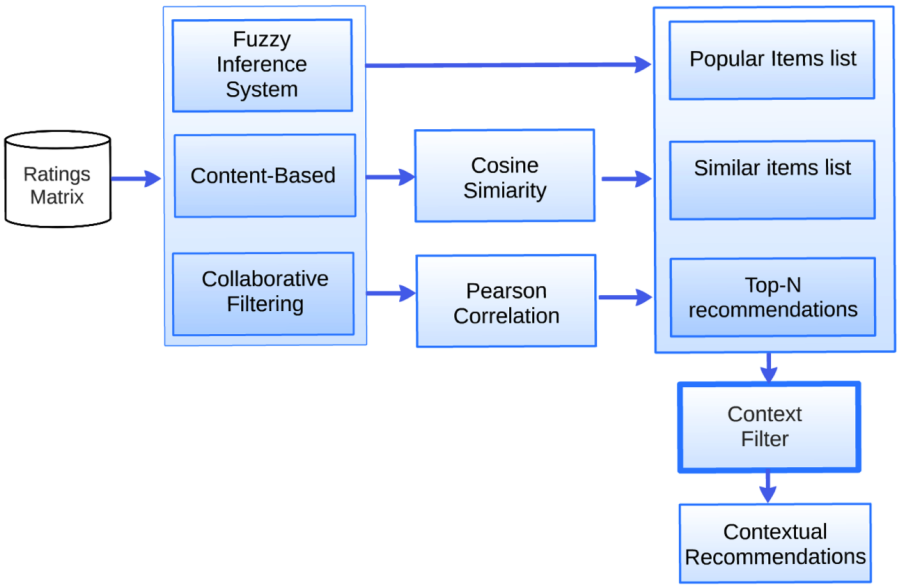


Figure 5.6: Recommender system methodology.

Table 5.4: Datasets description.

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

The schema of architecture for context- aware recommender system is depicted in Figure **??**.

Two tests were performed using TripAdvisor dataset, Table **??** 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 **??** the comparison shows that the algorithm has a acceptable performance, i.e., the

Table 5.5: Comparison of root mean square error.

Dataset	Algorithm	Error
TripAdvisor v2	collaborative filtering + Post-filtering	0.504
	collaborative filtering	0.994
	Pre-filtering + Relaxation	0.985

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 [**?**] that is evaluated with the same dataset.

The cosine similarity plays an important role in content-based because 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 **??**.

Fuzzy inference system can provides a list of popular items for each dataset, recommendations through averages 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 has biases that affects the results.

Table 5.6: Level of similarity among items in datasets.

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

5.4 Context-aware recommender system prototype

This section presents a context-aware recommender system prototype. The backend have been explained in chapter **??**, in sections **??** and **??** talk about experiments realized using the recommendation techniques proposed.

To develop the prototype was used python language, technologies as Django Framework 1.7, JavaScript, JQuery, Ajax, HTML5, Bootstrap 3.0 and PotsgreSQL for database.

Some dependencies and libraries were used also, it can review links of downloads in appendix **??**.

5.4.1 User Interfaces

The system starts in a landing page, the user should do *Sign in* or *Sign up* to create a new user to enter the home page.

Landing page contains the *Best restaurants* rated and *Featured top list restaurants* proposed by users, such as shows it the Figure **??**, the restaurants are updated while users add ratings, if the tendency is changed, the section displays the change in the

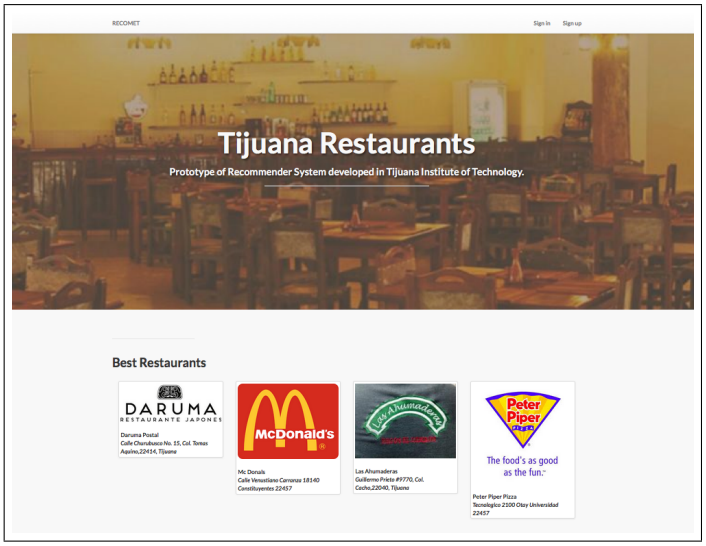


Figure 5.7: Landing page interface.

tendency.

The aim is to provide information for old and new users,the prototype tries to keep updated the current preferences of users constantly.

Home Page interface

Home Page (Figure **??**), shows the main menu, tags for user preferences, all filters to start the search of restaurants and the most popular restaurants in the community. Users can start exploring filters to find restaurants under their own criteria, each restaurant has a profile with complete information about the characteristics and opinions of other users.

When users click the restaurant picture the system redirects to the profile, for in-

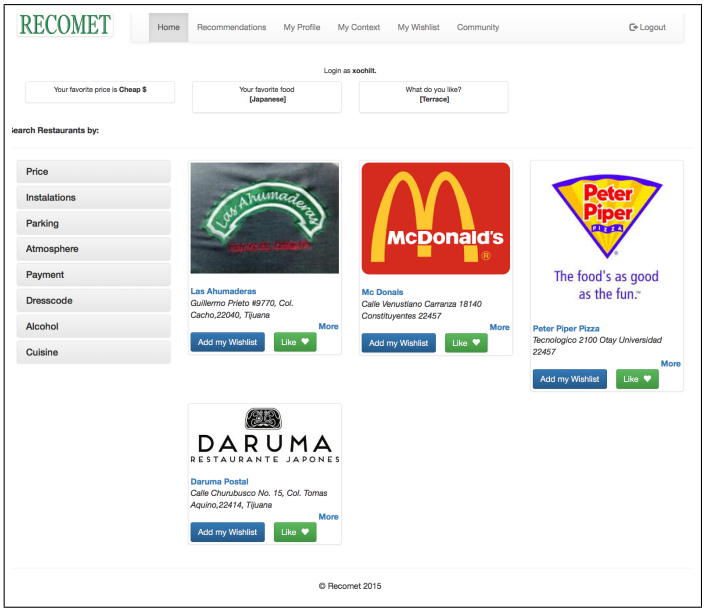


Figure 5.8: Home page interface.

stance Figure **??** shows the profile of Daruma restaurant, the Figure shows the general information of the restaurant, reviews or personal opinions of users that visited the restaurant, details about ratings, the chart of ratings and the user location using Google maps services. It is also added the button to *add wishlist*, this element will be explained in a posterior section.

My Recommendations interface

In recommendations interface users have the options to get recommendations such as was described in chapter **??**, four recommendation techniques works to suggest restuarants for users. Here, users can choose the prefered option to get information in the screen.

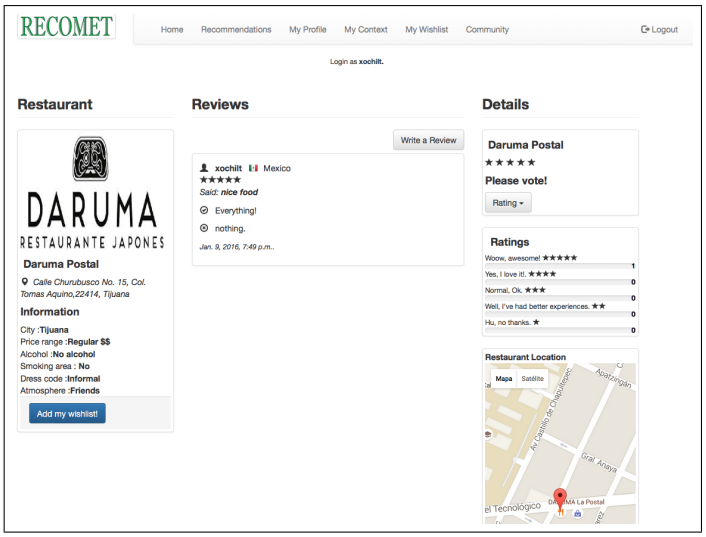


Figure 5.9: Restaurant profile interface.

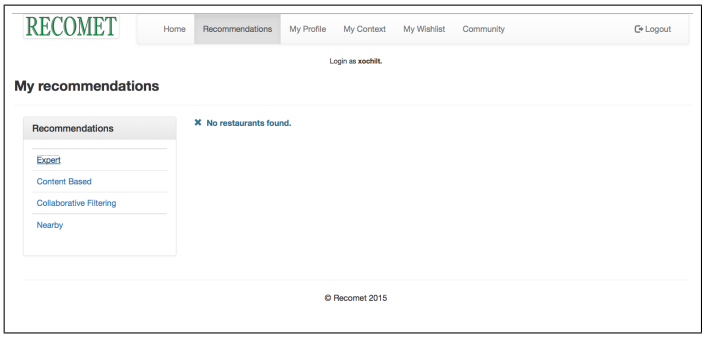


Figure 5.10: Expert recommendations interface.

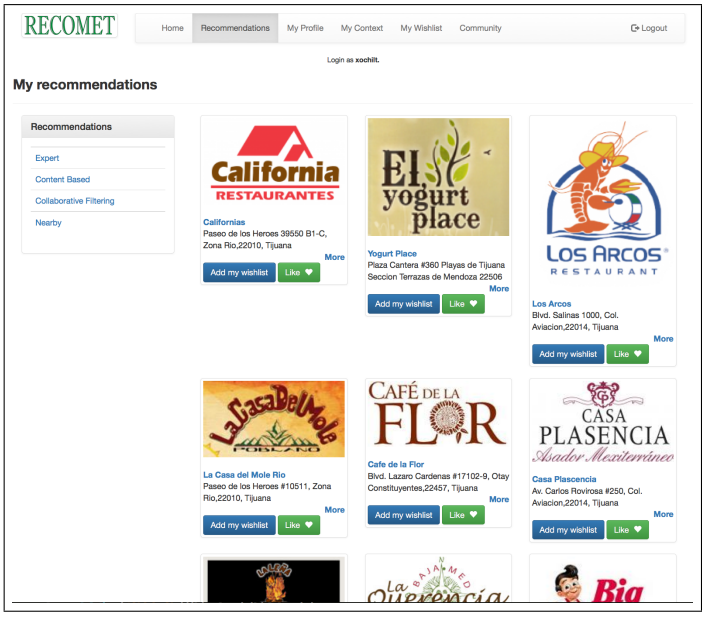


Figure 5.11: Content-based recommendations interface.

For instance, Figure **??** shows that the expert can not to recommend whether users dont have participations, because of expert's recommendation is based in fuzzy rules that involves the popularity of items (input variable), if the information is sparse the prototype can not get results and only shows an alert message. On the contrary case, it shows a list of popular restaurants.

Content-based recommendation finds the similar restaurants to the favorites of the user. In fact, makes a comparison of all restaurants and gets the more similars to display in the screen.

Figure **??** shows the results of the search, large amount of restaurants are showed because of the similarity among restaurants, this is the most efficiently way to get

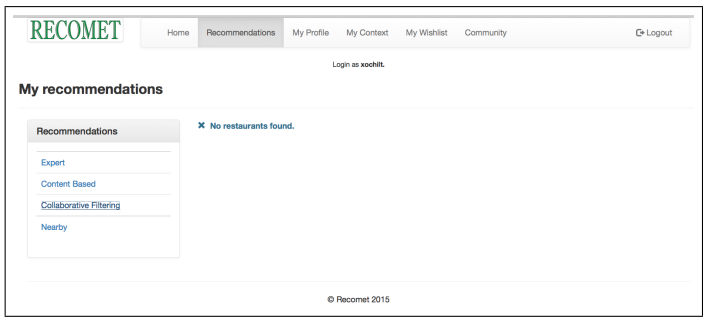


Figure 5.12: Collaborative filtering recommendations interface.

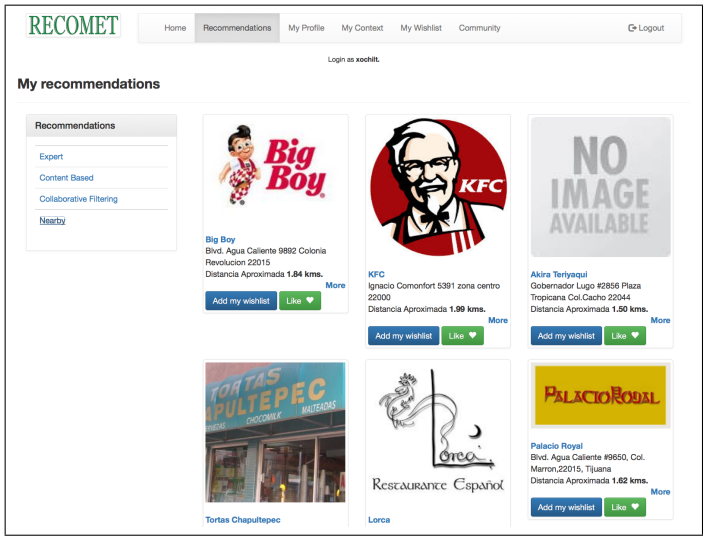


Figure 5.13: Nearby recommendations interface.

recommendations when the system has not enough information about user preferences and it the users community is small. It is not functional whether the user has not at less one vote with high rating (five stars).

Collaborative filtering recommendation is depicted in the Figure **??** this technique is based in users' opinions, if the system has not information of users, or whether the users community is small, cold-start problem highlights in results.

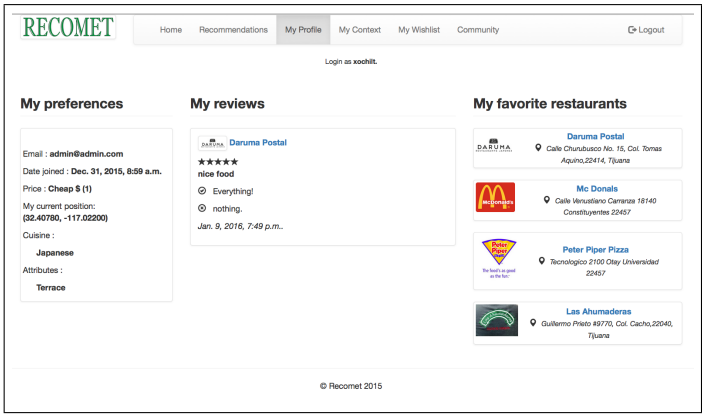


Figure 5.14: My Profile interface.

As in expert recommendations if the users participations is low or null, the results of recommendations will be empty and the alert message appears in the screen, else, the restaurants suggested will be displayed in the screen.

Nearby recommendations (Figure **??**) provides recommendations when the user previously specifies a geographical position. The distance to considers restaurants from the current position to two kilometers around. Then, recommendations are displayed in the screen if the location is provided in the context, else, an alert message notifies the lack of user current location.

My Profile interface

My Profile interface shows all the information of the user divided in three sections: my preferences, my reviews and my favorite restaurants as shown in Figure **??**.

The preferences represent the basic information and tastes of a user, it contains the

tastes and preferences of the user, some factors as price, position, cuisine and attributes of restaurants could be changing continuosly, it depends how the user want manage its information.

The prototype uses this information every time that a request of recommendation is done. The My reviews section shows all the reviews that the user typed, each review is stored in database and the user cannot delete it.

The stars represent the level of satisfaction of the user when he/she visits the restaurant, each review highlights the good and the bad things that the user perceives in the visit.

This information is valuable but, unfortunatly this prototype not uses it to infer possible tastes and preferences and subsequently, recommendations. This process might be a functionality integrated in the future.

My favorite restaurants section, displays restaurants rated with five stars. It means that are the more important options of the user to visit in this moment and in the future. These restaurants are related to the content-based recommendation because are the restaurants with high rating of this particular user.

My Context interface

My Context interface is divided in two sections: contextual information and current location as in the Figure **??**. Contextual information represents the user preferences,

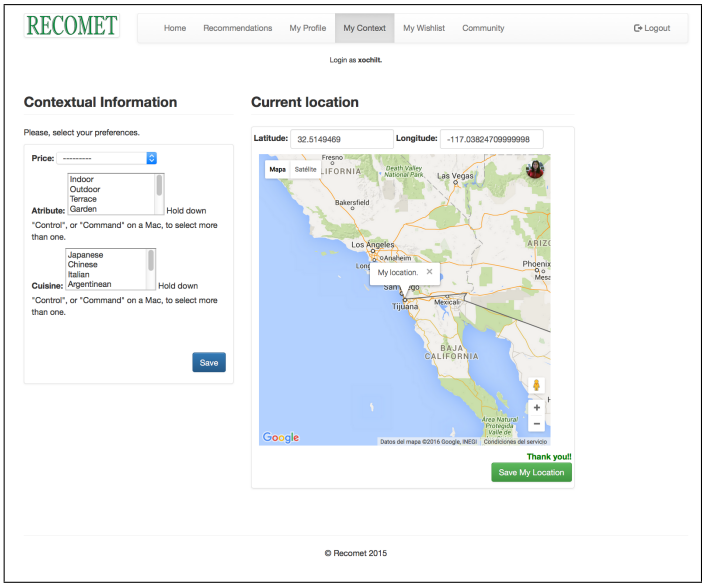


Figure 5.15: My Context interface.

for instance the price range that wants for a specific ocasion, the attributes are the characteristics of restaurants, so it selects as many as the user prefers, the limit is the number of characteristics displayed.

The same is for cuisines, a total of 30 type of cuisines were proposed considering the list of restaurants in the prototype, it can select as many as the user want.

All this information is stored and displayed in home page also in order to help the user to remember which is the information that the prototype is using to display recommendations.

Current location section shows the Google map to display the current location, previously the user ought to confirm that wants to share the current location.

When user click in the *Save My Location button*, the content in the box of latitude

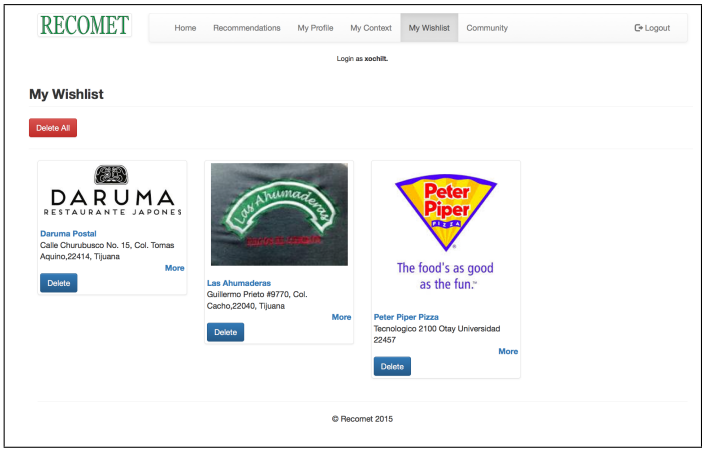


Figure 5.16: My Wishlist interface.

and longitude will be stored in the database(alert message confirms the action) and the status will be modified when user location changes, then, all the locations stored in database become user's historical information.

My Wishlist interface

My Wishlist interface contains the restaurants that are considered by the user as future options to vist (Figure **??**) or simply to have available information for a particular ocasion.

The wishlist allows the manipulation of restaurants, it means, user can add or delete restaurants as many as he/she wants.

The wishlist could be empty or full of restaurants, this information does not affect any functionality of the prototype.

Making a comparison with others platforms (for instance Amazone), this information

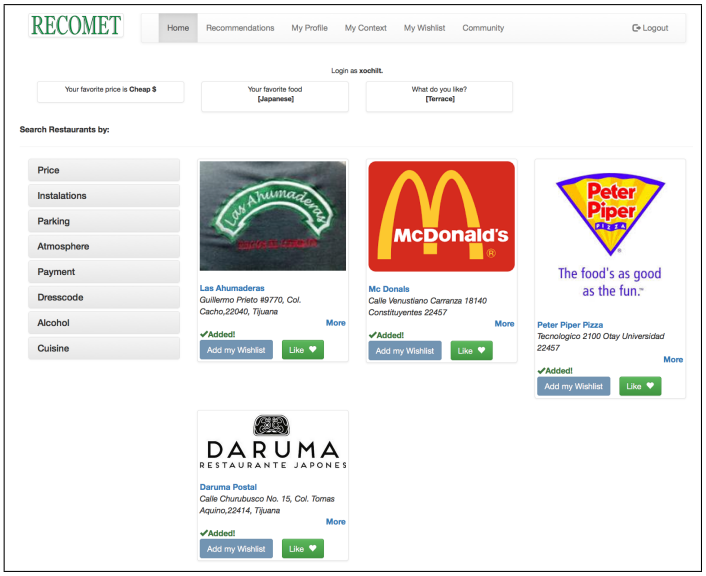


Figure 5.17: Functionality of wishlist button in Home page.

allows the user to have the own *personalized store*, in terms of products sales, the wishlist should be the future products to buy.

In a similar way, this prototype tries to infer the future tastes of the user or what restaurants could visit the user in a close future.

To facilitates the addition of restaurants the *Add my wishlist* button was added in every restaurant of the home page (Figure **??**) and in restaurants profiles (Figure **??**).

Community interface

The Community interface contains the reviews of all the users (Figure **??**) in order to display information that may helps another users to select any restaurant to visit.

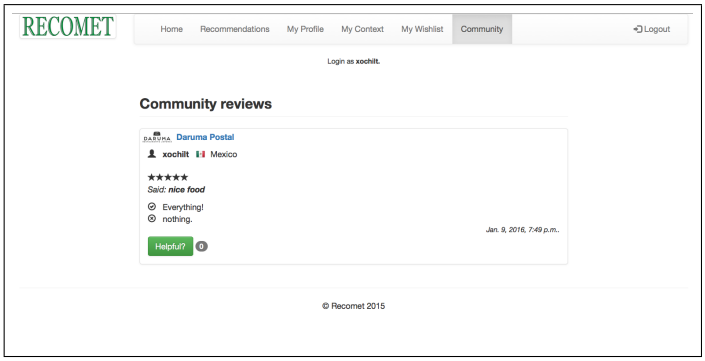


Figure 5.18: Community interface.

In the real life, the persons recommend some or another restaurant in the common language, remarking the why the restaurant is good or bad according their perception.

The Community section has the same goal, in fact, some reviews could be useful for the other users, this opinion may express it through the *Helpful button*. These are the functionalities in the prototype, in order to validate the performance, an on-line experiment was realized, the metrics of usability applied were Time-taks and Taks-success (mentioned in chapter **??**). In the next chapter will be explained the tests and the results obtained.

Chapter 6

System evaluation

Initially most recommenders have been evaluated and ranked on their prediction power, i.e, their ability to accurately predict the user's choices. However, it is now widely agreed that accurate predictions are crucial but insufficient to deploy a good recommendation engine.

In many applications people use a recommendation system for more than an exact anticipation of their tastes. Users may also be interested in discovering new items, in rapidly exploring diverse items, in preserving their privacy, in the fast responses of the system, and many more properties of the interaction with the recommendation engine.

We must hence identify the set of properties that may influence the success of a recommender system in the context of a specific application. Then, we can evaluate

how the system performs on the relevant properties [**?**].

In this thesis it performs the **on-line experiments**, this is maybe the most trustworthy experiment because is when the system is used with **real users**, typically **unaware** of the experiment. In this type of experiment it is possible to collect only certain types of data but this experimental design is closest to reality.

6.1 Usability Metrics

With the purposes of measuring the user experience, usability was used as evaluation metric of the prototype, then, two test were proposed: 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 [**?**].

The **task-success** is something that almost anyone can do. If the users can't 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 tests consist of a list of thirteen simple tasks that users shall perform in the system prototype. Before to start, a brief description about the system func-

tionalities and instructions to perform the test 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 "my favorite restaurants".*
7. *Add a review for 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.*

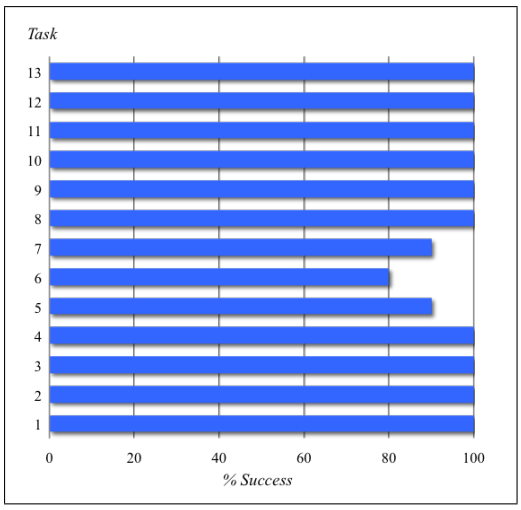


Figure 6.1: Chart of the percent of success for each task.

6.2 Enviroment set up

Each user makes the list tasks, the average time that each user used to finished the list was around ten minutes. The data was collected in every session and concentred in a chart in order to observe the users behaviour for each task.

In the Figure **??** the axis *(x, y)* represent the *task number* and *percent of success*, respectively. The chart shows that only three tasks weren't accomplished successfully, the task five, six and seven.

The issue in task five was that users can not found easily the reviews section in the screen because the reviews are in the restaurant's profile and not in the *Home page*. The issue in task seven is related of task five because the user couldn't find the manner to add a review, the user needs to click the restaurant profile to add

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

the review. The task six correspond to the favorites restaurants section that it is in the main screen, but the issue is that the user was confused to choose "wishlist container" instead of "favorites restaurants", both were showed in the *Home page*. Overall, these results mean a redisign in the prototype system to facilitate the performance of the tasks and create a more friendly interface.

By other side, the time that 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 [**?**].

The next test *task-on-time* is applied to measure time that an user uses to make each task. The time of task that each user used for each task is in Table **??**.

Time depends of the user capabilities and the complexity of the task. Some users perform and understand the task easily(as the user nine) but others take more time to perform the task(as the user three).

The "Null" value in Table **??** means that the user don't perform the task. After an analysis the user's feedback, it concludes that the problem is that task is not explained correctly, then, it needs to be more specific.

6.3 Results

To measure the efficiency of the metric, it choose a confidence interval. In this way, it is observed the time variability within the same task and also helps visualize the difference among the tasks to determine whether exists a statistically significant difference between these.

Figure **??**, shows the confidence interval for each task. The median was used to calculate the lower bound and upper bound of the confidence interval.

In order to have more precision in the confidence interval, it used the median instead the mean, the median corresponds the red numbers of the chart. Figure **??** also shows that task two and three show large confidence interval because of the delay of users to accomplish these task.

After tests, the USE *(Usefulnes*, *Satisfaction*, and *Ease of Use)* questionnaire [**?**]

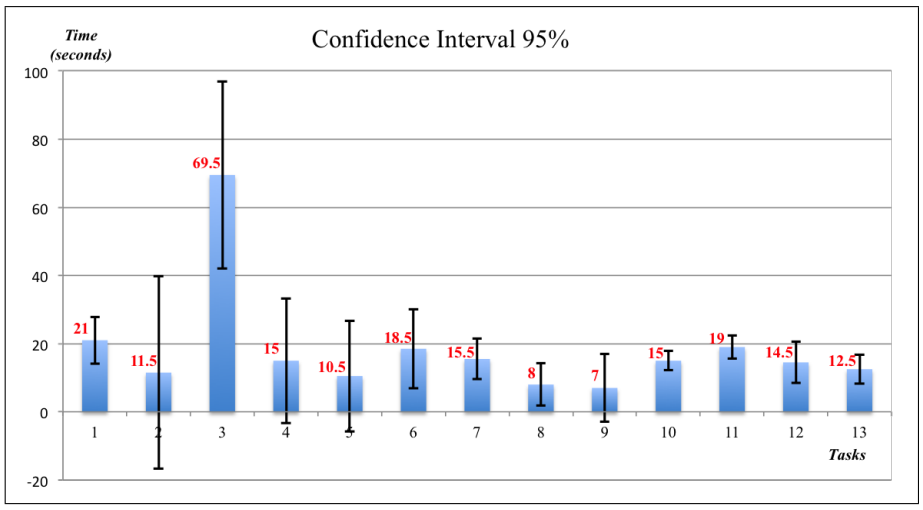


Figure 6.2: Confidence interval per task with a confidence level of 95%.

was applied in order to get the user's feedback and comments to know about the difficults in the test. Finally, it applied a questionnaire that measure the satisfaction level of users, this questionnaire serves to evaluate the experience that they had in the system interaction.

The *USE questionnaire* (appendix **??**) consists of 30 rating scales divided into four categories: *Usefulness*, *Satisfaction*, *Ease of Use*, and *Ease of Learning*. Each is a positive statement to which the user rates level of agreement on a Likert scale.

The USE questionnaire allows to get values for *Usefulness*, *Satisfaction*, *Ease of Use*, and *Ease of Learning*. The visualizing of the results is in the Figure **??**, where the four axis of the chart represent the values of percent which users rated positively this factors with respect to their experience with the system prototype.

The values are *Usability 83%*, *Satisfaction 84%*, *Easy of use 92%*, and *Easy of*

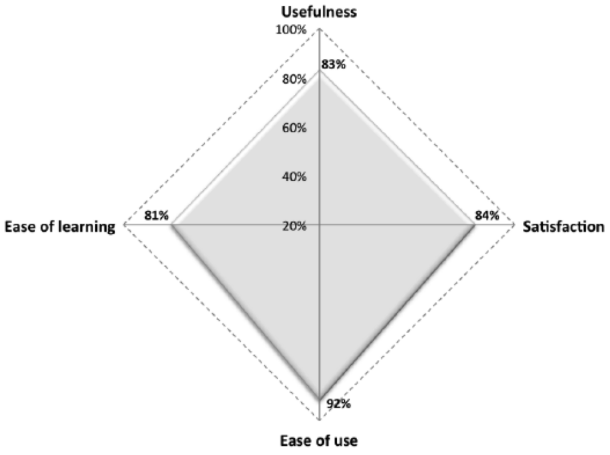


Figure 6.3: The radar chart that depicts the four axis evaluated.

Learning 81%, it means that the test was success, subsequently, as future work a second test will be make to compare the improves.

Chapter 7

Conclusions and future work

In the past, computer software was designed with little regard for the user, so the user had to somehow adapt to the system. This approach to system design is not at all appropriate today, the system must adapt to the user.

This is why design principles [**?**] are so important. The principles represent high-level concepts applied to the systems for the better design of user interfaces. We took this principles as a guideline to design the first prototype, however when the system is in a pilot test, we hope find the common problems that the users face normally. So, it was observed the user behaviour in the test, in order to identify the most frecuently difficults and doubts about proposed tasks.

Later, it was done a brief interview with the users in order to understand their feelings or mood, their experience, and overall, their opinion about the system pro-

totype. The conclusions are based in user's comments, then the common 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 represent the null values in data Table (see Table **??**), some users don't perform 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 to obtain the confidence interval (see Figure **??**).

By other hand, the USE questionnaire was useful to identify the weaknesses in the system prototype. The percent in the factors is in acceptable level (80%), it allows to say that the system has a good performance in the first test, but an increment in the number of users to perform the next possible test could get more real information.

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. 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) in the system prototype 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 contributes to improve the recommendations.

The proposed method was used in the system prototype in order to validate its performance, the case of study shows acceptable results but we consider that the

method could be efficient in others domains.

The challenge is to apply the method in a e-learning environment because the context involves more precise factors (for instance the level of noise, the level of light, the level of knowledge, the location of the users, etc.) and the recommendation process considers more conditions (or fuzzy rules) to make a recommendations, as well as the information of the user profile that contains more characteristics of the user preferences (as goals, level, learning style, activities, homework, score, average, etc.). The base of this future work is the Protoboard system [**?**] that is a e-learning platform for students of Tijuana Institute of Technology.

Publications

1. *Restaurant Recommendations based on a Domain Model and Fuzzy Rules. Xochilt Ramírez-García, Mario García-Valdéz. Recent Advances on Hybrid Approaches for Designing Intelligent Systems. Springer International Publishing Switzerland. (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).

Appendix A

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`

- System prototype programmed in Python language Url:`https://github.com/xochilt/recomet`.

Tijuana Restaurants dataset

Table **??** 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

Table A.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

in the url: `http://www.xramirezg.wixsite.com/ittresearch`.

Appendix B

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
```

Download Python code: `https://github.com/xochilt/recomet`

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 C

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). Users rate agreement with these statements on a 5-point Likert scale, ranging from strongly disagree to strongly agree. Statements in italics were found to weight less heavily than the others.

The on-line questionnaire is in the next link: `http://goo.gl/forms/LXQAerSKi4r0mTa33`.