Acknowledgments

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

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

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

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

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

Resumen

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

Abstract

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

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Recommender systems are a technique to 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 in a kind of item[**?**].

Recommender systems development initiated from a rather simple observation: individuals often rely on recommendations provided by others in making routine and daily decisions [**?**], [**?**].

Recommender systems try to imitate this human behaviour, to recommend a product in daily life is normally the most effective manner to achieve in this systems that the user prefers an item in a Web site.

Over the time, the improvements of recommender systems are focused on differents techniques that involve the context in the recommendation process.

The importance of contextual information has been recognized in different domains and disciplines. In recommender systems, context implies the situations around of the user when it is interacting with the system and all the information that this situation represents. In this work we use a definition proposed by Annin K. 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 important context in a specific recommender system.

However, some authors make an understandable comparison of different contexts such as Bazire et.al. [**?**] that compares different definitions of context in different fields and conclude that is complicated makes a unifying definition of context because of the nature of the concept in the disciplines. In computer sciences Fischer G. [**?**] defines context as the interaction between humans and computers in socio-technical systems that takes place in a certain context referring to the physical and social situation in which computational devices and environments are embedded. Also identifies the important aspects to consider when the context is used in that

framework: how it is the contextual data obtained , how the context is represented and what objectives and purposes the context has when used in a particular application.

Context-aware recommender systems are gaining even more attention because of their performance and adaptation for different domains, the way to improve personalized recommendations based in contextual factors is an important technique to increase the benefits in many domains. Nowadays many companies are incorporating some kind of context in their recommendation engines, application can be found in fields such as e-commerce [**?**], [**?**]; music [**?**],[**?**]; places of interest [**?**], movies[**?**]; vacation packages [**?**], [**?**]; travel guides [**?**]; e-learning[**?**] and restaurants[**?**].

The majority of these recommender systems are focused on recommending relevant items for users without additional contextual information such as time, location or companionship.

However, recent systems incorporate contextual information to make recommendations that deliver items adjusted to the users' current context. For instance, the site Sourcetone interactive radio [**?**] when selecting a song for a customer it takes into consideration the listener's current mood in order to provide a better recom-

mendation.

Contextual information is used to increase important characteristics such as user satisfaction, recommendation quality, usefulness and more.

In this thesis an architecture and methodology are proposed for designing context-aware recommender systems validated with a case study of a restaurant recommender system. The objetive is to demonstrate how the level of user satisfaction is increased through the use contextual information. The architecture is an hybrid of three techniques:

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

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

3. *Collaborative filtering technique* is based on the user profile to identify userś

preferences and to find neighbors that have the same taste. 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 the list of recommendations for the user and then in the next module are adjusted for user in context. This is the last step and is represented as *context filter* in the method, then the recommender system obtains a list of contextualized recommendations. The proposed methodology works simultaneously to obtain recommendations, the hybrid method allows the recommender system to generate suggestions despite having scarse user information. When the system faces the the cold-start problem or the over-specialization problem, the hybrid method responses properly to show results for the user. These problems are described in a later sction. The expertś recommendation contributes to this goal, the rules specified in the Fuzzy Inference System help to infer the most suitable restaurant considering the only the input variables and the knowledge represented by the rules.

The context-based filter then makes recommendations based on those items that are appropiate to the context of the user to produce the final recommendation list. The context-aware recommender system tries to increase the level of user satisfaction. To meet this goal two metrics were utilized to test the system in an on-line test.

Ten users were selected to interact with the system, subsequently an analysis about the system performance and main issues highlited by the users was realized.

The rest of this thesis is organized as follows: chapter 2 explains the state of the art in context aware recommender systems, chapter 3 contains the background of the related topics, chapter 4 describes the proposed context-aware recommender methodology describing each one of its components, then in chapter 5 the experimental set up and the results are presented, chapter 6 describes the evaluation of a case study a restaurant domain and finally, chapter 7 presents conclusions and a proposal for future work .

Chapter 2

State of the art

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

tagged of the user and the frecuency for each used tag. J.Yao et.al.[**?**] 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 movie recommendations. Trust takes on the role of a recommender system forming the core of an algorithm to predict a rating for recommendations of movies. This is an example of how the Semantic Web, and Semantic trust networks in particular, can be exploited to refine the user experience.

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

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

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

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

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

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

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

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

rent and past locations in the recommendation process. 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 presents 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 "I'm feeling LoCo" system [**?**] that proposes a ubiquitous location based recommendation algorithm that focuses on user experience by considering user preferences, time, location and similarity measures automatically, having Foursquare as a dataset. We also focus on user experience and aim that user input is minimal. The information om the user's social network, form of trans-

portation and phone's sensors is inferred to provide recommendation of places om the dataset. Reddy et.al[**?**] presents LifeTrack system that incorporates sensor information into song selection. The songs are represented in terms of tags that the user assigns in order to link the songs to the appropriate contexts in which they should be played. User feedback is incorporated to make a song more or less likely to play in a given context. Context considered relevant to song selection includes location, time of operation, velocity of the user, weather, traffic and sound. User locations and velocity are determined by GPS. Location information includes tags based on zip code and whether the user is inside or outside (inferred by the presence or absence of a GPS signal). The times of the day are divided out into configurable parts of the day (morning, evening, etc). The velocity is abstracted into one of four states: static, walking, running and driving. Use of accelerometers are planned to enable indoor velocity information. If the user is driving, an RSS feed on traffic information is used to typify the state as calm, moderate or chaotic. If the user is not driving, a microphone reading is used for the same purpose. Additionally, an RSS feed provides a meteorological condition (frigid, cold, temperate, warm or hot). For a better comprehension of the context, table 2.1 describes examples of contextual factors in different domains of application, specifies the contextual factors considered such as part of the context, the methodology for each application and kind of devices.

Table 2.1: Comparison of context-aware recommender systems.

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

Chapter 3

Background

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

3.1 Fuzzy Logic

Fuzzy logic originates on 1965 from the publication Fuzzy Sets[**?**] written by Lofti A. Zadeh from the University of Berkeley California for the journal Information and Control, based on the work of J. Lukasiewicz[**?**] about multi-valuated logic. In contrast with traditional logic, that uses absolute concepts when it refers to the real world, fuzzy logic defines these concepts with variable degrees of belonging, follow-

ing the reasoning patterns of human thought.

A distictive property of computational systems based on fuzzy logic theory is that, unlike those based on classical logic, they have the ability to acceptably reproduce a more natural form of reasoning, that considers the certainty of a proposition as a matter of degree. More formally we can say that if the logic is the science of the formal principles and policy reasoning, fuzzy or fuzzy logic refers to the formal principles of approximate reasoning, considering precise reasoning (classical logic) as a restrictive case. Thus, the most convenient features of fuzzy logic are its flexibility, tolerance for imprecision, its ability to model nonlinear problems and its roots in natural language. Although fuzzy logic is known by this name since the seminal work of Zadeh in 1965, the idea that lies behind it, and its origins date back to 2500 years ago. The Greek philosophers, including Aristoteles, believed that there were certain degrees of truthfulness and falsehood and even Platon worked with membership degrees.

Fuzzy logic has acquired a great reputation for the variety of its applications, which range from the control of complex industrial processes, device design to controlers of household electronics and entertainment devices, as well as diagnostic systems. Fuzzy logic is essentially a **multi-valued logic**[**?**], which is an extension of classical logic. The latter imposes on their statements only true and false values, however,

much of human reasoning is not as 'deterministic. The word fuzzy can be defined as: vague or confusing. Fuzzy systems are **knowledge-based systems** or **rule-based systems**. The kernel (core) of the fuzzy system consists of knowledge bases called **IF-THEN** fuzzy rules. Each fuzzy system is associated with a set of rules with a variety of representations, as in Figure 3.1 i, which can be obtained from numerical data or experts familiar with the problem at hand. Based on this information, actions are combined with background formatting rules/consequent, and then are added according to the theory of approximate reasoning to produce a nonlinear mapping of the input space $U = U_1 x U_2 x U_n$ the output space V, where $F_k^l U_k, k = 1, 2, n$, membership functions are type-1 antecedent, and $G^1 V$ is the membership function of type-1 consistent. The linguistic variables[**?**] are denoted by input $uk, = 1, 2, n$, and output linguistic variables are denoted by: $R^l : if x_1$ is F_1^l and x_2 is F_2^l and ... and x_n is F_n^l then V is G^1.

A fuzzy system consists of four basic elements shown in Figure 3.1: fuzzifier type-1, the fuzzy rule base, inference machine and defuzzifier type-1. The **fuzzy rule** is a collection of rules, which are combined in the inference machine to produce a fuzzy output. The fuzzifier maps numeric entries to fuzzy sets, which are then used as inputs to the inference engine, where the defuzzifier maps the fuzzy sets produced by the inference engine to traditional numbers.

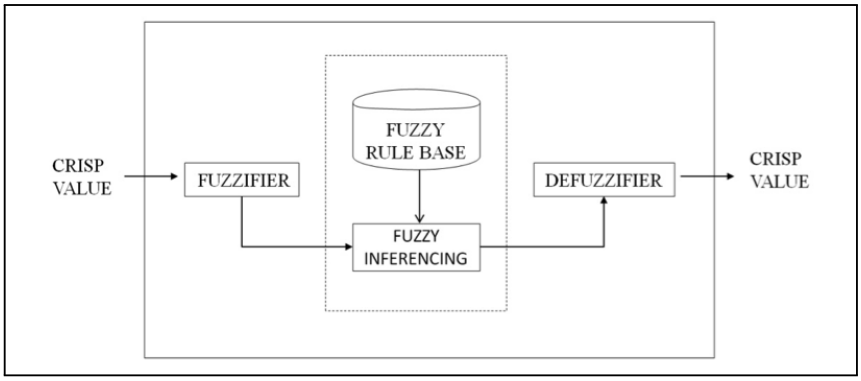


Figure 3.1: Estructure of a fuzzy logic system.

3.2 Context

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

1. **What a recommender system may know about these contextual factors.** A recommender system can have different types of knowledge, which may include the exact list of all the relevant factors, their structure, and their

values, about the contextual factors. Depending on what exactly the system knows (that is, what is being observed), it can classify the knowledge of a recommender system about the contextual factors into three categories:

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

- `Partially observable: Only some of the information about the contextual factors described above, is explicitly known. For example, the recommender system may know all the contextual factors, such as Time, PurchasingPurpose, and ShoppingCompanion, but not`

their structure. Note that there can possibly be different levels of partial observability. In this article we do not differentiate between them and group various cases of partially observable knowledge into this general category.

- Unobservable: No information about contextual factors is explicitly available to the recommender system, and it makes recommendations by utilizing only the latent knowledge of context in an implicit manner. For example, the recommender system may build a latent predictive model, such as hierarchical linear or hidden Markov models, to estimate unknown ratings, where unobservable context is modeled using latent variables.

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

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

   - **Dynamic**: This is the case when the contextual factors change in some

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

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

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

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

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

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

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

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

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

This definition is closer to the real about behaviour of **context-aware recom-**

mender system when it incorporates contextual information to get recommendations, in addition, is fewer confused and specific than other author's definitions.

3.3 Recommender systems

3.3.1 Collaborative Filtering algorithm

The idea of collaborative recommendation approaches is to exploit information about the past behavior or the opinions of an existing user community for predicting which items the current user of the system will most probably like or be interested in [**?**]. Recommender systems are useful in several types of applications, however, the big impact is mainly in web sites for sales in order to personalize the information for a particular user, then the system helps to promote another items to grow up the sales in the on-line storage. Collaborative filtering take a matrix of given *user-item* ratings as the input to generate a prediction for each item indicating to what degree the current user will like or dislike an item, subsequently the list of n recommended items for the user that contains items no reviewed by the user. Differents approaches are utilized for collaborative filtering such as: a) user-based nearest neighbor recommendation, b) Item-based nearest neighbor recommendation and c) model-based recommendation.

a) *User-based nearest neighbor* is the most used approach because of is easy of

implementation and efficient results. Only need the rating matrix to obtain recommendations for users. The neighborhood selection is taking the K nearest neighbors into account usind the threshold to define the size of the neighborhood. A reduced size of neigborhood can not make predictions, on the contrary, is the neighborhood is large the information for recommendations can not be significant.

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

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

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

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

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

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

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

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

to calculate predictions taking in account the latent factors.

3.3.2 Content-based algorithm

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

Content representation. The simplest way to describe catalog items is to maintain an explicit list of features for each item (also often called attributes, characteristics, or item profiles). For a book recommender, one could, for instance, use the genre, the authors name, the publisher, or anything else that describes the item and store his information in a relational database system. When the users prefer-

ences are described in terms of his or her interests using exactly this set of features, the recommendation task consists of matching item characteristics and user preferences.

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

3.3.3 Hybrid recommender systems

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

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

that combine several algorithms or recommendation components.

3.3.4 Context-aware recommender systems

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

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

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

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

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

3.3.5 Paradigms

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

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

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

is adjusted (contextualized) for each user using the contextual information.

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

Chapter 4

Proposed method

4.1 Data models

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

4.1.1 Restaurant model

An effective on-line recommender system must be based upon an understanding of consumer preferences and successfully mapping potential products into the consumers preferences[**?**]. 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 experience when they are consuming the product or visiting the place.

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

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

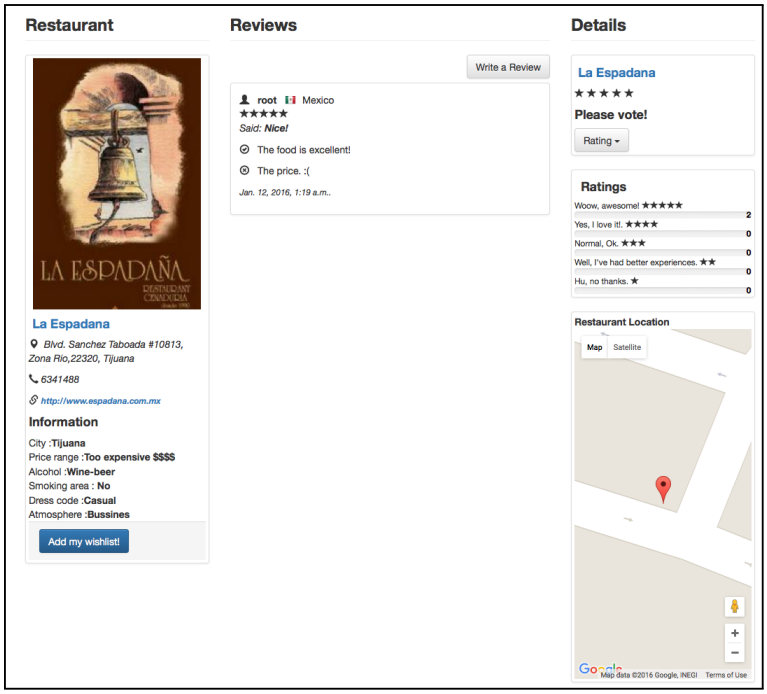


Figure 4.1: User interface of the restaurant model.

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

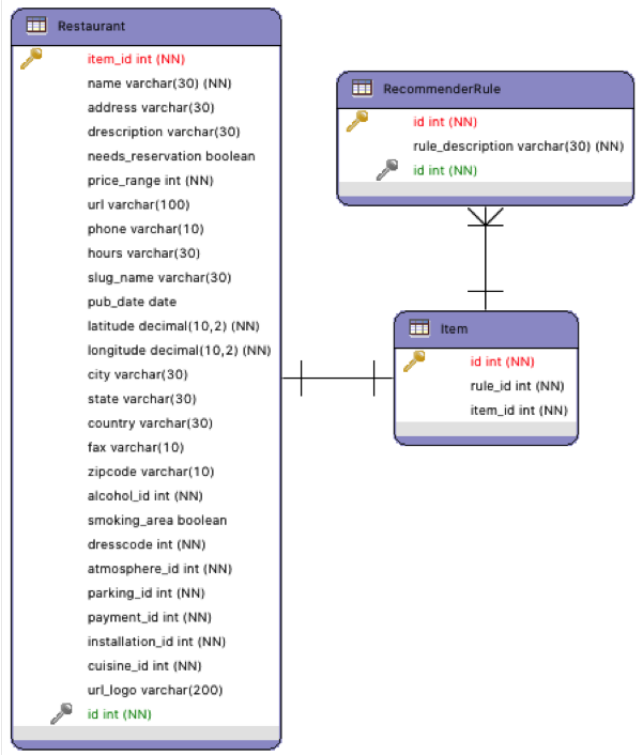


Figure 4.2: The data model of restaurant.

Data model

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

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

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

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

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

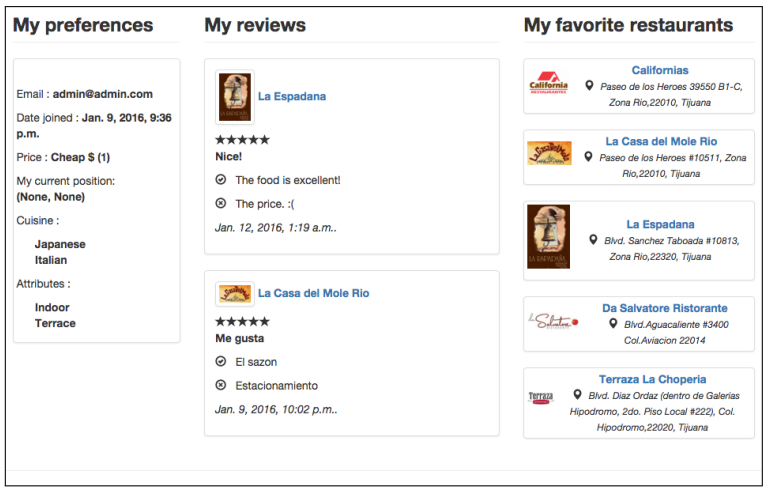


Figure 4.3: Example of user interface for user profile.

4.1.2 User model

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

In context-aware recommender system, user profile has contextual information such as: 1) price range, 2) current location, 3) cuisine types, 4) attributes or features of

restaurants that the user want, 5) the reviews posted, and 6) the favorite restaurants list. The user profile is stored in database and it is available for queries request, and it can be changed by users many times in a session. The information used to recommendations is the last one register stored. The user interface is represented in figure 4.3.

Data model

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

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

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

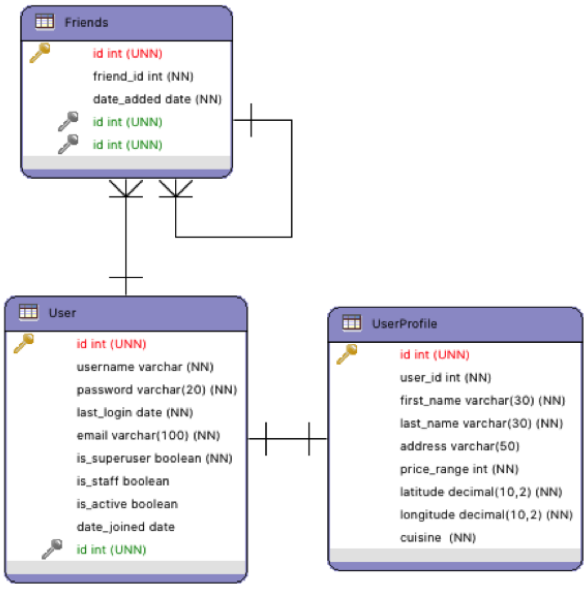


Figure 4.4: The data model of user profile.

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

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

4.1.3 Relational data model

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

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

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

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

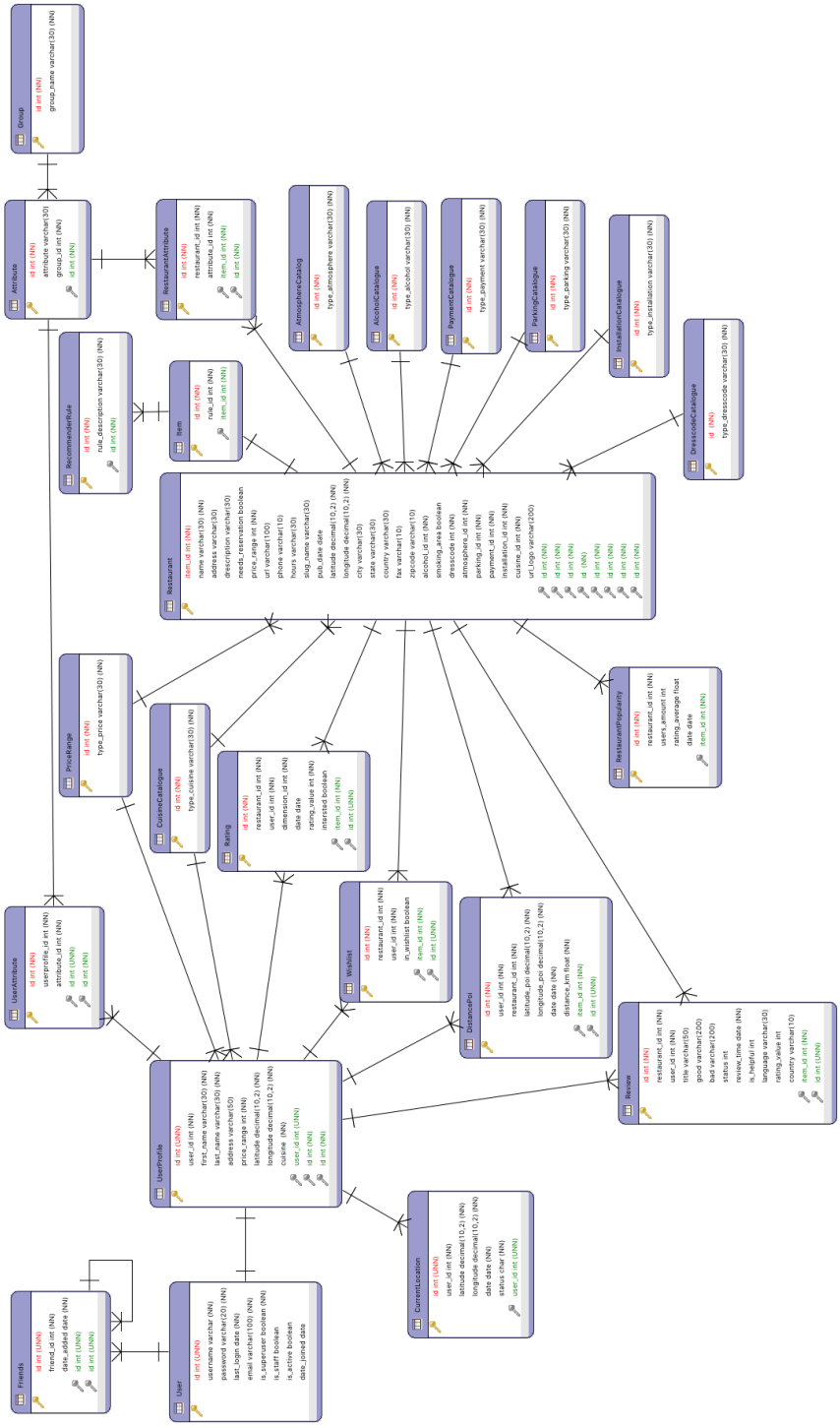


Figure 4.5: The relational database of context-aware recommender system.

4.2 Expert recomendation

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

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

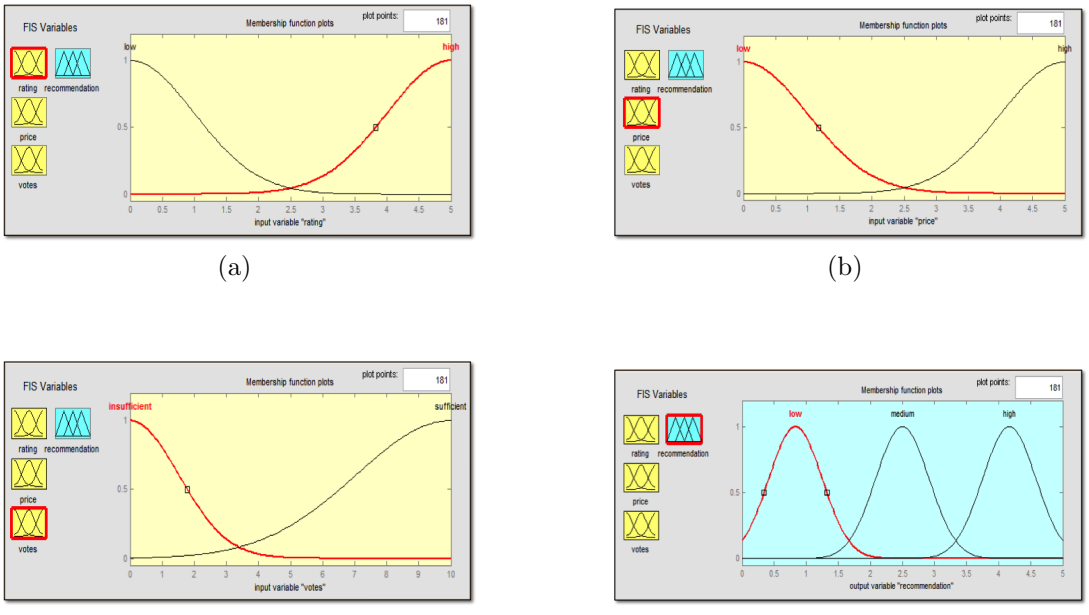
In this work, fuzzy logic is used to model fuzzy variables that highligh in the popularity of a restaurant. The context-aware recommender system has implemented a Fuzzy Inference System that represents the expert recommendation.

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

The Fuzzy Inference System proposed has 3 **input variables** (such as in previous work realized[**?**]): 1)*rating* is an average of ratings that has a particular restaurant inside the user community; the domain of variable is 0 to 5 and contains 2 membership functions labeled as *low* and *high*(figure 4.9a), 2)*price* represents the kind of price that has a particular restaurant; the domain of variable is 0 to 5 and contains 2 membership functions labeled as *low* and *high* (figure 4.9b), and 3)*votes* is used to

measure how many items have been rated by the current user, i.e., the participation of the user, if the user has rated few items (less than 10) is not sufficient to make accurate predictions(figure 4.9c), the domain of variable is 0 to 10 and contains 2 membership functions labeled as *insufficient* and *sufficient*.

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



(a) (b)

(c) (d)

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

The proposed Fuzzy Inference System(figure 4.7) represents the users experience

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

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

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

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

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

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

4.3 Fuzzy Inference System to assing weights

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

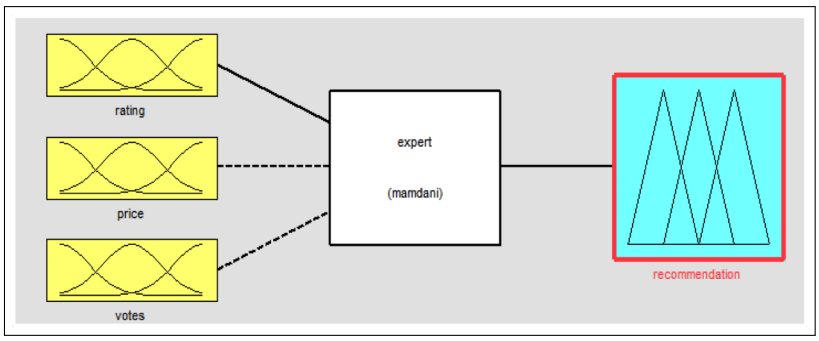


Figure 4.7: Fuzzy Inference System of expert.

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

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

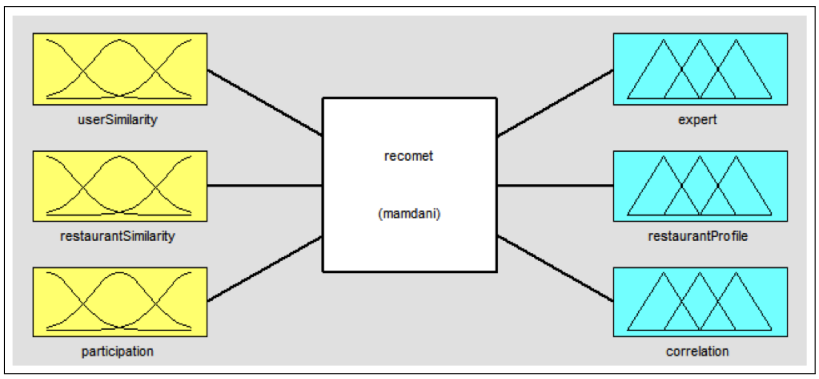
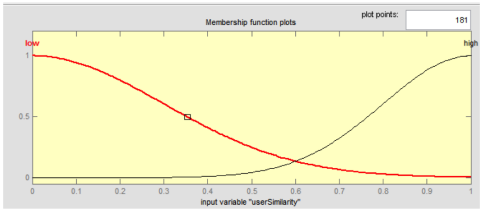
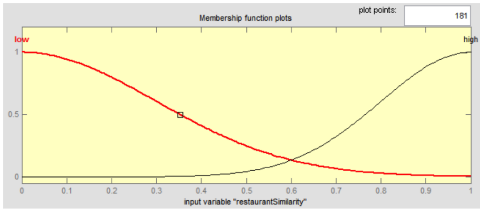


Figure 4.8: Fuzzy Inference System to assign weights.

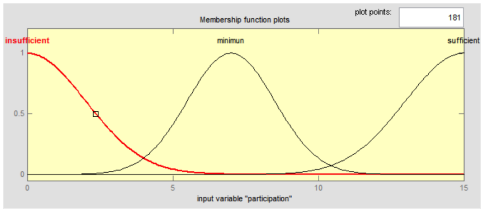
membership functions are in a range from 0 to 1 to get the value. Taking in account



(a)



(b)



(c)

Figure 4.9: The Gaussian membership functions of input variables.

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

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

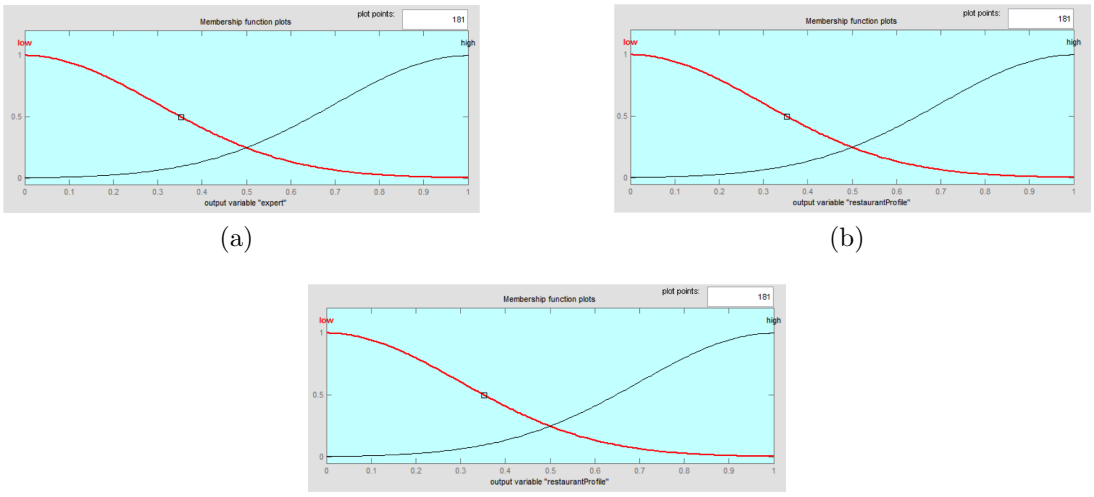


Figure 4.10: The Gaussian membership functions of output variables.

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

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

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

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

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

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

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

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

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

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

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

tion *is sufficient then* ***expert*** *is low,* ***restaurantProfile*** *is low,* ***correlation*** *is high.*

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

4.4 Contextual Recommendation

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

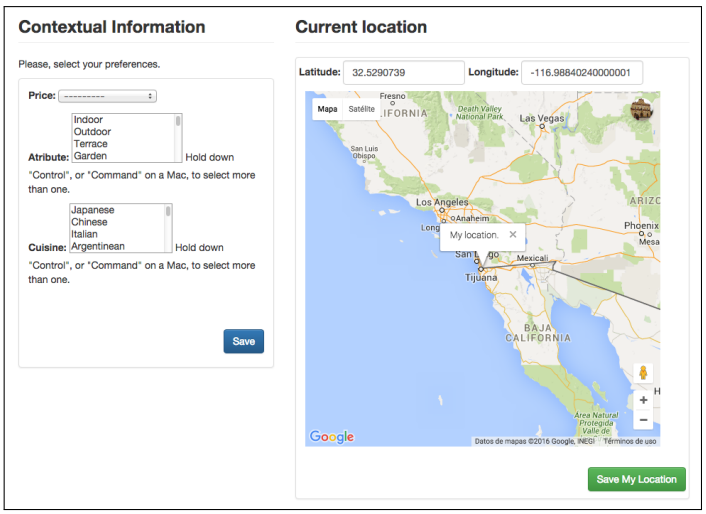


Figure 4.11: System interface to collect contextual information.

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

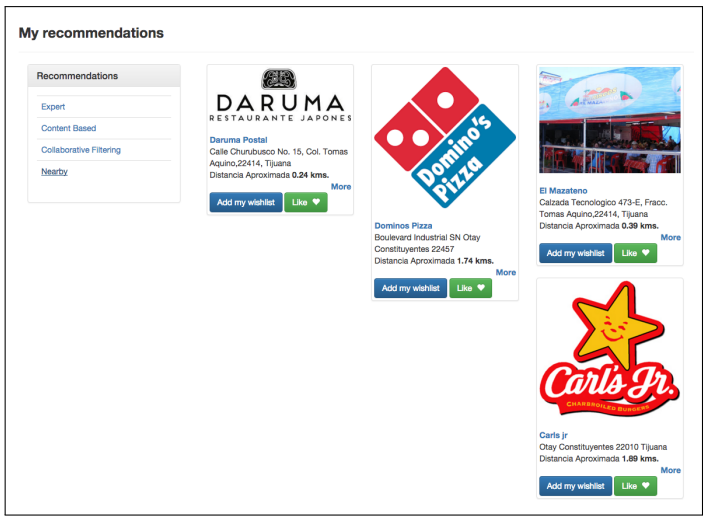


Figure 4.12: System inferface of recommendations for the user.

4.5 Architecture

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

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

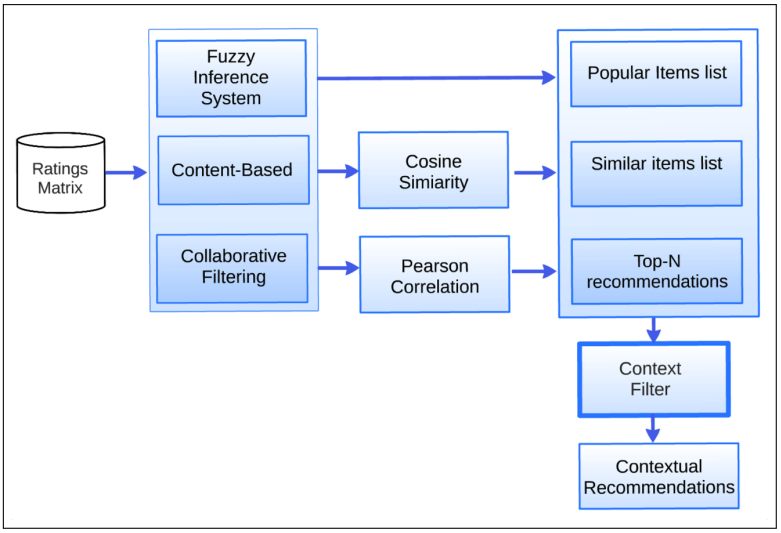


Figure 4.13: Context-aware recommender system architecture.

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

Chapter 5

Experiments and Results

In this chapter are explained the experiments performed in order to reach the goal of this thesis. Every experiment utilizes different datasets and context domains.

5.1 Tijuana Restaurants dataset

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

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

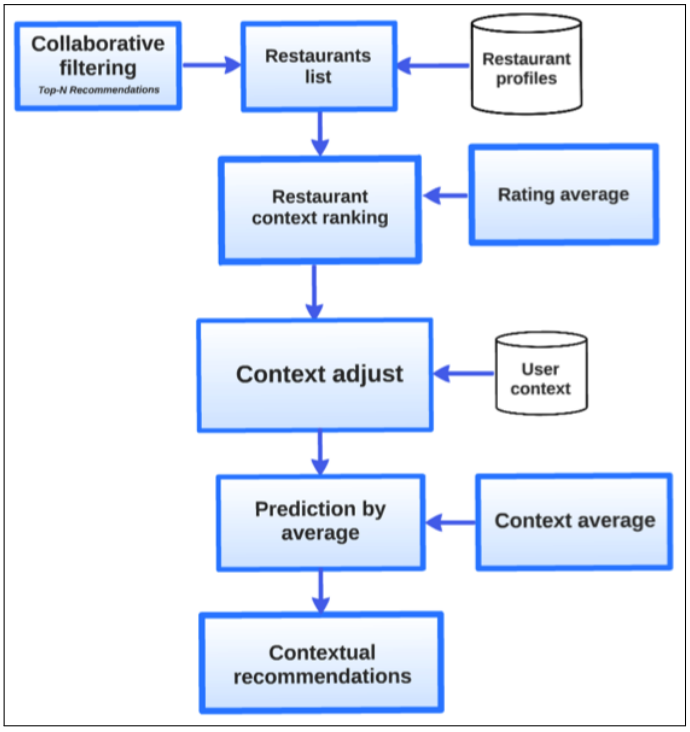


Figure 5.1: The Post-Filtering architecture for Tijuana restaurants.

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

Table 5.1: Questionnaire applied to collect contextual dataset.

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

In order to validate the proposed approach, data about restaurant preferences of users in different contexts was collected. The study subjects were students enrolled in a computer engineer major, a masters program and professors of the Tijuana Institute of Technology. A total of **50 users** answered a questionnaire; the questions were about their preferences for nearby restaurants and the technology used by them. The *questionnaire* consisted of **8 questions** and they rate restaurants from a list of 40 restaurants. Each restaurant chosen was rated 6 times one per context considered, a matrix rating with *1,422 ratings* were collected. The questions are shown in the table 5.1.

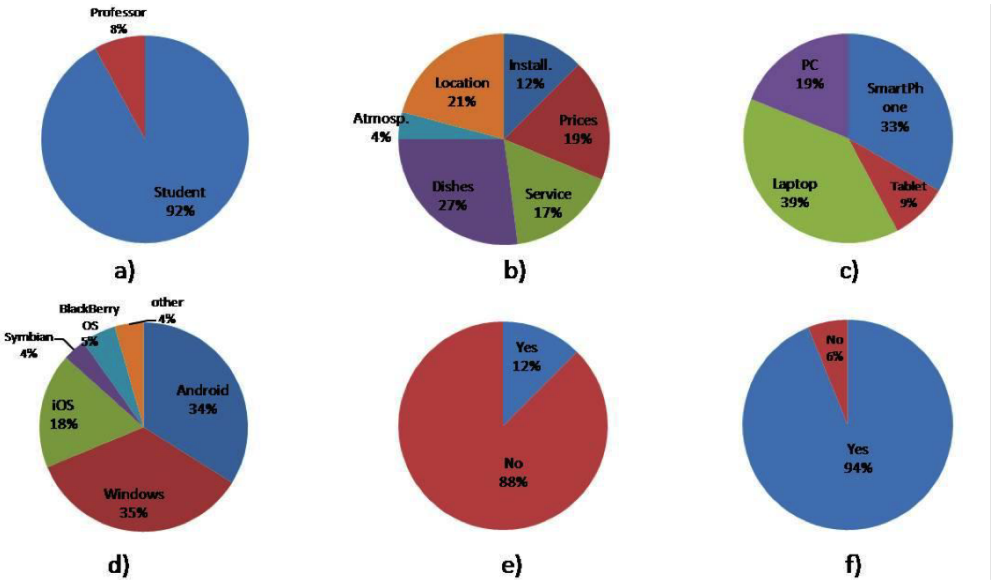


Figure 5.2: The chart shows the users preferences for questions from 1 to 6.

The user's answers from question 1 to question 6 are represented in the figure 5.2. **Figure 5.2a** represents the percentage of surveyed students and teachers; **figure 5.2b** the percentage of the element that users consider the most important to visit a restaurant; **figure 5.2c** represents the preferences of devices when are using Internet for restaurant recommendations; **figure 5.2d** represents the percentage of operating system that often used, **figure 5.2e** shows the percentage of users that use the Internet to search restaurants in Tijuana; and **figure 5.2f**, shows the percentage of users that would like using a restaurant recommender system of Tijuana. For questions 7 and 8 only the top-ten restaurants are shown, without/with the contextual situation. In figure 5.3a, the favorite restaurant is **Daruma**(178 votes),

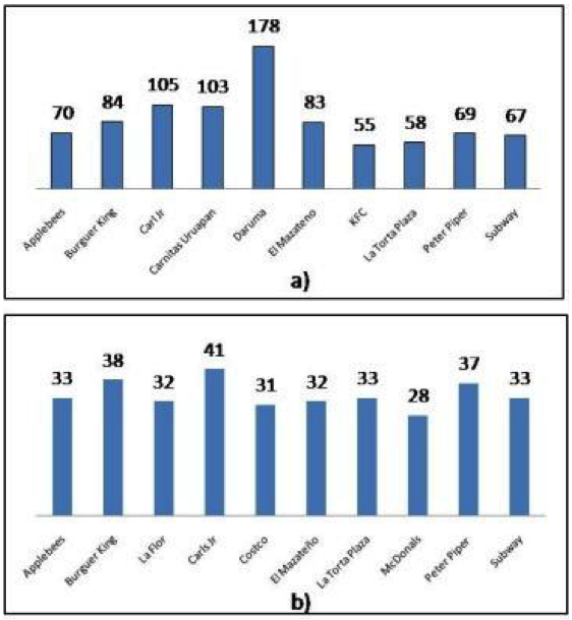


Figure 5.3: The chart shows the users preferences for questions 7 and 8.

whereas in figure 5.3b, **Daruma** does not appear in the top-ten. When considering the context *midweek*, the favorite restaurant was **Carls Jr.**, which appears in both graphs; this restaurant was also the most voted in the different contexts. Contextual recommendations of post-filtering approach depends of context *midweek* or *weekend*, which is the day when the restaurants were rated. Subsequently, the result of the query is refined according to the user context; the 6 contexts mentioned correspond to combinations of contextual factors shown in table 5.2. The dataset was explicitly collected from **50 users** whom answered questionnaire (see table 5.1). A total of 172 predictions was made for different users and the error **MAE=0.5859** when the context **midweek** for current user was considered. The observation for this result

Table 5.2: Contextual factors considered in the questionnaire.

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

is that using a small dataset the performance of the method proposed is limited. By other hand, having only one contextual factor does not improve the accuracy of the recommendations in this domain.

5.2 MovieLens dataset

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

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

- **MovieLens 1M Dataset**: Stable benchmark dataset, 1 million ratings from 6000 users on 4000 movies. Released 2/2003.

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

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

1/2009.

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

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

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

In [**?**] it is shown that the time division is beneficial and its performance depends on the items selection method and influence of contextual variables in item ratings.

In [**?**] the user profile is segmented into micro-profiles corresponding to a particular context, each context represents a time span in which recommendations for users are derived.

This experiment implements fuzzy logic on time segmentation, in order to improve user satisfaction by providing recommendations based to context and recents user preferences without discarding tastes in the past, as they include important infor-

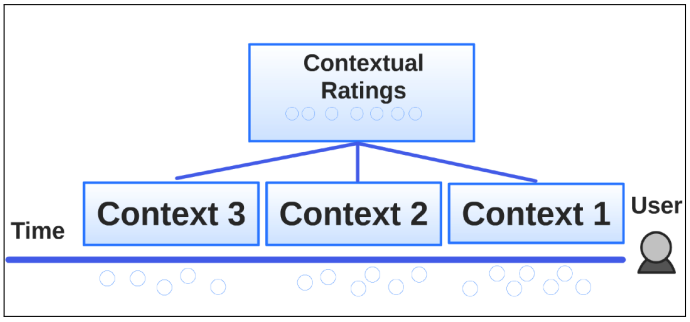


Figure 5.4: Time segmentation of contexts based on current user context.

mation for the recommender system proposed. The first phase is division of three time segments based on the current context of the user is performed such as in is depicted in figure 5.4.

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

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

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

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

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

Finally, the system gets the recommendations list for each user in different contexts. The recommendation process of pre-filtering is depicted in figure 5.5. The dataset used to test the algorithm was MovieLens(100000 ratings) with 943 users and 1,682 movies. The ratings were collected in a period of 2 years. MovieLens is not a contextual dataset, however, the timestamp was used to determine the rating time, i.e., in this way it was noted the day to know whether the rating time was in weekday or weekend. In this terms, the context was used. Then, the time for each context was

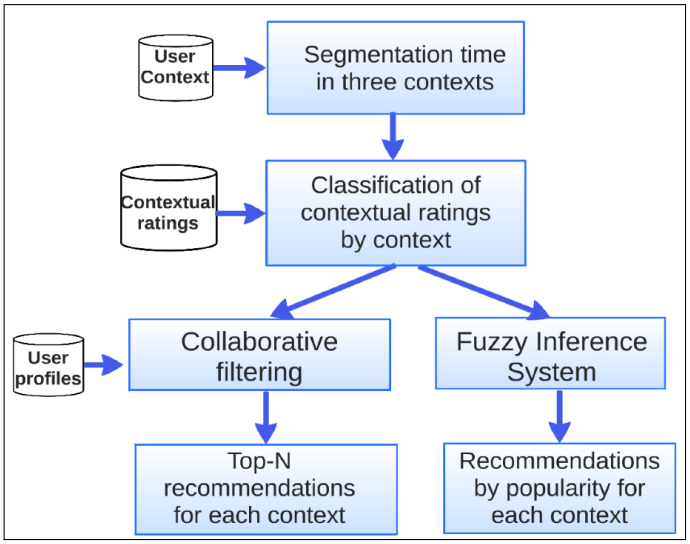


Figure 5.5: Pre-filtering process for context-aware recommender system.

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

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

divided in 3 months each one, this span covers 9 months before the user's current context.

The neighbors in each context are considered to recommend movies in that context only. An average of predictions are considered for add a movie to the top-N list of contextualized recommendations.The result in table 5.3 shows the error in three contexts. The error increase in context 3, in this context the ratings matrix is a little bit sparse; the error is justifiable because user has less participations.

5.3 Tripadvisor dataset

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

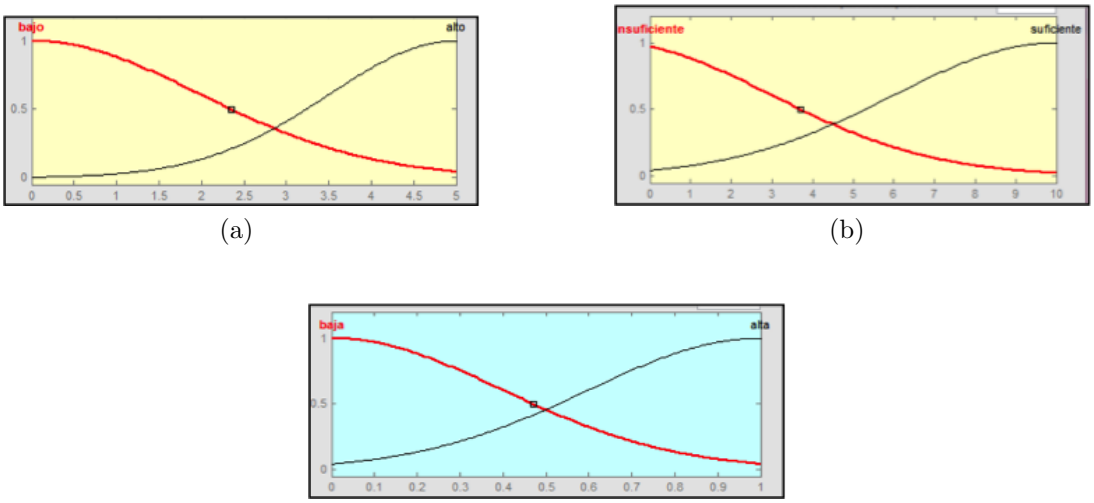
The first dataset contains 4669 contextual ratings, 1202 users and 1890 hotels; the second dataset contains 14175 contextual ratings, 2731 users and 2269 hotels. Data were collected of reviews online in tripadvisor.com. There is only one context: type of trip (family, friends, bussines, romantic and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

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

The context-aware recommender system uses the post-filtering paradigm[**?**] for adjust recommendations in context. The recommendation by popularity is through the Fuzzy Inference System depicted in figure 5.7, the Fuzzy Inference System contains the variables that are involved in the process to recommend in a human interaction, this process is the same that the recommender system does.

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

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



(a) (b)

(c)

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

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

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

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

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

4. *If* ***RatingAverage*** *is high and* ***UserParticipation*** *is sufficient then* ***rec-***

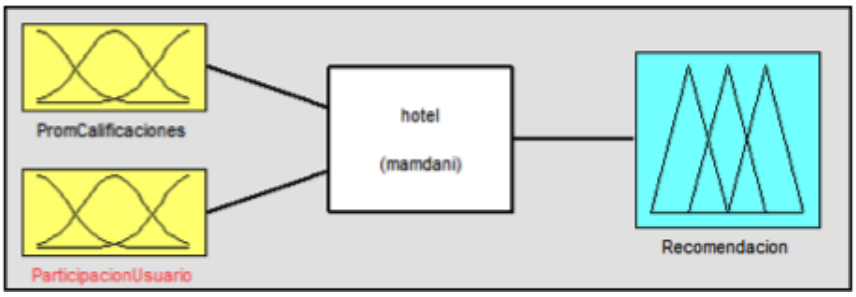


Figure 5.7: Fuzzy Inference System.

ommendation *is high.*

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

In other words, it makes a comparison of profiles of each item to determine the most similar to items the user has rated with highest score, context-aware recommender system proposed has a scale from 1 to 5. In the next step the outputs of every rec-

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

User profile		
Item1	Rating1	Context1
Item2	Rating2	Context2
Item3	Rating3	Context3

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

Context-aware recommender system identifies contextual data of the user profile (see

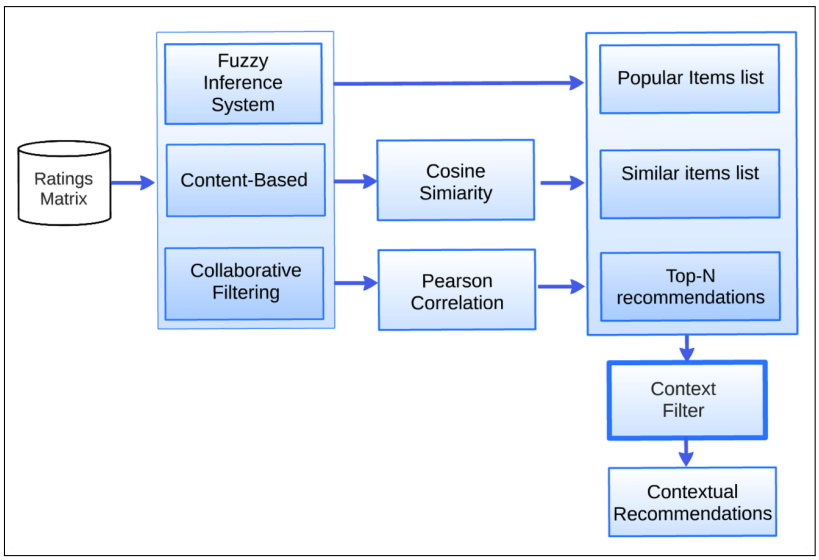


Figure 5.8: Recommender system architecture

table 5.4), and compares recommended items to filter those items that are adjusted to the user context.

The context filtering is the last step before to get the recommended items. The schema of architecture for context-aware recommender system is depicted in figure 5.8. Two experiments were performed using TripAdvisor dataset, table 5.5 describes the data sets and the scarcity percentage of the specified data. Scarcity of 99% mean that there are problems to recommend items because the information is not enought to get good recommendations.

By other side, in table 5.6 the comparison shows that the algorithm has a acceptable performance, i.e., the error falls into the range of results obtained with others algorithms. Then, contextual recommendations were evaluated with the Root Mean

Table 5.5: Datasets description.

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

Table 5.6: Comparison of RMSE.

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

Square Error in order to compare the results with context relaxation algorithm[**?**] that is evaluated with the same dataset.

The fundament of content-based is the cosine similarity; this means that if similarity value among items is high, the recommendations will improve the degree of user satisfaction. This is observed when calculating the similarity average in each dataset as shown in table 5.7.

FIS can provides a list of popular items for each dataset, recommendations through averages are obtained, and recommendations are conditioned to show it when the collaborative filtering and content- based are not delivering recommenda-

Table 5.7: Level of similarity among items in datasets.

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

tions because of data scarcity. However, the majority of popular items of dataset were rated in contexts: romantic, family and bussines, that means that the dataset has biases.

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

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

5.4 Datasets in matrix factorization

Filmtrust dataset

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

Table 5.8: Contexts in InCarMusic dataset.

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

InCarMusic dataset

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

5.4.1 Results

For experiments with matrix factorization technique the Graphlab toolbox was used. Both mentioned datasets and Movielens (1 millon and 10 millions) were used to test the algoritmh. The test involves K factors that are increasing for 50 iterations. previously, was done a test to identify what number of iterations are enough to get a good result with no overload of process in the algorithm. Results are depicted

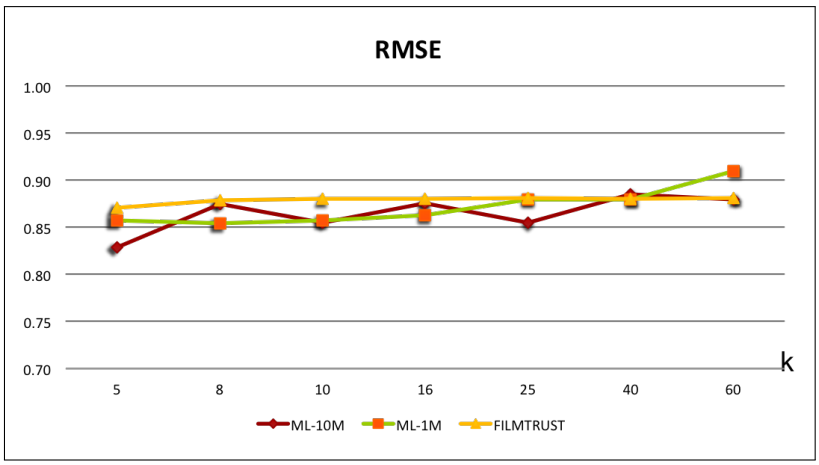


Figure 5.9: RMSE results of matrix factorization test.

in the chart 5.9 where the axis (x, y) represent the K value and the error value, respectively. The observations deal to small differences among the datasets, in a range of 0.80-0.90, and the high variability is in MovieLens dataset 10 millions. The big dataset implies more unstable behaviour, while in a small dataset (Filmtrust) the error is less variable. A comparison among MovieLens 1 million and 10 millions shows that there's not a significant difference.

By other side, other datasets were used to test matrix factorization under the same parameters to calculate the RMSE for each one. Table 5.9 presents the total of ratings of each dataset, the cosine similarity, it means how similar are the items into the dataset, and the RMSE error obtained in the test with matrix factorization technique. The datasets contain less ratings than the presented in the chart 5.9, according the table 5.9 is not possible to assum that matrix factorization has a better

Table 5.9: RMSE of datasets using matrix factorization.

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

performance with small datasets, because TripAdvisor and InCarMusic datasets obtain an error in the same range that the large datasets of the previuos chart.

Chapter 6

System evaluation

6.1 Metrics

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

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

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

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

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

6.2 Enviromental set up

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

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

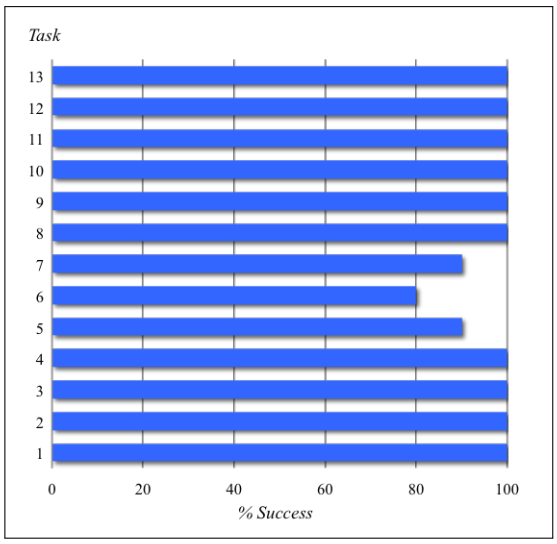


Figure 6.1: Representation of the percent of success for each task.

the task.

6.3 Results

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

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

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

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

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

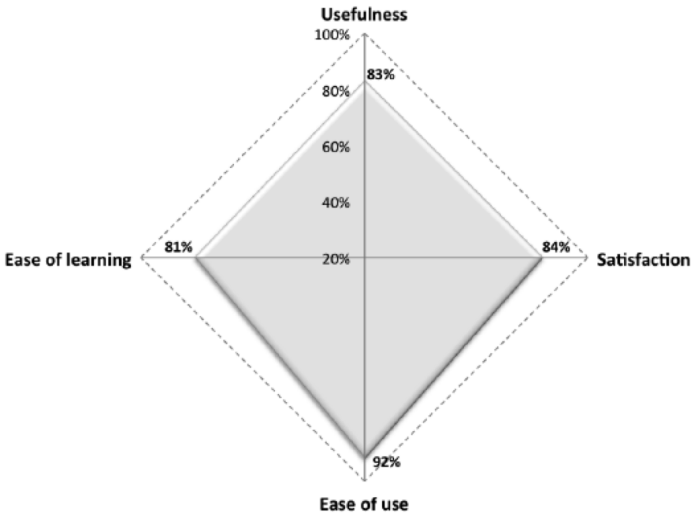


Figure 6.2: The radar chart that depicts the four axis evaluated in the questionnaire.

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

Chapter 7

Conclusions and future work

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Apply an statistical test to analize the results.

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

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

Publications

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

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

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

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

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

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

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

Appendix A

Pseudocode

Algorithm 1 Get Cosine similarity values

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

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

Algorithm 2 Collaborative filtering algorithm

Require: The userId.
Ensure: The Top-N list of recommendations for the current user.
  $ratingMatrix \leftarrow allRatings$
  Call $Recommendations \leftarrow getRecommendations()$ module
  return *Recommendations*

Algorithm 3 Content-Based Algorithm

Require: The user id.
Ensure: The Top-N list of recommendations.
  $RV \leftarrow$ All items that user rated with 5
  for *item* to size of RV **do**
    if *item* is not in RV **then**
      $UV \leftarrow itemid$
    end if
  end for
  $allItems \leftarrow [\,]$
  $getItemsProfilesUser \leftarrow$ Binary vectors of RV
  $allRatings \leftarrow$ Rating matrix
  for *item* to size of $allRatings$ **do**
    if *itemid* is not in $allItems$ **then**
      $allItems \leftarrow item$
    end if
  end for
  $getAllItemsProfiles \leftarrow$ Binary vectors of allItems
  $getCosineSim \leftarrow$ getItemsProfilesUser,getAllItemsProfiles
  for *item* to size of $highCosineSim$ **do**
    if $itemsimilarity \geq 0.8$ **then**
      $highCosineSim \leftarrow item$
    end if
  end for
  Sort $highCosineSim$ list
  return $itemProfiles$

Algorithm 4 Get item profiles

Require: The UV vector, allItems vector and boolean value of userProfile.
Ensure: The list of temProfiles in binary vectors.

```
if userProfile true then
    getItemsProfilesUser ← UV
    for itemp to size of UV do
        get binary vector of itemp
        itemProfiles ← itemp
    end for
else
    allItemProfiles ← allItems
    for itemp to size of allItems do
        get binary vector of itemp
        itemProfiles ← itemp
    end for
end if
return itemProfiles
```

Algorithm 5 Calculate Cosine similarity

Require: The itemProfileUser and itemProfileAll, both vectors in binary format.
Ensure: The cosine similarity value.

```
sum ← 0
normaItemUser ← 0
normaItemAll ← 0
for position to size of itemProfileUser do
    sumProduct ← sumProduct + (itemProfileUser[position] * itemProfileAll[position])
end for
for item to size of itemProfileUser do
    normaItemUser ← normaItemUser + itemProfileUser[item]^2
end for
for item to size of itemProfileAll do
    normaItemAll ← normaItemAll + itemProfileAll[item]^2
end for
squareRootUser ← squareroot(normaItemUser)
squareRootAll ← squareroot(normaItemAll)
cosineSimilarity ← sumProduct/(squareRootUser * squareRootAll)
return cosineSimilarity
```

Algorithm 6 Create a binary vector of item profile

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

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

Algorithm 7 Get recommendations

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

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

Algorithm 8 Get Pearson correlation

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

Algorithm 9 Matrix factorization

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

Ensure: The factorized matrix P and Q.

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

Appendix B

USE Questionnaire

Usefulness

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

Ease of Use

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

Ease of Learning

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

Satisfaction

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

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