Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: ***INFORMATION PROCESSING FOR MOOD-BASED CONTEXTUAL RECOMMENDATION IN MOBILITY ENVIRONMENTS***

MASTER DEGREE: ***MASTER IN SCIENCE IN TELECOMMUNICATION ENGINEERING & MANAGEMENT***

AUTHOR: ***Marc Planagumà i Valls***

DIRECTOR: ***Roc Meseguer***

SUPERVISOR: ***Alejandro Cadenas***

DATE: **March 25th, 2011**

TITLE: ***INFORMATION PROCESSING FOR MOOD-BASED CONTEXTUAL RECOMMENDATION IN MOBILITY ENVIRONMENTS***

MASTER DEGREE: ***MASTER IN SCIENCE IN TELECOMMUNICATION ENGINEERING & MANAGEMENT***

AUTHOR: ***Marc Planagumà i Valls***

DIRECTOR: ***Roc Meseguer***

SUPERVISOR: ***Alejandro Cadenas***

DATE: **March 25th, 2011**

Overview

Context-awareness is a technological trend that adds value in many fields. In this work, a research on context that is associated to public information contents is presented.

The global objective is to perform fully contextual recommendations in a variety of scenarios, where the context model extracted for information items is used as the basis for the enhanced recommendation.

Specific works provided on this thesis are the details about the mechanisms to extract context from user-generated contents. Such mechanisms are based on a combination of text mining algorithms, a flexible model and execution architecture that enables the processing of any kind of item for the purpose of context extraction.

Apart of the processing model details, a complete prototype is implemented and presented as a "*Telefonica I+D*" project called *Walkopedia*®, for a specific use case, especially interesting for telecommunications operators, as context-based recommendations in mobility environments, where getting information highly-personalized not only for the specific user but also to specific context is a must.

INDEX

IMAGES INDEX

TABLES INDEX

Chapter 1 Introduction

Project approach

This document defines a research and development made by me inside a "*Telefonica I+D"* project called *Walkopedia*®.

Walkopedia® is a recommendation service for events and venues in the city of London, depending on the context of the user. We can understand context as the location, weather and especially the intent or mood of the user.

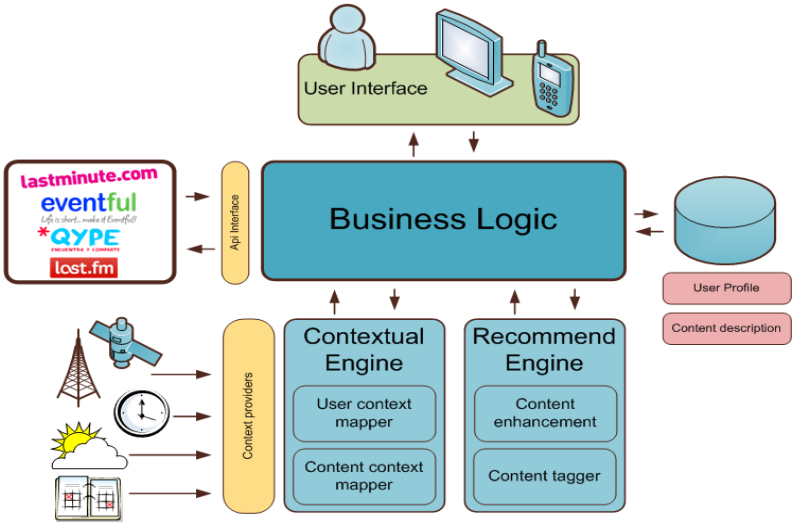

Figure 1 - Walkopedia® project architecture

As we can see in Figure 1 project has all necessary infrastructure to provide a commercial service, business logic, data storage, data input interfaces and end-users APIs. Besides the infrastructure the main functionality of the project is located in the data processing algorithms, content recommendation and contextualization of content.

My work in this project was the design a contextual recommendation solution and the implementation of necessary algorithms to obtain information retrieval, pattern detection and context inference over content. This work is the subject of this project and this document.

I also participated as a developer of *Walkopedia*® Infrastructure and the architecture defining but this work is not covered by this document. The explanation of *Walkopedia*® project is necessary to place the work presented on my master thesis, although *Walkopedia*® there isn't purpose of this document.

Background and motivation

The huge diversity of information available on the internet makes it necessary to look into mechanisms to filter or select that, especially in environments in which the user does not have too much time to dedicate to analyze the information and perform the filtering by him.

The mobility scenarios are a very clear example of the need to implement mechanisms to adapt the information available to the end user that is to consume that.

On this regard, emerging trends in context-aware have fostered a new breed of applications that rely on the intelligent analysis of user data in order to extract implicit or explicit information about the user's situation at a given moment, [1].

Contextual information, if processed in an appropriate manner, may become the key to deploy, select, manage or enrich different context-aware added value applications or services that will be located in different platforms, either hosted in the operator network, or located at elements that the subscriber may manage directly, including mobile and smart phones, [2]. The key feature of these applications is that they are able to intelligently adapt themselves to the appropriate conditions of the user and his environment.

In parallel to this technological trend, the richness, diversity and amount of user's information that are constantly uploaded to social networks by the user's base is increasing with each new social network or information service available on the web. This user's information includes among other types of information, contact lists, level of interaction among different users, user interaction with specific services, status tags, activities being done, preferences, and personal reviews about information previously existing.

It becomes necessary to link the context-aware applications in the mobility environment with the diversity of the personal information uploaded to the social networks. That will open up a new concept of services that will be absolutely personalized to the user both in terms of user's context and user's social information.

There is some significant work on this area, mostly oriented to the analysis of user generated content (UGC), [3], as well as context acquisition and management in different user environments [4].

That connection among these two trends will be based on the context acquisition, processing and management capabilities. But also it shall rely on the extraction and processing of the user information, mixing the information provided by experts or individuals with high reputation in a given subject ("wisdom of the few") with the mass information provided by all the existing users ("wisdom of the crowd"). That information extraction shall be based on the information consumer's context, in such a way that the final result is particularly useful for the user.

Nevertheless, in order to provide such capabilities in a real ubiquitous and global manner, telecommunications operators need to play a role on this. Telco operators are one of the main actors in this new market, mainly because they have the need and the opportunity: the need to expand their activities towards the value-added service market, in order not to get stuck in the pure transport activities which are being devaluated day by day.

Telecommunications operators also have the opportunity, given the huge user's base that allows them to handle very detailed and reliable user's information, and also thanks to the infrastructures they already own. This is especially true in the case of convergent operators that can access the user via different end-user devices, including contents consumed through IPTV (IP television), web browsing through PC, applications over mobile handsets as well as fixed telephony services.

Thanks to the global access networks deployed by the telecommunications operators, a lot of key information about users and can be accessed easily through telecommunications services, 3rd party services and lots of sensor networks and service providers to get even more data.

Exhaustive and accurate user data is the key factor empowering intelligent context-aware applications.

Global Project Definition

The work presented in this document is included in a research project of "*Telefonica I+D*". This project is part of the research strategy on services and contextual recommendation systems of "*Telefonica I+D*".

The project of "*Telefonica I+D*" is called *Walkopedia*® and it is part of a commercial product of *O2-UK* (owned by *Telefonica*) called *LookyThing*®. *Walkopedia*® is a backend service developed by "*Telefonica I+D*" in order to provide contextual recommendations depending on mood. *LookyThing*® is a commercial product that consists on a mobile application developed by *LastminuteLabs* iPhone that connects to the *Walkopedia*® service to obtain contextual recommendations about tourist content, venues, services and events.

My work presented in this thesis (Inside *Walkopedia*® Project) is the algorithm design and implementation over: content analysis, information retrieval and data-mining to obtain context inference over content.

In my project, a work that performs an enhanced analysis of the user information in specific data repositories is presented, for the touristic specific case. In this work a complete processing of all the information available for a given item (restaurant, bar, theaters, etc) is performed, by using specific algorithms designed for this purpose. As a result of such processing, the possibility of matching the processed content with contextual information from mobility users is created.

Finally a prototype of the complete system proposed using the algorithms presented on this project was implemented by "*Telefonica I+D*" as a *Walkopedia*® commercial product.

Objectives and proposals

The objective of this project is the study and implementation of algorithms for a mobile service based on the context in mood. The project presents the results of a research about enhanced algorithms to achieve objects contextualization.

In the first instance we wanted to study what is the best system architecture for contextual mobility. Also the study of algorithms is required for acquisition of context on the content.

Finally, we present a comprehensive solution implemented as a product are the conclusions of theoretical studies presented.

The technical proposals to achieve during this project are:

- Obtain content datasets about public venues.
- Item processing using Information retrieval techniques for enhancement content.
- Study of the state of art and design a context aware recommender system.
- Study and design Data Mining algorithms for contextual tagging
    - Association Rules Solution
    - Clustering solution
- Compare algorithms and Analyzing results

Structure Overview

The document of this project is structured as follows:

First of all we find the study of context management and existing knowledge about Context-aware recommender systems (CARS). This State of art study about the CARS will show us the technical reasons that justify the solution presented.

Then on the chapter 3 the document presents the technical solutions and the algorithms designed and implemented on this thesis by me.

On the next chapters we will find the system description of "*Telefonica I+D*" prototype where the research results of this project are applied. Finally the document exposes the results evaluation and the conclusions.

Chapter 2 Context-Aware Recommender Systems

Before explaining the design and implementation work about the algorithms developed on this project is necessary to introduce the current state of the art of context-aware recommendation systems.

Before discussing the role and opportunities of contextual information in recommender systems, we start by discussing the general notion of context. Then, we focus on recommender systems and explain how context is specified and modeled there.

What is Context?

Context is a multifaceted concept that has been studied across different research disciplines, including computer science (primarily in artificial intelligence and ubiquitous computing), cognitive science, linguistics, philosophy, psychology, and organizational sciences.

In fact, an entire conference – CONTEXT (see, for example, http://context-07.ruc.dk) – is dedicated exclusively to studying this topic and incorporating it into various other branches of science, including medicine, law, and business. In reference to the latter, a well-known business researcher and practitioner C. K. Prahalad has stated that “the ability to reach out and touch customers anywhere at any time means that companies must deliver not just competitive products but also unique, real-time customer experiences shaped by customer context” and that this would be the next main issue (“big thing”) for the CRM practitioners (Prahalad 2004) [12]

Since context has been studied in multiple disciplines, each discipline tends to take its own idiosyncratic view that is somewhat different from other disciplines and is more specific than the standard generic dictionary definition of context as “conditions or circumstances which affect something” (Webster 1980). Therefore, there exist many definitions of context across various disciplines and even within specific subfields of these disciplines. Bazire and Brézillon (2005) [24] present and examine 150 different definitions of context from different fields. This is not surprising, given the complexity and the multifaceted nature of the concept.

Since we focus on recommender systems in this work and since the general concept of context is very broad, we try to focus on those fields that are directly related to recommender systems, such as data mining, e-commerce personalization, databases, information retrieval, ubiquitous and mobile context-aware systems, marketing, and management. We follow (Palmisano et al. 2008) [13] [14] in this chapter when describing these areas:

- **Data Mining.**
- **E-commerce Personalization**
- **Ubiquitous and mobile context-aware systems**
- **Databases.**
- **Information Retrieval**
- **Marketing and Management**

As this chapter clearly demonstrates, context is a multifaceted concept used across various disciplines, each discipline taking a certain angle and putting its "stamp" on this concept. To bring some "order" to this diversity of views, Dourish (2004) [15] introduces taxonomy of contexts, according to which contexts can be classified into the representational and the interactional views. In the representational view, context is defined with a predefined set of observable attributes, the structure (or schema, using database terminology) of which does not change significantly over time.

In other words, the representational view assumes that the contextual attributes are identifiable and known a priori and, hence, can be captured and used within the context-aware applications. In contrast, the interactional view assumes that the user behavior is induced by an underlying context, but that the context itself is not necessarily observable. Furthermore, Dourish (2004) [15] assumes that different types of actions may give rise to and call for different types of relevant contexts, thus assuming a bidirectional relationship between activities and underlying contexts: contexts influence activities and also different activities giving rise to different contexts.

Modeling Contextual Information in Recommender Systems

Classics Recommenders

Recommender systems emerged as an independent research area in the mid-1990s, when researchers and practitioners started focusing on recommendation problems that explicitly rely on the notion of *ratings* as a way to capture user preferences for different items.

For example, in case of a movie recommender system, John Doe may assign a rating of 7 (out of 10) for the movie "Gladiator," i.e., set R_{movie}(John_Doe, Gladiator)=7. The recommendation process typically starts with the specification of the initial set of ratings that is either explicitly provided by the users or is implicitly inferred by the system. Once these initial ratings are specified, a recommender system tries to estimate the rating function *R*

$$\boldsymbol{R: User \times Item \rightarrow Rating}$$

for the (user, item) pairs that have not been rated yet by the users. Here *Rating* is a totally ordered set (e.g., non-negative integers or real numbers within a certain range), and *User* and *Item* are the domains of users and items respectively. Once function *R* is estimated for the whole *User*×*Item* space, a

recommender system can recommend the highest-rated item (or *k* highest-rated items) for each user. We call such systems *traditional* or *two-dimensional* (2D) since they consider only the *User* and *Item* dimensions in the recommendation process.

In other words, in its most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. This estimation is usually based on the ratings given by this user to other items, ratings given to this item by other users, and possibly on some other information as well (e.g., user demographics, item characteristics).

Note that, while a substantial amount of research has been performed in the area of recommender systems, the vast majority of the existing approaches focus on recommending items to users or users to items and do not take into the consideration any additional contextual information, such as time, place, the company of other people (e.g., for watching movies).

Context representation in recommender systems

Motivated by this, we explore the area of ***context-aware recommender systems*** **(*CARS*)**, which deal with modeling and predicting user tastes and preferences by incorporating available contextual information into the recommendation process as explicit additional categories of data. These long-term preferences and tastes are usually expressed as *ratings* and are modeled as the function of not only items and users, but also of the context. In other words, ratings are defined with the *rating function* as

$$\boldsymbol{R: User \times Item \times Context \rightarrow Rating},$$

where *User* and *Item* are the domains of users and items respectively, *Rating* is the domain of ratings, and *Context* specifies the contextual information associated with the application. To illustrate these concepts, consider the following example.

> ***Example 1***. Consider the application for recommending movies to users, where users and movies are described as relations having the following attributes:
>
> - Movie: the set of all the movies that can be recommended; it is defined as Movie (MovieID, Title, Length, Release Year, Director, Genre).
> - User: the people to whom movies are recommended; it is defined as User (UserID, Name, Address, Age, Gender, Profession).
>
> Further, the contextual information consists of the following three types that are also defined as relations having the following attributes:

- Theater: the movie theaters showing the movies; it is defined as Theater (TheaterID, Name, Address, Capacity, City, State, Country).
- Time: the time when the movie can be or has been seen; it is defined as Time (Date, DayOfWeek, TimeOfWeek, Month, Quarter, Year).
- Companion: represents a person or a group of persons with whom one can see a movie. It is defined as Companion (companionType) , where attribute companionType has values "alone", "friends", "girlfriend/boyfriend", "family", "coworkers", and "others".

Then the rating assigned to a movie by a person also depends on where and how the movie has been seen, with whom and at what time. For example, the type of movie to recommend to college student Jane Doe can differ significantly depending on whether she is planning to see it on a Saturday night with her boyfriend vs. on a weekday with her parents.

As we can see from this example and other cases, the contextual information *Context* can be of different *types*, each type defining a certain aspect of context, such as time, location (e.g., Theater), companion (e.g., for seeing a movie), purpose of a purchase, etc. Further, each contextual type can have a complicated structure reflecting complex nature of the contextual information. Although this complexity of contextual information can take many different forms, one popular defining characteristic is the *hierarchical* structure of contextual information that can be represented as trees, as is done in most of the context-aware recommender and profiling systems, including (Adomavicius et al. 2005) [17] and (Palmisano et al. 2008) [13].

For instance, the three contexts can have the following hierarchies associated with them:

***Theater**:* TheaterID → City → State → Country;
***Time**:* Date → DayOfWeek → TimeOfWeek;
Date → Month → Quarter → Year.

Furthermore, we follow the representational view of (Dourish 2004) [15], and assume that the context is defined with a predefined set of observable attributes, the structure of which does not change significantly over time. Although there are some papers in the literature that take the interactional approach to modeling contextual recommendations, such as (Anand and Mobasher 2007) [22] that models context through a short-term memory (STM) interactional approach borrowed from psychology, most of the work on context-aware recommender systems follows the representational view. As stated before, we also adopt this representational view and assume that there is a predefined finite set of contextual types in a given application and that each of these types has a well-defined structure.

More specifically, we follow (Palmisano et al. 2008) [14], and also (Adomavicius et al. 2005) [17] [19] to some extent, in this work and define the contextual

information with a *set* of *contextual dimensions* ***K***, each contextual dimension ***K*** in ***K*** being defined by a set of *q* attributes $\boldsymbol{K = (K_1,..., K_q)}$ having a *hierarchical structure* and capturing a particular *type* of context, such as *Time*, or *CommunicatingDevice*. The values taken by attribute K_q define *finer* (more *granular*) levels, while K_1 values define *coarser* (less granular) levels of contextual knowledge.

Contextual information was also defined in (Adomavicius et al. 2005) [17] as follows. In addition to the classical *User* and *Item* dimensions, additional contextual dimensions, such as *Time*, *Location*, etc., were also introduced using the OLAP- based *multidimensional data (MD) model* widely used in the data warehousing applications in databases (Kimball 1996, Chaudhuri and Dayal 1997) [23]. Formally, let D_1, D_2, ..., D_n be dimensions, two of these dimensions being *User* and *Item*, and the rest being contextual.

Each dimension *Di* is a subset of a Cartesian product of some attributes (or fields)

$$\boldsymbol{A_{ij}, (j = 1,...,k_i),}$$
$$\boldsymbol{Di \subseteq A_{i1} \times A_{i2} \times ... \times A_{iki}}$$

where each attribute defines a domain (or a set) of values. Moreover, one or several attributes form a *key*, i.e., they uniquely define the rest of the attributes (Ramakrishnan and Gehrke 2000). In some cases, a dimension can be defined by a single attribute, and $k_i = 1$ in such cases. For example, consider the threedimensional recommendation space

$$\boldsymbol{User \times Item \times Time}$$

where the *User* dimension is defined as:

$$\boldsymbol{User \subseteq UName \times Address \times Income \times Age}$$

and consists of a set of users having certain names, addresses, incomes, and being of a certain age. Similarly, the *Item* dimension is defined as:

$$\boldsymbol{Item \subseteq IName \times Type \times Price}$$

and consists of a set of items defined by their names, types and the price. Finally, the *Time* dimension can be defined as:

$$\boldsymbol{Time \subseteq Year \times Month \times Day}$$

and consists of a list of days from the starting to the ending date (e.g. from January 1, 2003 to December 31, 2003).

Given dimensions D_1, D_2, ..., D_n, we define the *recommendation space* for these dimensions as a Cartesian product $S = D_1 \times D_2 \times ... \times D_n$. Moreover, let *Rating* be a rating domain representing the ordered set of all possible rating values. Then the *rating function* is defined over the space $D_1 \times ... \times D_n$ as

$$R: D_1 \times ... \times D_n \rightarrow Rating.$$

For instance, continuing the *User*×*Item*×*Time* example considered above, we can define a rating function *R* on the recommendation space *User*×*Item*×*Time* specifying how much user $u \in User$ liked item $i \in Item$ at time $t \in Time$, $R(u,i,t)$.

Visually, ratings $R(d_1,...d_n)$ on the recommendation space $S = D_1 \times D_2 \times ... \times D_n$ can be stored in a multidimensional cube, such as the one shown in Figure 2. For example, the cube in Figure 2 stores ratings $R(u,i,t)$ for the recommendation space *User*×*Item*×*Time*, where the three tables define the sets of users, items and times associated with *User*, *Item*, and *Time* dimensions respectively.

For example, rating $R(101,7,1) = 6$ in Figure 2 means that for the user with User ID 101 and the item with Item ID 7, rating 6 was specified during the weekday.

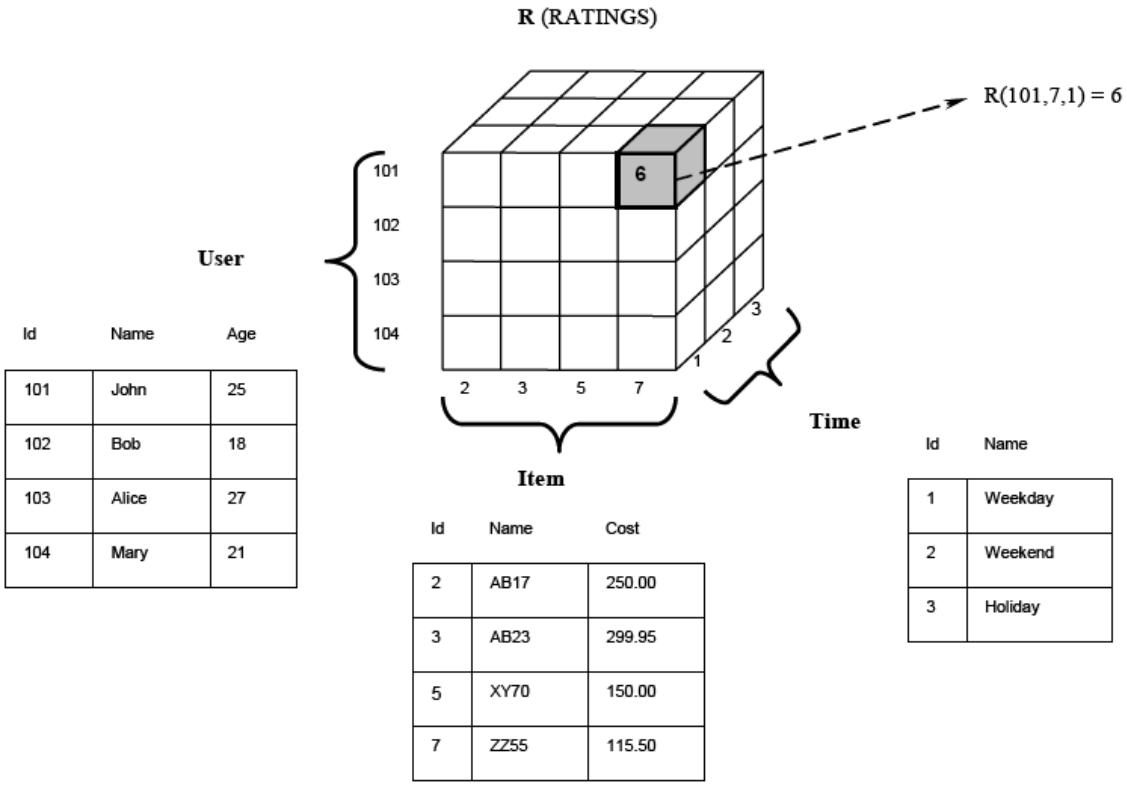

Figure 2 - 3D recommendation space [reference 18]

The rating function *R* introduced above is usually defined as a partial function, where the initial set of ratings is known. Then, as usual in recommender systems, the goal is to estimate the unknown ratings, i.e., make the rating function *R* total.

The main difference between the multidimensional (MD) contextual model described above and the previously described contextual model lies in that contextual information in the MD model is defined using classical OLAP hierarchies, whereas the contextual information in the previous case is defined

with more general hierarchical taxonomies, that can be represented as trees (both balanced and unbalanced), directed acyclic graphs (DAGs) or various other types of taxonomies. Further, the ratings in the MD model are stored in the multidimensional cubes, whereas the ratings in the other contextual model are stored in more general hierarchical structures.

We would also like to point out that not all contextual information might be relevant or useful for recommendation purposes. Consider, for example, a book recommender system. Many types of contextual data could potentially be obtained by such a system from book buyers, including:

a) Purpose of buying the book (possible options: for work, for leisure, ...)
b) Planned reading time (weekday, weekend, ...)
c) Planned reading place (at home, at school, on a plane, ...)
d) The value of the stock market index at the time of the purchase.

Clearly some types of contextual information can be more relevant in a given application than some other types, and there are several approaches to determining the relevance of a given type of contextual information.

In particular, the relevance determination can either be done *manually*, e.g., using domain knowledge of the recommender system's designer or a market expert in a given application domain, or *automatically*, e.g., using numerous existing feature selection procedures from:

- **machine learning** (Koller and Sahami 1996) [25]
- **data mining** (Liu and Motoda 1998) [26]
- **statistics** (Chatterjee et al. 2000) [27]

based on existing ratings data during the data preprocessing phase.

Obtaining Contextual Information

The contextual information can be obtained in a number of ways, including:

- **Explicitly**, i.e., by directly approaching relevant people and other sources of contextual information and explicitly gathering this information either by asking direct questions or eliciting this information through other means. For example, a website may obtain contextual information by asking a person to fill in a web form or to answer some specific questions before providing access to certain web pages.

- **Implicitly** from the data or the environment, such as a change in location of the user detected by a mobile telephone company. Alternatively, temporal contextual information can be implicitly obtained from the timestamp of a transaction. Nothing needs to be done in these cases in terms of interacting with the user or other sources of contextual information – the source of the implicit contextual information is accessed directly and the data is extracted from it.

- **Inferring** the context using statistical or data mining methods. For example, the household identity of a person flipping the TV channels (husband, wife, son, daughter, etc.) may not be explicitly known to a cable TV company; but it can be inferred with reasonable accuracy by observing the TV programs watched and the channels visited using various data mining methods. In order to infer this contextual information, it is necessary to build a predictive model (i.e., a classifier) and train it on the appropriate data. The success of inferring this contextual information depends very significantly on the quality of such classifier, and it also varies considerably across different applications. For example, it was demonstrated in (Palmisano et al. 2008) [14] that various types of contextual information can be inferred with a reasonably high degree of accuracy in certain applications and using certain data mining methods, such as Naïve Bayes classifiers and Bayesian Networks.

Finally, the contextual information can be "hidden" in the data in some latent form, and we can use it implicitly to better estimate the unknown ratings without explicitly knowing this contextual information.

For instance, in the previous example, we may want to estimate how much a person likes a particular TV program by modeling the member of the household (husband, wife, etc.) watching the TV program as a latent variable. It was also shown in (Palmisano et al. 2008) [14] that this deployment of latent variables, such as intent of purchasing a product (e.g., for yourself vs. as a gift, work-related vs. pleasure, etc.), whose true values were unknown but that were explicitly modeled as a part of a Bayesian Network (BN), indeed improved the predictive performance of that BN classifier.

Therefore, even without any explicit knowledge of the contextual information (e.g., which member of the household is watching the program), recommendation accuracy can still be improved by modeling and inferring this contextual information implicitly using carefully chosen learning techniques (e.g., by using latent variables inside well designed recommendation models). A similar approach of using latent variables is presented in (Anand and Mobasher 2007).

We focus on the representational view of (Dourish 2004) [15], and assume that the context is defined with a predefined set of contextual attributes, the structure of which does not change over time. The implication of this assumption is that we need to identify and acquire contextual information before actual recommendations are made. If the acquisition process of this contextual information is done explicitly or even implicitly, it should be conducted as a part of the overall data collection process. All this implies that the decisions of which contextual information should be relevant and collected for an application should be done at the application design stage and well in advance of the time when actual recommendations are provided.

One methodology of deciding which contextual attributes should be used in a recommendation application (and which should not) is presented in (Adomavicius et al. 2005) [17]. In particular, (Adomavicius et al. 2005) [18] propose that a wide range of contextual attributes should be initially selected by the domain experts as possible candidates for the contextual attributes for the application. For example, in a movie recommendation application described in *Example 1*, we can initially consider such contextual attributes as Time, Theater, Companion, Weather, as well as a broad set of other contextual attributes that can possibly affect the movie watching experiences, as initially identified by the domain experts for the application.

Then, after collecting the data, including the rating data and the contextual information, we may apply various types of statistical tests identifying which of the chosen contextual attributes are truly significant in the sense that they indeed affect movie watching experiences, as manifested by significant deviations in ratings across different values of a contextual attribute. For example, we may apply pairwise t-tests to see if good weather vs. bad weather or seeing a movie alone vs. with a companion significantly affect the movie watching experiences (as indicated by statistically significant changes in rating distributions).

This procedure provides an example of screening all the initially considered contextual attributes and filtering out those that do not matter for a particular recommendation application. For example, we may conclude that the Time, Theater and Companion contexts matter, while the Weather context does not in the considered movie recommendation application.

Context-Aware Architectures

To start the discussion of the contextual preference elicitation and estimation techniques, note that, in its general form, a traditional 2-dimensional (2D) (User×Item) recommender system can be described as a function, which takes partial user preference data as its input and produces a list of recommendations for each user as an output. Accordingly, Figure 3 presents a general overview of the traditional 2D recommendation process, which includes three components: data (input), 2D recommender system (function), and recommendation list (output).

Note that, as indicated in Figure 3, after the recommendation function is defined (or constructed) based on the available data, recommendation list for any given user u is typically generated by using the recommendation function on user u and all candidate items to obtain a predicted rating for each of the items and then by ranking all items according to their predicted rating value. Later in this chapter, we will discuss how the use of contextual information in each of those three components gives rise to three different paradigms for context-aware recommender systems.

Figure 3 - 2D recommender architecture [reference 18]

Traditional recommender systems are built based on the knowledge of partial user preferences, i.e., user preferences for some (often limited) set of items, and the input data for traditional recommender systems is typically based on the records of the form <user, item, rating>. In contrast, context-aware recommender systems are built based on the knowledge of partial contextual user preferences and typically deal with data records of the form <user, item, context, rating>, where each specific record includes not only how much a given user liked a specific item, but also the contextual information in which the item was consumed by this user (e.g., context = Saturday).

Also, in addition to the descriptive information about users (e.g., demographics), items (e.g., item features), and ratings (e.g., multi-criteria rating information), context-aware recommender systems may also make use of additional context attributes, such as context hierarchies Based on the presence of this additional contextual data, several important questions arise: How contextual information should be reflected when modeling user preferences? Can we reuse the wealth of knowledge in traditional (non-contextual) recommender systems to generate context-aware recommendations? We will explore these questions in this chapter in more detail.

In the presence of available contextual information, following the diagrams in Figure 4, we start 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... for each user.

However, unlike the process in Figure 4, which does not take into account the contextual information, we can apply the information about the current (or desired) context c at various stages of the recommendation process. More specifically, the context-aware recommendation process that is based on contextual user preference elicitation and estimation can take one of the three forms, based on which of the three components the context is used in, as shown in Figure 4:

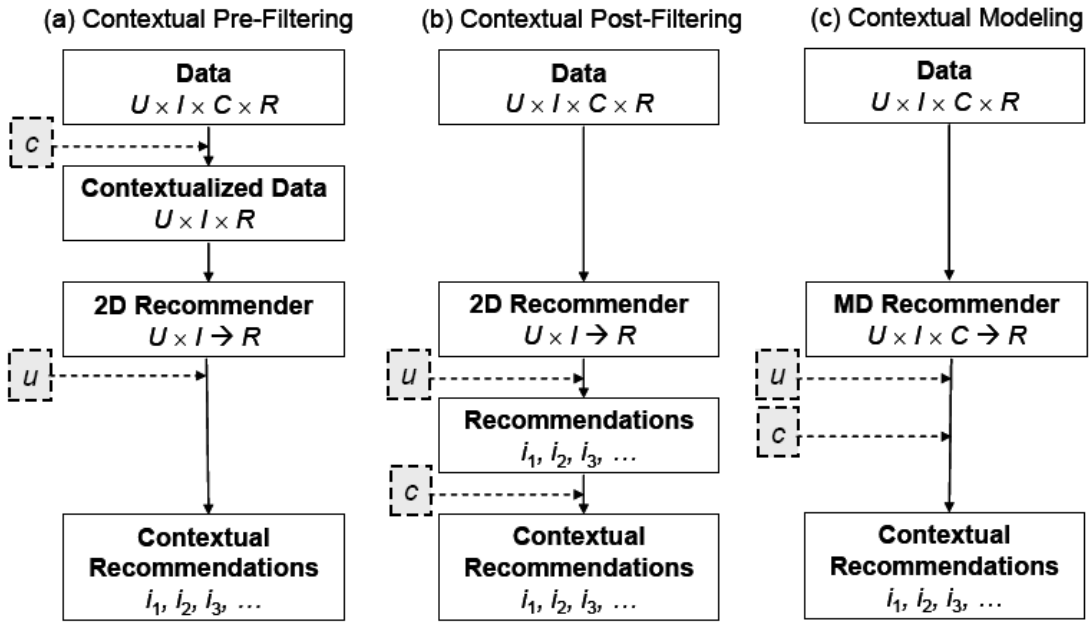

Figure 4 - Context-Aware architectures [reference 18]

Contextual pre-filtering (or contextualization of recommendation input). In this recommendation paradigm (presented in Figure 4*a*), contextual information drives data selection for that specific context. In other words, information about the current context *c* is used for selecting only the relevant set of data records. 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 (presented in Figure 4*b*), 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 (presented in Figure 4*c*), contextual information is used directly in the modeling technique as part of rating estimation.

Contextual Post-Filtering

In the presented architectures, the selected architecture for our system is the ***Contextual Post-Filtering***. This means that the algorithms presented in this project have the aim of: Feeding a recommender and filtering content according to context on the output of this recommender.

The definition of a Post-Filtering Contextual architecture on a Context-Aware system is as follows:

As shown in Figure 4*b*, the contextual post-filtering approach ignores context information in the input data when generating recommendations, i.e., when generating the ranked list of all candidate items from which any number of top-N recommendations can be made, depending on specific values of N. Then, the contextual post-filtering approach adjusts the obtained recommendation list for each user using contextual information. The recommendation list adjustments can be made by:

- Filtering out recommendations that are irrelevant (in a given context), or
- Adjusting the ranking of recommendations on the list (based on a given context).

For example, in a movie recommendation application, if a person wants to see a movie on a weekend, and on weekends she only watches comedies, the system can filter out all non-comedies from the recommended movie list. More generally, the basic idea for contextual post-filtering approaches is to analyze the contextual preference data for a given user in a given context to find specific item usage patterns (e.g., user Jane Doe watches only comedies on weekends) and then use these patterns to adjust the item list, resulting in more contextual" recommendations, as depicted in Figure 5.

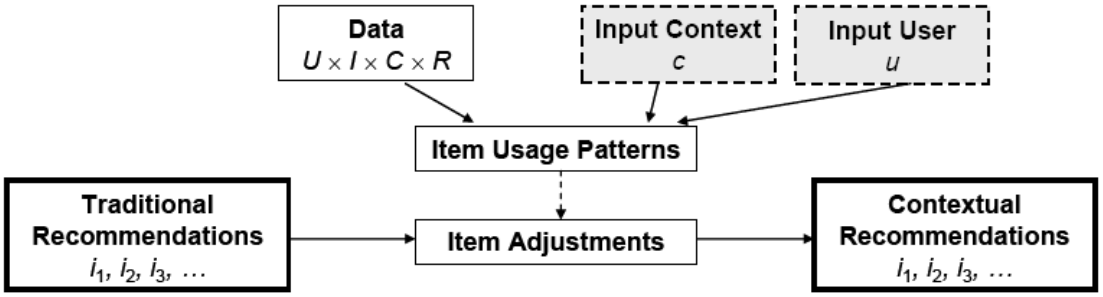

Figure 5 - Final phase of the contextual post-filtering approach: recommendation list adjustment [reference 18]

As with many recommendation techniques, the contextual post-filtering approaches can be classified into heuristic and model-based techniques. Heuristic post-filtering approaches focus on finding common item characteristics (attributes) for a given user in a given context (e.g., preferred actors to watch in a given context), and then use these attributes to adjust the recommendations, including:

- Filtering out recommended items that do not have a significant number of these characteristics (e.g., to be recommended, the movies must have at least two of the preferred actors in a given context), or
- Ranking recommended items based on how many of these relevant characteristics they have (e.g., the movies that star more of the user's preferred actors in a given context will be ranked higher).

In contrast, model-based post-filtering approaches can build predictive models that calculate the probability with which the user chooses a certain type of item in a given context, i.e., probability of relevance (e.g., likelihood of choosing movies of a certain genre in a given context), and then use this probability to adjust the recommendations, including:

- Filtering out recommended items that have the probability of relevance smaller than a pre-defined minimal threshold (e.g., remove movies of genres that have a low likelihood of being picked), or
- Ranking recommended items by weighting the predicted rating with the probability of relevance.

Panniello et al. (2009) [14] provide an experimental comparison of the exact pre-filtering method versus two different post-filtering methods – Weight and Filter – using several real-world e-commerce datasets.

The Weight post-filtering method reorders the recommended items by weighting the predicted rating with the probability of relevance in that specific context, and the Filter post-filtering method filters out recommended items that have small probability of relevance in the specific context. Interestingly, the empirical results show that the Weight post-filtering method dominates the exact pre-filtering, which in turn dominates the Filter method, thus, indicating that the best approach to use (pre- or post-filtering) really depends on a given application.

As was the case with the contextual pre-filtering approach, a major advantage of the contextual post-filtering approach is that it allows using any of the numerous traditional recommendation techniques previously proposed in the literature (Adomavicius and Tuzhilin 2005) [18] [20] . Also, similarly to the contextual pre-filtering approaches, incorporating context generalization techniques into post-filtering techniques constitutes an interesting issue for future research.

Chapter 3 Mood-based contextual recommendation in mobility environments

Solution definition

In the previous chapter 2, I presented the current state of the art context of recommender systems to understand the solution proposed in this project.

So the solution chosen is the "**Contextual Post-Filtering**" explained in the previous chapter as shown in Figure 6
.

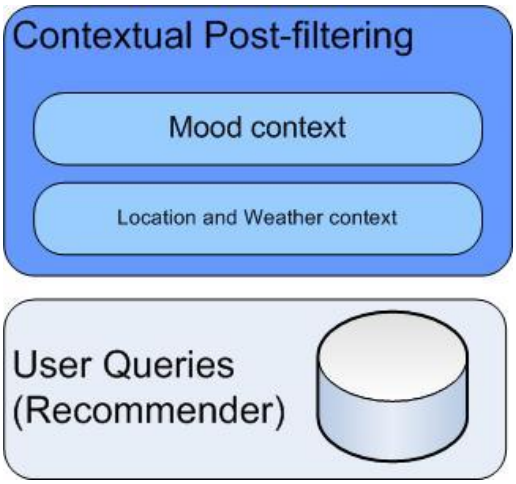

Figure 6 - Contextual post-filtering

When we know the theoretical solution chosen and the state of the art on context-aware recommenders research in this chapter are presented the design and implementation of the necessary algorithms to implement a contextual Post-Filtering to obtain the mood-based contextual recommender system prototype.

The presented algorithms can be differentiated into two necessary functions for the system:

In the foreground are presented the algorithms for information retrieval and content enhancement required by the system to obtain the context inference.

Secondly, are presented two different algorithms with different behaviors but the same purpose to obtain context inference and context filtering about the content.

Use case description

At the time of writing, application developers try to increase the use of this information to enhance the level of personalization of the applications developed. Such information will be useful not only to get user contextual environment but also to contextualize the information contents that are used by the applications in real time.

Such information contents are referred to as the content items. Given the current technology trend, services and applications will progressively become more dependant of the context and, for instance, those that could help an outsider located in a strange city, could be much more useful if they use available contextual information of the user [5].

If we consider a user that is located in a given location, looking for something to do during some spare time available. It is possible to design a big database of activity items and a recommendation system.

However, if such a system tries to get the contextual environment of the user in order to provide better recommendations, it makes a big difference to get the contextual environment conditions of the potentially recommendable items. This way, the user's context and the item context can be matched as a part of the recommendation mechanism itself, in order to get enhanced recommendation results.

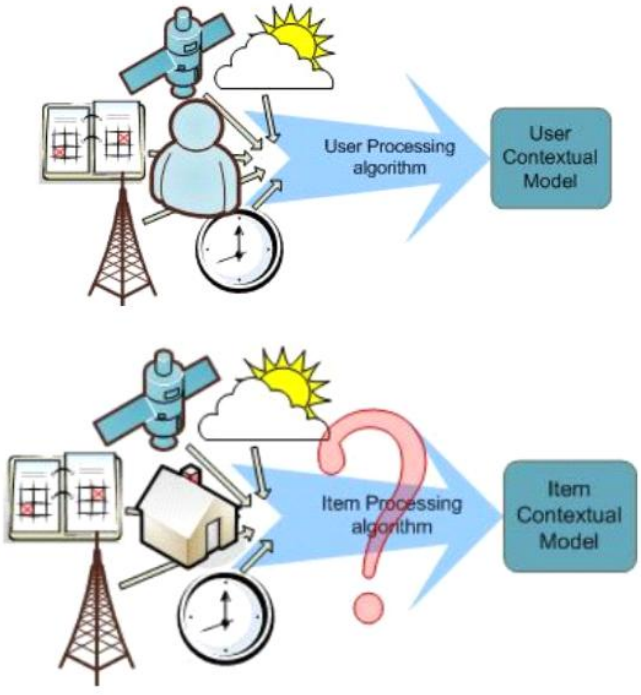

Figure 7 - Context acquisition diagram

So the objective is to monitor the situation of the user to extract such information and process that in order to obtain the context of both the user and the item consumed by him. Although there is quite a lot of work on context information monitoring, in real environments such information is usually related to aspects such location, weather or current date/time. Even though a significant research on how to use this information to get a better model of a user is available, a parallel research to define the procedures for the context information to be useful to model a content item is required. As has been presented, it will enable enhanced contextual recommendation mechanisms.

As shown in Figure 7, different sources of information can be used to model the context of the user. In order to take advantage of such techniques, a similar procedure is applied to the processing of item context. During the recommendation process it will be possible to get a similarity relationship value between context of the items and the user, and provide a context-based recommendation composed by items that are below a distance threshold in the context space.

A contextual model of an item taking its contextual information like the item location (or the location in which it is commonly consumed), the local weather or what is the user's feedback about the specific item among other types of information.

In the following chapters it will be described the proposed mechanisms used to create this context item model that can be applied to all recommendable items that are available. The use case for which this research has been performed is the context-based recommendation scenario of items (restaurants, bars, cinemas, etc) to be consumed by users in a mobility environment. The key contextual dimension to be modeled is the user's mood. That information will be obtained from the user

This is a usual situation very common in tourism in an unknown city or even in a city that the user knows, but in a given area that is new for the user that consumes the service.

Item processing algorithm

In order to obtain a context-aware recommendation service based on user's mood in a mobility environment, it is a must to perform item's context extraction and processing. Such information acquisition requires of a reliable source of descriptive data for each item, both social information as well as expert information about the item.

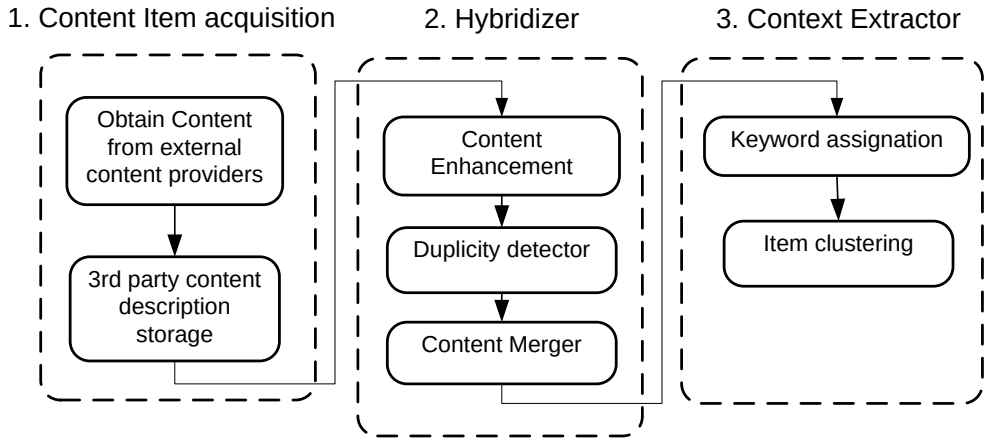

Figure 8 - Item processing flow

The processing flow proposed in the system is depicted in Figure 8.

Firstly, the acquisition of the items from different information sources available is required. Such items will be the different places to go, leisure activities in a given area. That information is usually available from different content providers on the internet. Secondly, such raw information shall be jointly processed to identify items that are actually the same, but coming from different providers, as well as data format normalization. Finally, the contents are labeled with contextual corresponding mood, by following different techniques that will be presented in later sections.

The following part of document presents the algorithms designed and implemented by processing the content. This is the main research on this project and focuses on tree different functions of processing:

- **Content acquisition**: Obtain data from different sources.

- **Hybridizer Module**: At first we need to recover necessary information to detect duplicate content and improving the data.

- **Context Extractor**: Secondly we need the inference of context on the content from the content data itself. To achieve this inference two algorithms different are designed and implemented. These algorithms have the same goal but different performances that are complementary. The algorithms are called: *Keyword assignation* and *Item Clustering*

Content item acquisition

Such data can be obtained from different collaborative servers across the Internet as we can see in Figure 9. The item description and the user rating of each item increases significantly day by day, being the geopositioned items one of the most common ones.

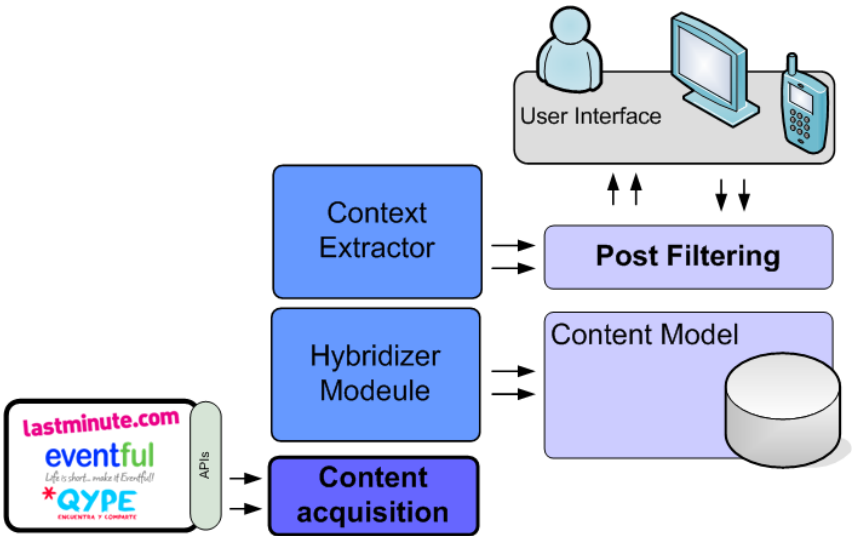

Figure 9 - Content acquisition

Given that the use case described is a mobility scenario, the geopositioned items are the ones in which we are mostly interested in. After a quick analysis of the available items, in the tourism framework, a quick classification of them can be made, into Venues (restaurants, bars, pub, parks, etc) and Events (concerts, theatre performances, movies, etc). Such geopositioned items usually carry associated information, both descriptive as well as valorative.

Two types of such information can be identified for a given item: Item description coming from an expert (in the case of a restaurant, that would be the owner), and social information about the item (that would be the reviews of the restaurant from the users).

The main problem with collaborative content is the level of dispersion and repetition of the data about a given item. In order to solve such issue and improve the quality of the data, it becomes necessary to detect such duplicities and aggregate social information in a smart manner.

Hybridizer Module

The function of the hybridizer module is to enhance the quantity and quality of information, both descriptive and valorative, about the geopositioned items.

This is a critical step that will impact the quality of the final results, so a step-wise procedure is followed (Figure 10).

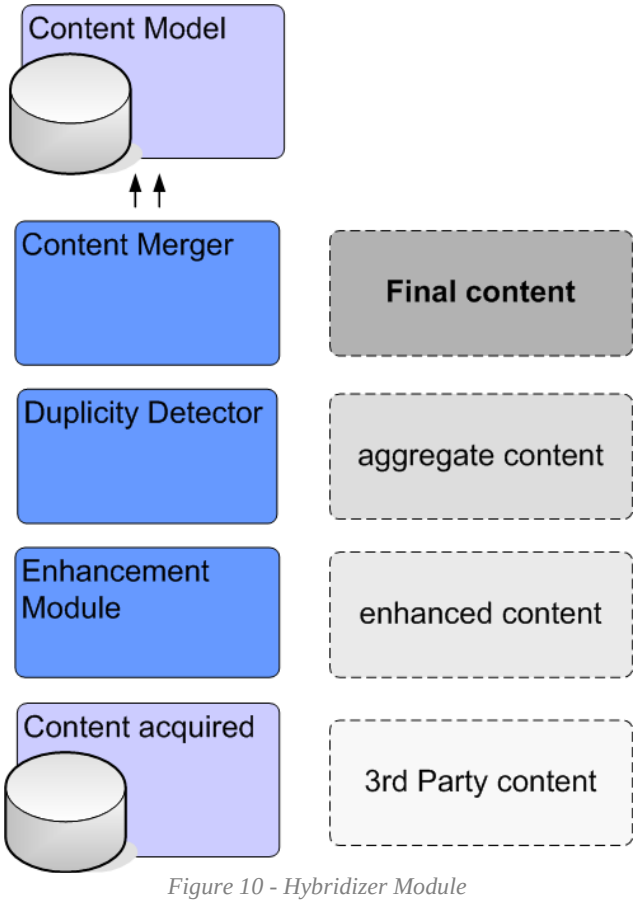

Figure 10 - Hybridizer Module

Enhancement Module

Hybridizer module enhances the item contents obtained from content providers external to the system. The contents retrieved are mostly of social nature (user reviews, etc), so the integrity and uniformity of the information is not guaranteed.

On this regard, the content enhancements are the following:

- Time enhancement. The format of time and date is usually very diverse, and it needs to be normalized.

- Description parsing. The item description is usually an open free text, so it may include date information, URL, pricing, etc. That information shall be extracted through regular expressions.

- Hash generation. In order to make it easier a later duplicate identification, a unique item Hash function is generated from the title, description and location.

Duplicity Detector

The hybridizer module also includes the capability of duplicate detection among the different items obtained from external providers. Such capability permits to obtain a single copy of each item and complete the description with all information coming from the associated sources.

In the implemented data model, it is separated the content that is generated internally (by aggregating external information) from the items directly obtained from the content providers. Based on such structure, it is differentiated:

- Providing content: exact definition of the Event or the Venue as it is obtained from external content providers.

- Item content: definition of the Event or Venue generated internally in the processing system from the providing contents.

The duplicity detector uses different methods with the following functionality:

- **Same providing.** It detects the duplicity among contents already existing in the system database. That is performed by checking whether the same item has already been retrieved from external provider in order to update the content.

- **Same item.** The duplicity of providing contents is checked across the set of content providers from which the original providing content is coming from. That duplicity is performed through the Hash function obtained from the title, description and location.
  Such duplicity identification mechanism is very useful as it is able to aggregate contents that refer to the same content item, but coming from different external content providers. For instance it is very common that the same event happening in different days of the week are considered different separated items.

- **Similar item.** It detects the duplicity of “providings”, obtained from different sources, that are very similar one to the other, or that actually refer to the same item. Given that the item information is usually coming from the users, the descriptive difference may be significant, as it happens usually with the UGC (User Generated Content).

Based on this, the duplicity detector shall include a distance-based algorithm to infer the similarity of two separate items depending on the result o a cascade comparison [6] of the descriptions and ratings of the item. The algorithm executed will be different based on the type of item.

In the case of an event, initially, similar items after a Stemming [7]/Stop words processing over the item title are selected as candidates to be the same item.

The second step executed for the events, aims to obtain a closer distance value as a result of how near the items are located, based on geographic coordinates, as they are all geolocated items. The information used for such a purpose is very diverse, as the longitude, latitude, postal code, street name, etc.

Finally, a least distance value procedure is applied over all the information associated to the item candidates. That way any overlapping will be detected and managed properly.

In the case of the venues, the two first steps are executed in the reverse order. The geographical filtering is executed in the first place, as it is the most discriminating phase for venues. The third step is finally executed for venues also.

Content merger

When content duplicity situations are detected, based on previous procedures, it becomes necessary to combine the information coming from the different sources into one single item into the system database.

Such combination will allow us to handle a much richer set of information for each of the items processed as well as generate specific inferred information about the items that cannot be obtained by any external information source by itself.

Context Extractor Algorithms

Thanks to the hybridizer module, the quantity and quality of information about the items is significant, and is an excellent dataset for the processing algorithm that provides contextual information for each item (Figure 11).

Such context inference techniques for the items based on the descriptive information are described in the next chapter.

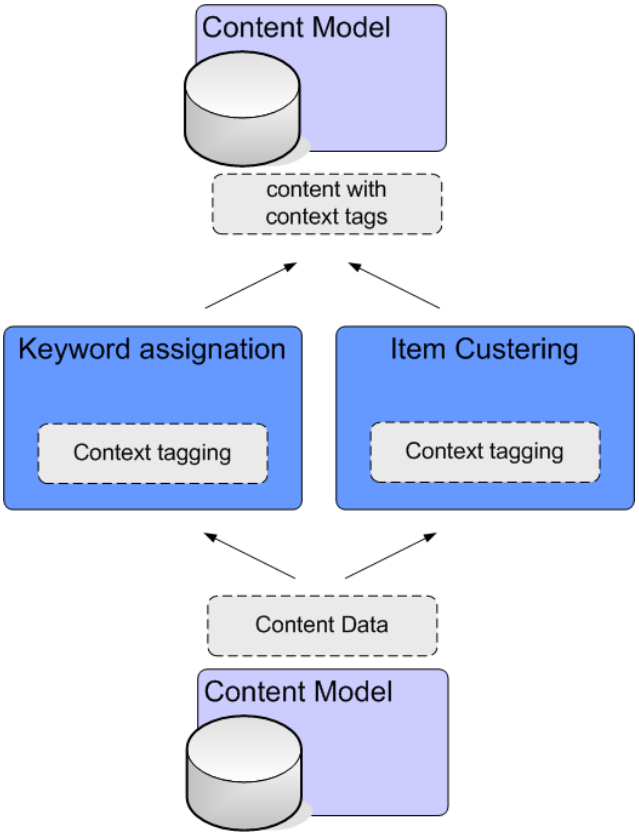

Figure 11 - Context Extractor Algorithms

Keyword assignation

One of the methods used to assign context (in our specific use case that would be user mood) to the available items is performed from all the metadata associated to the item. That metadata are the categories, tags, reviews or descriptions, which have been aggregated after the procedure described in the previous chapter.

That is based on keyword search mechanism, by using regular expressions and statistical methods [8]. The purpose of this mechanism is to obtain a set of user moods associated to each item. As a starting point, it is defined a discrete set of possible user moods.

The user mood for a given item is obtained with the following mechanism.

- If a given item tag matches perfectly with a given user mood or a synonym, the mood is assigned to the item.

- If a user review contains a word that matches perfectly a user mood or a synonym, that mood is appended to the item's list of moods. In order to get a minimum false-positives noise, the specific mood shall be present in a configurable percentage of the user reviews (in the presented implementation we have tested a value of 15% with good results).

- Finally, the item category is also used to map it to one or more user moods.

It is important to clarify that the quality and reliability of the results of this procedure rely on the previous hybridizing mechanisms that provide a very rich data set for each item. That way the contextual processing will provide results significantly better.

There are several critical points in the algorithm that extracts user moods from the item information.

- All the synonyms of the user moods, as well as the set of moods itself, is key for the whole procedure. So that information shall be defined by experts.

- The keyword assignation mechanism followed to assign items to moods is quick and not very costly in terms of processing resources. However it has important drawbacks. Its reliability is based on a minimum number of user reviews per item.

- Synonyms assigned to more than one user mood that may drive to unstable situations and not proper mood assignments.

Some solutions to these issues can be:

- Global statistical models [9] that evaluate all available words, getting a score for each mood, reflecting how likely it is the item to be associated to each mood. That will depend on the global combination of words across item tags and reviews.

- Linguistic models that process the words with lexical resources (dictionaries, etc), and calculate the semantic distances to the concepts that define each user mood. The processing of the user reviews can then be improved with a text pre-parsing that may allow to tune or pre-process the sentences.

Item clustering

Based on the previous issues and drawbacks, the proposed system includes an enhanced solution to extract context from the item content using a Reputation-based item clustering.

The general idea is to use a vectorial space model to index the items, and then train a statistical classifier. The classification of the descriptive vectors of the content items will generate a space where the context will be extracted.

The main advantage of this proposal is that it is a global conceptual architecture that provides a lot of flexibility to handle potentially any kind of types of context and contents.

This proposal can be described into several phases as we can see on Figure 12.

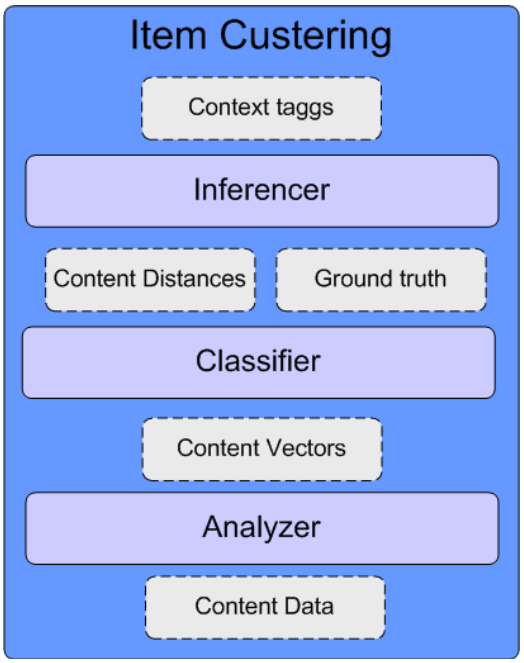

Figure 12 - Item clustering

I. ***Analyzer.***

Each item will be considered as an n-dimensional vector. That vector will be analyzed to be indexed, searched, classified, etc. Each vector dimension is a different word. So the maximum dimension of the vector space is the maximum number of different words available across all the items associated information. The analyzer element will extract the vector out of each item. It is convenient to implement such element in two separate steps, indexer and vectorizer.

The indexer will generate a separate index file, by processing the items associated text and information. The flow implemented by the indexer is the following.

- Text processing, cleaning strings that are not to be used (in the presented implementation these were the HTML tags).

- Tokenizer. Separate the texts into words.

- Stop words. Delete from the text the words with no semantic meaning (articles, pronouns, etc).

- Stemming. Delete non-semantic suffixes & prefixes, leaving just the root of the word.

- Term weighting. Ponderate the impact of each word occurrence, by following standard procedures, like TF-IDF (term frequency–inverse document frequency) [10]. The source of each occurrence (tag, review, title, description) may impact the weight.

- Indexing. Store each term occurrence in an index file for each item.

The vectorizer will generate the vector of each item from the index file. The vectors will be generated in the proper format to feed the chosen classifier. The vector dimensions will depend on the concepts obtained in the index phase.

II. *Classifier*

The role of the classifier function is to calculate the distance among the vectors obtained previously, in order to cluster them accordingly. In the presented implementation, a classical algorithm is used, kNN (k-nearest neighbors algorithm) [11] with Euclidean distance.

III. Inferencer.

Once the content is fully analyzed and the classifier can be fed with vectors to arrange the content based on the similarities, the classification shall be used to extract context from the item.

In order to extract such context information, a training items set is required. Such set shall be tagged with user moods by experts called ground truth. Such training set will be used to assign context labels to all the rest of contents as per the implemented classification.

Chapter 4 System prototype

In order to demonstrate the presented algorithms, a prototype is developed with the purpose of getting a complete system with the capability of obtaining and processing items and finally publishing a service for mobile users.

The system is implemented in a server with Ubuntu 9.10 32bits with a 2,5Ghz Core 2 Duo processor with 2Gb of RAM.

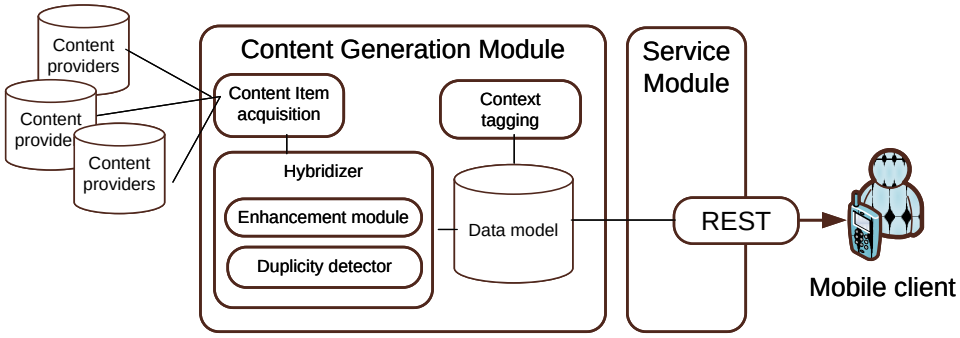

Figure 13 - System prototype diagram

As per the database technology, MySQL Server version 5.1.37 was chosen. Tables were created in MyISAM type, useful to be able to activate GIS (Geographic Information Support), needed to be able to detect items inside a given area in a efficient way.

The Backend itself is developed using Java technology. Specifically the Backend was developed on top of a Spring Framework. Spring increase development productivity and runtime performance while improve test coverage and application quality.

In order to publish the service to external clients, a RESTful (Representational State Transfer) web API was developed. It is a collection of resources with three defined aspects: a defined base use url, the MIME type of the data supported by the web service (JSON, XML, etc.) and the set of operations supported by the web service using HTTP methods (e.g., POST, GET, PUT or DELETE).

Logical architecture

There are two main modules involved in the architecture of the prototype. A main module is the one that implements the database, the Content Generation Module. A separate module allocates the service itself, runs the business logic and it is called Service Module. Every module can be launched independently, so the system can work in content generation activities while the service is online for the users to access.

This is very important because the content generation module is an activity that can take potentially a significant amount of time, depending on the amount of information sources to integrate in the system.

A. Content Generation Module

This is the module where the item processing algorithm, as has been described in detail in previous sections, is placed. Its purpose is to create a complete database that Service Module will use to give contextual user recommendations to the mobile clients.

Firstly the prototype database shall be fully populated with content items, so an interface is developed to integrate different content providers. Through such interface module can integrate any content provider available on the internet. This interface capability is used by the Content Acquisition stage presented in chapter 2. Each content provider has his different way of store item data (i.e. data formats), so the Content Enhancement stage is included to clean and homogenize the content providers information.

At this point, the Hybridizer module, defined in chapter 3 is invoked, in order to create a common representation of the items downloaded via the different content providers. Finally, the mood tagging module is executed to associate a list of moods to every item.

B. Service Module

The Service Module implements the interface with mobile clients. Accordingly, it will be in charge of managing the connections, access restrictions and data representation for clients, etc.

As shown in Figure 13, Service Module receives all incoming connections from mobile users. Depending on the request URI path, every connection is derived to the correct internal module to handle users, items, recommendations, etc. After checking the user's credentials, the Data Model is accessed. Such access is implemented with Hibernate (open source Java persistence framework), and will manage all the queries to the system, caches and connection pool.

In early stages of the implementation it was clearly identified that the geopositioned database queries were a significant bottleneck in terms of performance and resources, so GIS libraries-assisted queries are implemented.

Example use of mobile-server interaction

The message interchange between client and server is presented at this point. User registration and initialization would be required but is not included in the flow for simplicity.

The main call is the recommendation retrieval. In this request the user sends his position, desired radius and time for the recommendation, as well as user mood:

http://domain.com/moodprototype/user/{user_id}/reco?longitude={longitude}&latitude={latitude}&time={time}&radius={radius}&mood={mood}

The user's mood may be selected following different procedures of context acquisition or retrieval.

- Explicit obtention: the user can select the mood manually or through some kind of questions game.
- Profiling technique: a profiling mechanism may be used in order to get the most likely mood that the user may have.

Some other techniques to get the context (in this case, the mood) of the user may also apply.

The client will retrieve a JSON from the server with items inside the desired place and time ranges, including the following information per item: item identifier, distance to the user's location and moods for the item. This interaction is depicted in Figure 14.

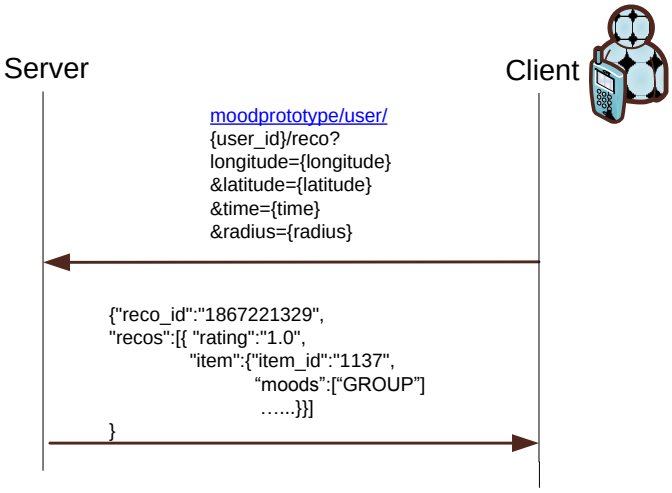

Figure 14 - Client / Server interaction

Chapter 5 Evaluation Results

Service and algorithms validation

As part of the project developed in "*Telefonica I+D"* the user experience was validated in some tests with real users. This test has been made to customers in London by *O2/UK* using the iPhone application developed in the prototype.

The conditions and results of these tests are not published in this work because it is confidential by *O2/UK* Company. User satisfaction data may not be published, but we know the success of the test users' acceptance and the approval of the prototype to enter on deployment phase to obtain a commercial product.

So I can assert and prove the effectiveness of the system and its algorithms in order to provide recommendations based on user context in mood. Thus fulfilling the main objective of this project.

Technical results

Configuration

A full execution test has been performed over the platform described in chapter 4, with the algorithms described in chapter 3.

The execution run time is set to 1 hour, and during that period, the platform processed a large database of "providings" (items just acquired from external content providers), hybridizing them, detecting duplicates and extracting their associated context (associated mood).

The context assign is obtained using the two algorithms presented on chapter 3 Compare ***Keyword assignation*** and ***Reputation-based item clustering.***

The test providings set is composed by a distribution of venues and events in a given geographical area. The initial number of providings in the database is 45,000.

Results

The obtained results are divided in two typologies. On first case we can observe the results of the ***Item processing algorithm*** with the enhancement hybridizer and merging phases.

The processing results are provided in Table 1.

Parameter	Value
Candidates processed in 1 hour	24,337
Identifications of same item in 1 hour	16,219
Identifications of similar item in 1 hour	1,585
Items with content enhanced	8,118
Number of items finally included in the Data Model	6,533

Table 1 - Test evaluation results

As can be observed, the number of items that are finally incorporated into the Data Model is around the 26% of the number of items processed. Given the level of aggregation of information in the final set of items, as well as the additional context information for each one of them, it is clear the added value of the procedure. That added value is based not only on the enhanced information generated throughout the process, but also due to the significantly smaller number of items to handle in a service with a potentially big number of subscribers.

The second case of study is the compare with the two contextualization algorithms presented using some parameters. As we have know in chapter 3 we have a static algorithm using association rules called Keyword assignation and a dynamic algorithm clustering the items depends on user interaction called Reputation-based item clustering.

The processing results are provided in next tables:

	Keyword	Clustering
Number of items on DB	5.000	5.000
Number context tags	17.532	4.781
Ratio context tag for item	3,5	0,95
Number of Item without context tag	752 (15%)	1.853 (36%)
Execution Time	13 minutes	25 minutes

Table 2 - Algorithms results with a very low number of contents.

	Keyword	Clustering
Number of items on DB	15.000	15.000
Number context tags	43.577	31.498
Ratio context tag for item	2,9	2,1
Number of Item without context tag	1.823 (12%)	3.156 (21%)
Execution Time	41 minutes	50 minutes

Table 3 - Algorithms results with a low number of contents.

	Keyword	Clustering
Number of items on DB	45.000	45.000
Number context tags	163.945	188.743
Ratio context tag for item	3,7	4,2
Number of Item without context tag	5.903 (13%)	3.965 (7%)
Execution Time	168 minutes	63 minutes

Table 4 - Algorithms results with a normal number of contents.

As we can be seen in the previous tables of the two algorithms behaviors are different but are not far from expected.

In the case of keyword association algorithm we can see that the ratio remains constant (context tag for item). The algorithm follows statics associative rules. Therefore, increasing items in the database does not affect their behavior. Otherwise the runtime increase is very important, so that would not be scalable.

In the case of the clustering algorithm we observe the phenomenon "cold start" in the items context association. This means that with few items in the database the algorithm is not able to get a good allocation of context tags, but as the items grow the efficiency of the algorithm increases exponentially.

Although the privacy conditions, with the data that we have like a researchers we can draw several conclusions:

- First the need to combine the two algorithms to solve the problem of "cold start"

- Second, the client satisfaction is higher with the use of the clustering algorithm.

- Finally, the system developed is part of a pilot product provided by *O2/UK LastminuteLabs* and you and your current (2011) is about 150 users/day and more than 1500 petitions/day.

Chapter 6 Conclusions and next steps

In this work it is presented a complete system that aims to match the user context with the context of items out of a complete catalogue to be recommended. The items to be recommended are public information items that are provided by the users and uploaded to generic information repositories available in the internet, so a great complexity of the problem is coming from the fact that these contents are user-generated, and therefore not structured or uniform in format etc.

A complete information processing mechanism is described in detail, including the algorithms and procedures to extract the context associated to a given information item. In order to focus the proposal, a use case is selected, that is the tourist recommendations in mobility environments, being the information items the restaurants, bars, cinemas, etc available in different internet content providers.

In that situation, the context of the user is considered to be the mood, as well as the location or time of day. The mood of the user may be acquired through different procedures (explicitly or implicitly), but the mood associated to a given item shall be extracted as described in this work.

A full implementation prototype is developed and presented, along with experimental results, to verify the performance and general behavior of the algorithmic solution.

Major findings

As main conclusions we can observe are:

- The Contextual Post-filtering architecture is an efficient way of contextualization of content.

- The results are rated positively by users without need to create a complex contextual modeling system.

- The need to improve the content for: duplication detection and enhancement content has proved useful. This has delivered a best content and it makes easier the inference of context labels.

- It was also concluded that we need a combination of two algorithms for assigning context presented to solve different problems:

    - The keywords based algorithm for static allocation using association rules get to avoid the effect of "cold-start." This means that it is capable of assigning context tags faster than other when the content is still very new.

- The Reputation-based clustering algorithm gives better results tailored to the users and the ability to learn and adapt to dynamic changes by users. We have shown that this algorithm has a processing time of the initial content higher than the last.

- This combination of initial velocity by the first and the quality, learning and the dynamism brought by the second becomes an optimal solution with proven success.

Environmental impact

This document defines a software project without industrial production needs. Only the implementation of the *Telefonica I+D* prototype production (where the algorithms presented on this project are included.) have production needs with environmental impact.

The environmental impact of the prototype implementation and service production *Telefonica* and *O2/UK* is confidential information and are not objective in this study.

Although we can arrive to the conclusion that the environmental impact of the productive needs to implement the service is much lower than the efficiency which allows the service to the users in their mobility management with the consequent reduction of environmental impact

Future Work

Next steps of this research are the following:

- Extend the types of information items that can be processed through this system.

- Extend the context dimensions, apart from user mood, that can be handled by the system, in order to be able to enhance the contextual model for both the user and the items.

- Improve the clustering algorithm performance and research new solutions to apply on the contextual post-filtering.

- Perform tests with real users in order to identify potential drawbacks of the system.

Chapter 7 References

[1] Stabb, S.; Werther, H.; Ricci, F.; Zipf, A.; Gretzel, U.; Fesenmaier, D.R.; Paris, C.; Knoblock, C.; , "Intelligent systems for tourism," Intelligent Systems, IEEE , vol.17, no.6, pp. 53- 66, Nov/Dec 2002.

[2] L. Capra, W. Emmerich, C. Mascolo, “CARISMA: context-aware reflective middleware system for mobile applications,” Software Engineering, IEEE Transactions on, vol.29, no.10, pp. 929-945, Oct. 2003.

[3] C. Baladron, J.M. Aguiar, B. Carro, A. Sanchez, “Integrating User-Generated Content and Pervasive Communications” Pervasive Computing, IEEE, vol.7, no.4, pp.58-61, Oct.-Dec. 2008.

[4] van Sinderen, M.J.; van Halteren, A.T.; Wegdam, M.; Meeuwissen, H.B.; Eertink, E.H.; , "Supporting context-aware mobile applications: an infrastructure approach," Communications Magazine, IEEE , vol.44, no.9, pp.96-104, Sept. 2006

[5] Ja-Hwung Su; Hsin-Ho Yeh; Yu, P.S.; Tseng, V.S.; , "Music Recommendation Using Content and Context Information Mining," Intelligent Systems, IEEE , vol.25, no.1, pp.16-26, Jan.-Feb. 2010

[6] Micheli, A.; Sona, D.; Sperduti, A.; , "Contextual processing of structured data by recursive cascade correlation," Neural Networks, IEEE Transactions on , vol.15, no.6, pp.1396-1410, Nov. 2004

[7] Bhamidipati, N.L.; Pal, S.K.; , "Stemming via Distribution-Based Word Segregation for Classification and Retrieval," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on , vol.37, no.2, pp.350-360, April 2007

[8] Perrone, M. P.; Russell, G. F.; Ziq, A.; , "Machine learning in a multimedia document retrieval framework," IBM Systems Journal , vol.41, no.3, pp.494-503, 2002

[9] Yanjun Li; Congnan Luo; Chung, S.M.; , "Text Clustering with Feature Selection by Using Statistical Data," Knowledge and Data Engineering, IEEE Transactions on , vol.20, no.5, pp.641-652, May 2008

[10] Yu-Gang Jiang; Jun Yang; Chong-Wah Ngo; Hauptmann, A.G.; , "Representations of Keypoint-Based Semantic Concept Detection: A Comprehensive Study," Multimedia, IEEE Transactions on , vol.12, no.1, pp.42-53, Jan. 2010

[11] Man Lan; Chew Lim Tan; Jian Su; Yue Lu; , "Supervised and Traditional Term Weighting Methods for Automatic Text Categorization," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.31, no.4, pp.721-735, April 2009`

[12] Prahalad C.K. (2004): Beyond CRM: C.K. Prahalad Predicts Customer Context is the next Big Thing. American Management Association McWorld.

[13] Palmisano, C., Tuzhilin, A. and Gorgoglione, M. (2008) Using Context to Improve Predictive Models of Customers in Personalization Applications."" IEEE Transactions on Knowledge and Data Engineering, 20(11), pp. 1535 - 1549.

[14] Panniello, U., A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. 2009. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In Proceedings of the ACM Recommender Systems Conference (RecSys 2009), pp. 265-268.

[15] Dourish, P. 2004. What we talk about when we talk about context. Persistent and UbiquitousComputing, p. 19 - 30.

[16] Adomavicius, G., R. Sankaranarayanan, S. Sen, and A. Tuzhilin (2005) Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach. ACM Transactions on Information Systems, 23(1):103-145, 2005.

[17] Adomavicius, G. and A. Tuzhilin (2005). Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions, IEEE Transactions on Knowledge and Data Engineering, 17(6).

[18] Adomavicius, G. and A. Tuzhilin (2005). Incorporating Context into Recommender Systems Using Multidimensional Rating Estimation Methods. In Proceedings of the 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI 2005), Reading, United Kingdom, October 2005.

[19] Adomavicius, G., A. Tuzhilin, and R. Zheng (2010). REQUEST: A Query Language for Customizing Recommendations. Information Systems Research. Forthcoming.

[20] Palmisano, C., Tuzhilin, A. and Gorgoglione, M. (2008) Using Context to Improve Predictive Models of Customers in Personalization Applications."" IEEE Transactions on Knowledge and Data Engineering, 20(11), pp. 1535 — 1549.

[21] Panniello, U., A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. 2009. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In Proceedings of the ACM Recommender Systems Conference (RecSys 2009), pp. 265-268.

[22] Anand, S.S. and Mobasher, B., 2007. Contextual recommendation. WebMine, LNAI 4737, B. Berendt et al. (eds.), pp. 142 — 160.

[23] Chaudhuri, S. and Dayal, U. 1997. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26(1):65-74.

[24] Bazire M. and Brézillon P. (2005). Understanding Context Before Using It. In A. Dey et al., editor, 5th International Conference on Modeling and Using Context, CONTEXT 2005, Springer-Verlag.

[25] Koller, D. and Sahami, M. 1996. Toward Optimal Feature Selection. In Proceedings of the 13th International Conference on Machine Learning, Morgan Kaufmann.

[26] Liu, H. and Motoda, H. 1998. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers.

[27] Chatterjee, S., Hadi, A. S., and Price, B. 2000. Regression Analysis by Example. John Wiley & Sons, Inc.