Journal of Mobile Multimedia, Vol. 9, No.3&4 (2014) 189-213

IMPROVEMENT QUALITY OF THE RECOMMENDATION SYSTEM USING THE INTRINSIC CONTEXT

LATIFA BABA-HAMED

RIIR Laboratory, University of Oran, Algeria
lbabahamed@yahoo.fr

REDA SOLTANI

RIIR Laboratory, University of Oran, Algeria
reda.soltani@hotmail.fr

The traditional recommendation systems provide a solution to the problem of information overload. They provide users with the information and content which are the most relevant for them. These systems ignore the fact that users interact with systems in a particular context. Context plays an important role in determining users' behavior by providing additional information that can be exploited in building predictive models. Context-aware recommendation systems take this information into account to make predictions in order to improve the performance of the filtering process. Most existing Context-aware systems use the extrinsic context. In this paper, we propose an intrinsic contextual recommendation system that we can apply to the recommendation of contents in general (i.e. book, Url, item, product, movie, song, restaurant, etc.). The context in our approach is extracted from the set of attributes for the object itself. Our system use a contextual pre-filtering technique based on implicit user feedback. To show the performance of the recommendation process, we consider the movie domain as a case study.

Key words: Recommendation, context, user profile, preference, matching operator, precision.

1 Introduction

Recommender systems are powerful tools helping on-line users to leverage information overload by providing personalized and relevant recommendations. Relevance is measured by the similarity between content and user profile which consists of a set of preferences. A more complete definition of the user profile is presented in [17].

RSs operate according to three filtering strategies, namely, content-based filtering CBF (which consists of matching between a user profile and content descriptors, in order to recommend appropriate products) [18, 25], collaborative filtering CF (which assumes that users who had common interests in the past, will continue, probably, to share the same tastes in the future) [12, 13, 20], or hybrid filtering (which is a combination of these two approaches) [22].

Rating prediction in recommender systems relies primarily on the information of *how*, *who* rated *what*. There are many methods that take additional data about the *who* (personal and professional

information about the user) or the *what* (item attributes like movie genres or the product description) into account.

Besides such data about the entities involved in the rating events, there is possibly also information about the situation in which the rating event happens, e.g. the current location, the time, the temperature, or the current mood of the user. Such situational information is usually called context. One of the most frequently cited definitions of a context was proposed by Abowd, Dey et al [2]: « 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 ».

Context can be introduced at different levels of recommendation process. In [4] (and also shown in Figure 1), three different algorithmic paradigms for incorporating contextual information into the recommendation process are presented:

1. **Contextual pre-filtering (PreF):** contextual information is used to filter out irrelevant ratings before they are used for computing recommendations with standard (2D) methods.

2. **Contextual post-filtering (PoF):** contextual information is used after the classical (2D) recommendation methods are applied to the standard recommendation data.

3. **Contextual modeling (CM):** contextual information is used inside the recommendation-generating algorithms.

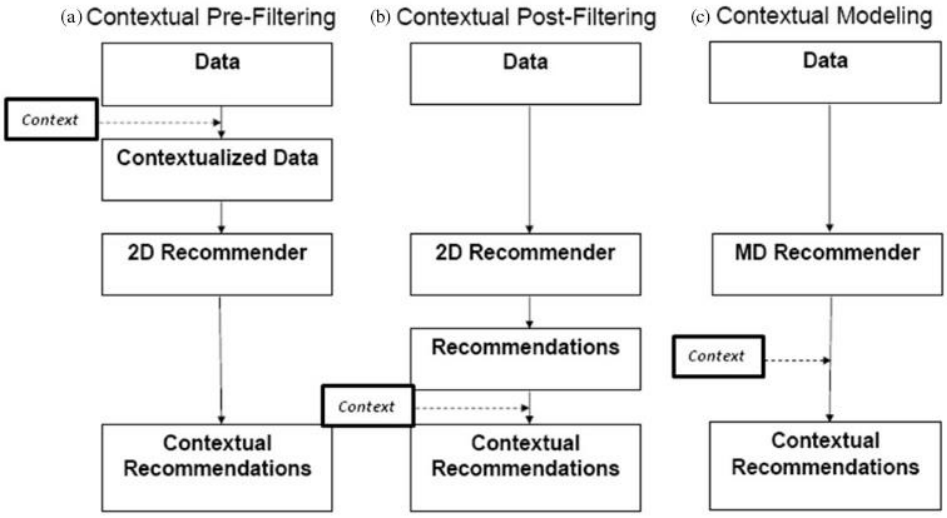

Figure 1 How to use context in the recommendation process.

Panniello et al. [21] compare the pre-filtering and the post-filtering approaches and identify which method dominates the other and under which circumstances.

The use of contextual information in the field of recommendation systems is very topical. Several researches have emerged, but very few of them were interested in the intrinsic context (i.e. context that is part of the object itself).

In this paper, we present a new approach for the consideration of intrinsic contextual information and its impact on the recommendation process. We can apply this approach for the recommendation of contents in overall (book, Url, article, product, movie, song, restaurant, etc.). For the illustration and the evaluation, we chose the movie domain. The rest of the paper is organized as follows. In Section 2, we recall the data model and the formalism followed. Section 3 is devoted to the architecture of the contextual system and its main components. An instantiation of our approach in the case of movies recommendation is given in Section 4. In Section 5, we present the results of the evaluation of the developed system. In Section 6, we present the main context-aware recommender systems and we compare them on the basis of a certain number of criteria. Finally, Section 7 concludes the paper and gives some perspectives

2 Data Model

In this section, we give all the basic concepts definitions that we manipulate in this paper.

2.1 Preference

A preference is a formula for prioritizing a set of objects related to the interests and needs of a user. There are two types of preferences: qualitative and quantitative. Quantitative preferences are expressed with binary ordering relations (eg. >) and used to define a partial order on objects. Quantitative preferences are expressed through functions scores that assign scores to different objects. This type of preferences allows defining a total order between objects. In this article, we shall focus on quantitative preferences that we modeled as follows:

Preference *pi* is a couple *pi(pri, wi)* where *pri* is a predicate <*attribute, operator, value*> and *wi* is a real number between 0 and 1 that represents the degree of interest of the user in relation to the predicate *pri*. 0 reflects the minimum preference, 1 reflects the maximum preference.

Example: Let R a relation schema modeling cars, R (mark, model, price, color, mileage, power, nb Horses, nbSeats). A user can define the following preferences: p1 (<color,=,'Red'>, 0.9) and p2 (<Price,'> '1.4M-AD>, 0.2).

2.2 Context

In this article, we define the context of the point of the objects and not the users. Context is any information which affects the interest that a user can express on an object. We model the context as a conjunction of predicates defined on contextual attributes. We distinguish two types of contexts: intrinsic and extrinsic context. The first type is defined on the set of attributes for the object itself. For instance, in the relation R, the *mileage* and *type* attributes can be considered as contextual attributes.

Extrinsic context is a set of attributes relating to the environment of user interaction. It can provide information on the geographic location of the user (IP address), the time period of its interaction (date and time), the media used for interaction, etc. For example, from the date, we can detect an event that can change the preferences of the user momentarily. Thus, an Algerian user may recommend historical

and revolutionary objects (books, songs, movies, documentaries, etc.) in the periods surrounding a historical event as 1st November or 5th July.

2.3 Contextual preference

To capture the variability of the user preferences in different contexts, we have introduced the concept of contextual preference. Contextual preference is a valid preference in a particular context. It must be taken into account if the user is in this context. We use the following formalism to model the contextual preference: *pc: pi* | context = *cj*, where the preference *pi* is defined in the context *cj*.

Example: on the relation R of cars, a user can set both the following contextual preferences: *cp1* [(<Price,'> '1.4M-AD>, 0.2) | Type =' Renault '] *cp2* [(<Price,' <', 1.7M-AD>, 0.8) | Type = 'BMW']. In *cp1*, the user expresses the fact that he gives a score of 0.2 in the tuples of the relation R (car) for which the price attribute takes a value greater than 1.4 million Algerian Dinars (AD) in the context of type car 'Renault' (intrinsic context). At the same time, the user gives a score of 0.8 for cars with a price of up to 1.7 million AD in the context of type car equal to BMW.

2.4 User profile

In our approach, the user profile is defined by the set of contextual preferences that the system automatically learns by analyzing the interaction logs. *Pu* = {*cp1, ..., cpn*}. This model captures the fact that a user may have different preference scores on the same object based on its context. For example, a user can give a very high score to the genre 'war movies' when the context is related to a historical event and at the same time assign a very low score on the same genre in other contexts of the year.

2.5 Description of content

Beyond of tuples of a database, we are interested in the recommendation contents in general (Book, Url, item, product, movie, song, etc.). In this article, we consider that all objects are modeled by conjunctions of predicates. *Dc* = {pr_1, ... pr_n}. Example: *Dc* (book1) = {<title,=, Le Chao des Sens>, < author,=, AhlamMosteghanemi>, < editor =France Meyer>, ...}

3 Recommendation Process

In this section, we first present the general architecture of the system. Then we detail the process of creating user profiles. Finally, we show how to solve the context.

3.1 The system architecture

Figure 2 shows the different components of our recommendation system.

When a user connects to our system, his profile is immediately identified as well a subset of potentially recommendable contents. In the first step, the context resolution module identifies relevant intrinsic and extrinsic contexts for the user who is logged on. The second step (matching) is to measure the similarity between the user profile and all the contents descriptors potentially recommended. For this, we used the global similarity, denoted $Sim_{globale}$, which we presented in [8] and which we recall

here in a Formula 1. This measure combines two similarities: the Pearson numeric similarity (denoted by sim_{pref} in Formula 1) and using the semantic metric of Jiang & Conrath [16] (denoted by sim_{sem} in the Formula 1) to improve the relevance of the results of the recommended products to the users. The matching module returns an ordered list of relevant contents for user.

$$Sim_{globale}(Pu, C) = \alpha \times sim_{sém}(Pu, C) + \beta \times sim_{préf}(\overrightarrow{Pu}, \overrightarrow{C}) \quad (1)$$

where ($\alpha + \beta = 1$).

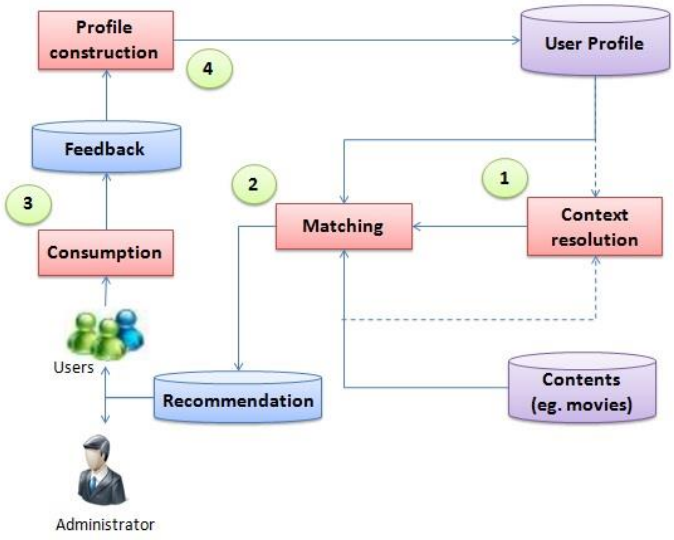

Figure 2 Recommendation System Architecture.

In the third step (consumption), the user browses the list of recommendations and noted the contents that are offered. The scores given by the user are stored in logs and will be considered as the user feedback. The last step (construction and updating profile) which can also be seen as the first step of our process is to analyze the logs of the user to find his contextual preferences and create his profile. It should be noted that the user profile is constantly updated by this module at the same time as the user's preferences change.

3.2 Profiles user creating

We propose in this paper a new approach for the automatic creation of profiles based on the analysis of interaction logs (also call log files or feedback). We recall that a user profile is a set of contextual preferences. We will explain in the following section, how to create profiles from interaction logs considering the intrinsic contexts.

The interaction logs that we use have a common structure in several commercial recommendation systems (eg MovieLens, LastFM, Pandora, etc.). What we are interested in these logs are the ratings given by users as: Rating <$content_l$, $user_k$, $score_{lk}$>, such as the user (k) gives a preference score ($score_{lk}$) to the content (l). In order to extract contextual information from the logs preferences, we transform them by expansion of vote and propagation of scores based on the contents descriptors. We

get, after processing, tuples with the following form: <$user_k$, $context_i$, $predicate_j$, $score_{lk}$> where context_i is a context predicate (intrinsic context) describing the content_l, pedicate_j is a no contextual predicate of the content_l and score_lk is none other than the score expressed by user_k on content_l.

Algorithm 1 : Profiles creating

Inputs :
L : log file
Output :
Pu : User Profile
Begin
C := Instances_Contexts(L)
T := Predicates (L)
P_u := {}
For each c_i in C do
For each t_j in T do
s_{ij} := Score (t_j, c_i, L)
P_u=P_u U {(t_j, s_{ij}| c_i)}
End For
End For
returns P_u.
- **Instances_Contexts (L):** returns the set of all instances of the intrinsic context which are present in the log files L.
-**Predicates (L):** returns the set of all non contextual predicates in L and which are relevant for the user profile.
-**Score (t, c, L):** returns the average of the assigned scores to the predicate t in the context c.
End

Algorithm 1 describes the process of creating profiles from processed logs. The main idea of this algorithm is to calculate the score for each context predicate in each context instance. This will allow our system to capture the preferences variability in different contexts. In other words, this process is used to model the fact that a user has different levels of preferences on the same object as shown in the following example: Samira Profile (<kind_song=sentimental,0.9|singer=Céline Dion>, <kind_song= sentimental,0.1 |singer=Lady Gaga>) where Samira gives a high score to sentimental songs when they are performed by Celine Dion and a very low score on the same kind of song when they are performed by Lady Gaga.

3.3 Context resolution

The second important component of our SR is the context resolution. In the general case, this component takes as input the candidates content descriptors for recommendation, the user profile and the information on the context of the user (eg IP address, date and time of connection, used media, etc.) and returns the intrinsic and extrinsic contexts to consider in the calculation of recommendations to the user. We describe in what follows the resolution process of intrinsic contexts (for the extrinsic context resolution, you can see [1]).

In its classical schema, the resolution context process is extracting the intrinsic context directly from the content identified as potential candidates for recommendation. That is to return all the contextual predicates describing the contents candidates. But in some cases, contexts extracted by this process are not compatible with the user's profile. In the case of Samira's profile for example, we can distinguish two contexts (singer = Celine Dion) and (singer = Lady Gaga). Assuming a sentimental song by Shakira is identified as a candidate for the recommendation, then the context resolution process will return the context (singer = Shakira). However, the recommendation system will not know what Samira's preference take into account to evaluate the relevance of this song (0.9 or 0.1?).

Algorithm 2: Context resolution

Inputs:
C :=Instances_Contexts(ensembleCandidats) // context instances included in the candidate contents to the recommendation.
Cu := Instances_Contexts(Pu) //contexts instances included in the profile Pu
Output :
Cc : set of pairs(ci,xi) where ci is a courant context. xi is a correction factor of user preferences, xi in [0,1].
Begin
Cc := {}
For each ci in C **do**
If ci in Cu **then**
Cc:= Cc U {(ci,1)} // none correction
Else
c* := {c*|sim(c*,ci)=Max(sim(c*,cj)),
j=1..||Cu||}
Cc:= Cc U {(c*,sim(c*,ci))}
End If
End For
returns Cc.
-sim(c1,c2) : returns the similarity between the two instances of context c1 and c2.
End

To solve this problem, we propose an approach based on the similarity between contexts. The intuition behind our approach is that users like songs of similar singers. If we have the information that Sharika is most similar (sings the same style) to Lady Gaga than it is with Celine Dion, then it would be more appropriate to consider the preferences of Samira on Lady Gaga (<song_kind= sentimental,0.1 |singer=Lady Gaga>) to evaluate the songs of Shakira. Since Shakira is not Lady Gaga, then we also define a preference correction factor which is given by the similarity between Shakira and Lady Gaga. More details on the use of the correction factor will be given in the next section.

Algorithm 2 describes the process of contexts resolution. The algorithm takes as input the set of context instances given by the candidates' contents and contexts contained in the user profile; it returns a set of pairs (context, correction factor) which are compatible with the user profile.

3.4 Matching profile / content

The most important process in a recommendation system is to evaluate the relevance of content to a particular user. Some RS decide that content is relevant to a user if the users who are like him have consumed this content and have considered it as relevant to them. This is called RS-based on collaborative filtering (CF). Other RS predicts the relevance of content to a user by calculating the similarity (correlation) between the user profile and the content descriptor. This is called RS-based on content-based filtering (CBF). In what follows, we give a description of our matching process of profiles and context in CBF RS.

As shown in Figure 2, the matching process takes as input the user profile, a set of candidates' contents and all current contexts. It returns an ordered list of recommendations. To generate this list, the matching process calculates the similarity between the user profile and each candidate content descriptor. These are then arranged in decreasing order of their relevance to the user (similarity profile). The K first contents are then recommended to the user.

The matching between a content descriptor and a user profile in a given context, Sim (Pu, DescCt, Cxt), consists of two steps:

1 - Filtering and correction of relevant preferences. This step is to choose among all the user preferences, those that are relevant to the content (DescCt) in context (Ctx). These preferences are then corrected by multiplying them by the corresponding correction factor to the context (Ctx). The result of this step is a weighted (preferences) predicates vector (From DescCt ∩ Pu).

2 - Evaluation of the content relevance. This step is to measure the similarity between the weighted vector of the first step and the vector of the content descriptor. We proposed for this, to use two known measures that are *cosine* and the *weighted average* (see [8] for more details on the implementation of these measures).

4 Instantiation of the algorithms for the movies recommendation

In this section, we show through concrete example how we have instantiated our approach in the field of recommendation movies. We recovered the logs from the MovieLens website. In these logs, we have 100k tuples representing user votes on films. The votes are in the following form (Id_user, Id_film, score), where score is a natural number between 1 (low preference) to 5 (high preference).

4.1 Creating film profiles

In order to create profiles which take into account the context we used data coming from IMDB for the transformation of logs. The movie descriptor contains the following information: title, genre, actors, and directors. We consider that the director attribute is a contextual attribute (intrinsic context) and the "genre" attribute is the only attribute concerned by the preferences. It is should be noted that a film can have several genres. After transformation of logs, we get a table that looks like Table 1.

From this table, we calculate the user's preference (average score of ratings) for each genre of film in the context of each director for which the user has seen at least one film.

Table 2 shows a profile fragment of the user number 3. The preference score of the film genre "Thriller" is 3.5. This score is obtained, using Table 1, by calculating the average of the score of the film genre Thriller number 6 which is equal to 5 and the score of the film genre Thriller number 3006 which is equal to 2. Thus we have created profiles for all users present in our database.

Table 1 Extended and enriched Log.

User ID	Movie ID	Director	Genre	Ratings
...	...	...	...	...
3	6	Mann Michael	Action	5
3	6	Mann Michael	Crime	5
3	6	Mann Michael	Drama	5
3	6	Mann Michael	Thriller	5
3	3006	Mann Michael	Thriller	2
3	3006	Mann Michael	History	2
3	3006	Mann Michael	Drama	2
3	3006	Mann Michael	Biography	2
...	...	...	...	...

Table 2 Preference example

User ID	Director	Genre	Average
...	...	...	...
3	Mann Michael	action	5
3	Mann Michael	Thriller	3.5
3	Hyams Peter	Action	4
...	...	...	...

.

4.2 Similarity between directors (intrinsic context)

As we explained, in Subsection 3.3, when a new context is presented to the user (typically a director for which the user has not seen a movie before). In this case, the system must predict the preferences that would have had the user with respect to the movies genres produced by this unknown director. For this, we need to find among the known directors by the user (those in the profile), the one that is most similar to the unknown director.

We represent each director (intrinsic context) by a vector of weighted genres, Director (<Action, 0.7> … <Drama, 0.2>). The weight (score between 0 and 1) associated with each genre shows the percentage of films of this genre made by the director concerned (number of films of a certain genre / total number of films). For example: <Action, 0.7> shows that 70% of films made by the director, are Action movies. Table 3 gives some of the vectors representing the directors.

Table 3 Directors vectors.

Director \ Genre	**Action**	**Adventure**	**…….**	**Drama**
Johnston Joe	0,7	0,5	…….	0,2
Mann Michael	0,3	0,4	…….	0,7
……..	……	……	…….	……

After computing the vectors of all the directors, we can proceed to fill the similarity matrix directors / directors. This latter is a symmetric square matrix, containing all directors of the base (see Table 4). We use the Pearson correlation to calculate the similarity between two directors. This measure is based on the calculation of the correlation between two vectors $X\ (x_1, ..., x_n)$ and $Y\ (y_1, ..., y_n)$ by the Formula (2):

$$\text{Sim (x,y)} = r_p = \frac{\sum_{i=1}^{N}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (2)$$

In Table 4, we note that the similarity Sim (Ri, Ri) = 1 (because the director is similar to itself). Sim (Ri, Rj) = 0.7 indicates that the director Ri is similar to director Rj with a value of 0.7

Table 4 Similarity matrix between directors.

	R_1	R_2	…	R_l	…	R_i	…	R_n
R_1	**1**							
R_2		**1**						
…			**1**					
R_i				**1**		**0,7**		
…					**1**			
R_j						**1**		
…							**1**	
R_n								**1**

4.3 Calculation of recommendations

We will give in this subsection a concrete example of calculating recommendation on the films realized by an unknown director for our users. Take, for example, a user U1 with the profile shown in Table 5.

Table 5 Profile of U1.

User	Director	Genre	Average
U1	R1	Action	3,5
U1	R1	Adventure	4
U1	R1	Drama	2
U1	R2	Action	2,5
U1	R2	Adventure	3
U1	R2	Drama	4

We find that user U1 has only seen films of the two directors R1 and R2. Now imagine, the insertion of a new film in the database. This film is described as follows: Film (Title = Titanic, Genre = {Drama, Adventure, Action}, Director = R3).

To estimate the interest of the movie "Titanic" for the user U1, the context *Director = R3* must be resolved. This is equivalent to find which among R1 and R2 is most similar to R3; for this we use the matrix of similarities given in the Table 6. It should be noted that this matrix contains 427 directors who realized a total of 4141 films.

Table 6 Global matrix of similarities

	R1	**R2**	**R3**	**…..**
R1	1	0,4	0,7	…..
R2		1	0,6	…..
R3			1	…..
…..		…..	…..	…..

.

From Table 6, we deduce easily that the director R3 is more similar to director R1 than R2. Indeed, Sim (R1, R3) = 0.7> Sim (R2, R3) = 0.6. Therefore, Profile of user U1 will be enriched with contextual preferences predicted and corrected from those relating to director R1 (see Figure 3).

Director	**Action**	**Adventure**	**…….**	**Drama**
R3	3,5*0,7	4*0,7	…….	2*0,7

⇧ ⇧ ⇧

Director	**Action**	**Adventure**	**…….**	**Drama**
R1	3,5	4	**……..**	2

Figure 3 Enrichment of the user profile.

Finally, the interest score of "Titanic" relative to U1 is calculated as follows: Interest (Titanic, U1) = 1/3x (3.5 * 0.7 +4 * 0.7 +2 * 0.7) = 2.21. Comparing this interest score to a predefined score, the system can decide if the Titanic film is recommended or not to the user U1.

5 Recommendation process evaluation

In this section we show how we proceeded to evaluate our system in terms of precision, taking into account the context. We compare, also, the results of this evaluation to those of the evaluation system without context to show the impact of the context on the recommendation process.

The evaluation system was based on the test platform made during the project APMD (Personalized Access to Masses of data) [6, 23] from which a benchmark consisting of 21 users who have watched the same 100 film was extracted. In this benchmark, we extracted one part (training set) for learning user profiles. The other part (result set) was used to test the recommendation process.

This evaluation involved calculating vectors of directors, the creation of the director's matrix and six tables whose schemas are as follows:

T-usernote (userId, movieId, note)

Training-set (userId, movieId, genre, note)

Profil-user (userId, genre, préférence1)

Result-set (userId, movieId, note, prédiction)

Contextual-Profil-user (userId, director, genre, préférence2)

Directors (movieId, director)

Where userId is the user identifier, MovieID represents the identifier of the film, note is the rating assigned by the user to a movie he watched, genre represents the kind of film (the genres considered are those presented in [7]), préférence1 represents the preference (regardless of context) that can have a user for a given film genre, prediction is the value of the global similarity calculated by the matching operator, director represents the film's director, and préférence2 represents the preference which the user can have for a given a genre of movie for a given director (intrinsic context).

The table T_usernote contains 21 users who have watched the same 100 films. From the table T_usernote, we filled the tables Training_set and Result_set randomly using a random procedure that we have implemented. This procedure will give the following distributions: {(70.30) (50.50) (30.70), etc.}. For example, for the first couple (70.30), the first component is the 70 films that serve to fill the table Training_set (corresponding to the portion Training set) and the second is the remaining 30 among the initial 100 films, used to fill the table Result_set (corresponding to the test part).

The table Training_set, for this example contains 21 users, 70 movies for each user, the genres associated with each film and the note of the film propagated on its genres. The table Result-set for the distribution (70, 30) contain 21 users, 30 movies for each user, the notes associated with each film and

the prediction column used for calculating the relevance of each film for each user. This utility will be given by the similarity between the user profile and the descriptor of each film. This column is initially empty.

The user-profile table serves to generate a user profile from the Training-set table. It contains 21 users, the genres of films derived from the Training-set table, and the preference value (preference 1) of a genre (for a given user) which is obtained by calculating the average of the scores of this genre in the Training-set table. An example of calculation of user preferences, regardless of intrinsic context, is given in [8].

The table Contextual-Profile-user can generate a profile of a user taking into account the intrinsic context director, from the join between the tables Directors and Training-set. It contains 21 users, the genres of films derived from the join of tables Training-set and Directors, and the preference value (preference 2) of a genre for a given user according to the director. This value is obtained by calculating the average of the scores of this genre by considering the same director.

The table Directors is used to know the director of each film. It helps in building the contextual profile of a user, in the creation of vectors of directors, and in the preparation of the matrix of directors. To evaluate the accuracy of our recommendation system, we set up the scenario of extraction of the TopK films most interesting to the user. To calculate the precision we proceed as follows:

For a partition (Training set / Result set) and a given user, we proceed in a first step, to the prediction of interest score which would have given, by the user, to each film belonging to the result-set. These scores are calculated by the matching operator of our recommendation system. At the end of this step, the films belonging to the Result-Set are classified according to their predicted scores. The second step is to extract the K first films (those with the highest scores) and to put them in the set "TopKpredicted". In a third step, the films of the entire Result-Set are ordered in descending order according to the true ratings given explicitly by the user (MovieLens data [19] integrated into the platform APMD). The K films, which received the highest notes by our user, are placed in the set "TopKreal". The last step is to calculate the precision according to Formula (3).

$$precision = \frac{TopKpredicted \cap TopKreal}{K} \quad (3)$$

This process is repeated for each user. The average of this calculation for all users, delivers the precision value for the value K. Varying the number K allows us to obtain further precision values that will enable us to trace the precision curve.

We made several experiments by making changes on the coefficients of the global similarity measure and on the value of K for a fixed size of the table Training-set. The values of the coefficients used are: $\alpha = 0.75$ and $\beta = 0.25$ and the considered values of K are: 5, 10, 15, and 20 films. The figure 4 shows the curves of precision with and without consideration of intrinsic context for the distribution (70, 30).

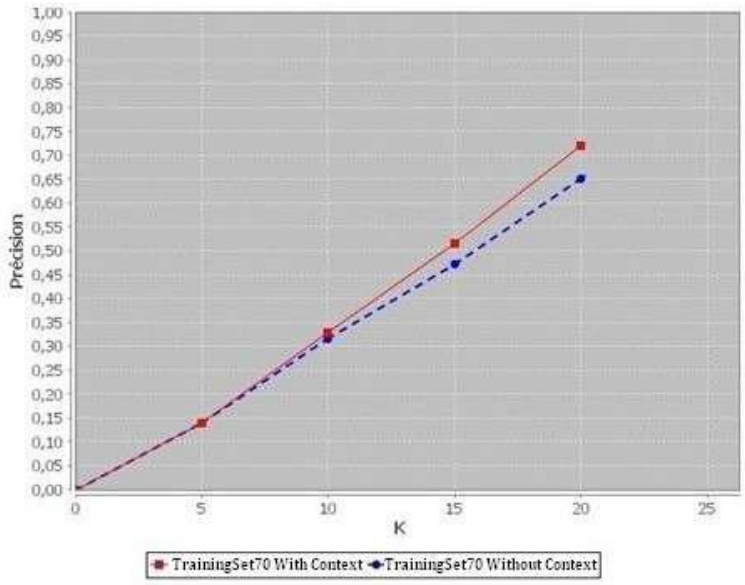

Figure 4 Precision by the method of TOP k (Global Similarity With and Without Context and size of Traininge-Set = 70).

We note, in Figure 4, that when the value of K increases the precision curve, with context, increases. The precision curve with context is over the precision curve without context. We can conclude that the use of the context has a positive impact on the recommendation process.

6 Related works

Several context-aware systems have emerged recently. In this section we start by the presentation of the main systems, and compare them using different criteria (see Table 8).

6.1 Main context-aware recommender systems

The micro-profiling approach [9] aims to recommend unknown songs / artists to a user. The recommendation system type is the pre-filtering because in this approach the data are filtered by the context (the time: time of day, day, month or year) before making any recommendations.

In this approach, we assume that the users' tastes change with a period and may be similar in the same period. For example, a user prefers to listen to a certain kind of song while working and another kind of song before sleeping.

We have a user profile in which the songs / artists listened to by the user are stored. This profile will be broken down so that it adapts to each context: in this approach, time is the time of day, it will be divided into several segments according to the songs that user listens to the most during each time (time segment) of the day.

The problem with this approach is how to divide the hours of the day? That is to say how much divide time into segments, because the definition of the morning can change from one user to another. So the error of partitioning time should be minimal.

The contexts are implicitly extracted, which makes quite a difficult task because we do not know what the user dislikes. We can not deduct what he likes from what he heard as albums by artists or songs and when he listened to in the day.

To estimate the error (is off-line), the generated preferences predictions are compared to user preferences explicitly obtained.

The different steps of the method are:

• Extraction of implicit data (preferences) and divided into several contextualized segments.
• Transform each segment into a matrix A (user x item) where A (u, i) corresponds to the explicit preferences.

The use of these micro-profiles improves the accuracy of the recommendation. One of the problems with this method lies in the fact that the partitioning of time differs from one user to another (rather than being fixed for all users), which may decrease the accuracy of the system.

DaVI [14] can be applied on the item-based collaborative filtering algorithm, it treats the contextual attributes as new items (virtual items) in the data set. This means that it adds a new row and column for each different value of the context to the former similarity matrix and calculates the corresponding similarity values, among the values of the context and the other items. For example the attribute day Di = {1, .., 31} can be considered as a virtual and contextual object, in computing recommendation (see Table 7).

If we look at the context date and more precisely the context days of the week, then we could recommend for an individual who has watched an action movie on Tuesday, the same kind of film every Tuesday of weeks (deducing that this individual enjoys watching action movies every Tuesday). This will improve the quality of recommended items.

Table 7. Similarity matrix with the day as context attribute

	$\mathbf{i_1}$	$\mathbf{i_2}$	…….	$\mathbf{i_k}$	$\mathbf{d_1}$	…..	$\mathbf{d_n}$
$\mathbf{I_1}$	1	$Sim(i_1, i_2)$	…….	$Sim(i_1, i_k)$	$Sim(i_1, d_1)$	…..	$Sim(i_1, d_n)$
$\mathbf{I_2}$	$Sim(i_2, i_1)$	1	…….	$Sim(i_2, i_k)$	$Sim(i_2, d_1)$	…..	$Sim(i_2, d_n)$
…….	…….	…….	1	…….	……	…..	…..
$\mathbf{I_k}$	$Sim(i_k, i_1)$	$Sim(i_k, i_2)$	…….	1	$Sim(i_k, d_1)$	…..	$Sim(i_k, d_n)$
$\mathbf{D_1}$	$Sim(d_1, i_1)$	$Sim(d_1, i_2)$	……	$Sim(d_1, i_k)$	1	…..	$Sim(d_1, d_n)$
……..	……..	……	……..	…….	….	1	….
$\mathbf{D_n}$	$Sim(d_n, i_1)$	$Sim(d_n, i_2)$	……	$Sim(d_n, i_k)$	$Sim(d_n, d_1)$	…..	1

News @ hand [11] is a news recommender system that uses semantic technologies to provide several types of recommendations. A typical news recommendation page in News@hand is classified into eight different sections: headlines, world, business, technology, science, health, sports, and entertainment. When the user is not logged in the system, he can browse any of the previous sections, but the items are listed without any personalized criterion. When the user is logged in, recommendation and user profile editing are enabled, and the user can browse the news according to his and others' preferences in different ways.

The user preferences are represented as a vector $u_m=(u_{m,1},\ldots, u_{m,k})$, where $u_{m,k} \in [-1,1]$ is the weight that measures the intensity of interest that brings the user u_m to the Ck concept. If the weight is close to 1, then the user like the concept and if it is close to -1, the user does not like. Items are represented as vectors $i_n = (i_{n,1},\ldots, i_{n,k})$ with $i_{n,k}$, is the weight of the concept $\in [0,1]$, which measures the importance of a concept in the content of an item. Interest in items can be calculated by comparing the user's preferences with the items vectors.

MyMap [13] is a mobile recommendation system (that is to say, a recommendation system that can be installed on a mobile device such as a PDA, iPhone, etc.) which is based on the idea of using a map to provide customized recommendations of places of interest for tourists.

MyMap decides which information must be provided and how to present it, from an XML representation of domain knowledge. For this purpose, the system uses two components:
- *Mobile User Profile Manager (MUP)* uses formalized profiles (see Figure 5) according to the UbisWorld language is an interesting approach to model a part of the real world as a city and adapt dynamically to access sources information (see http://www.w2m.org).
- *Selection and presentation module of information:* has like task the generating of items descriptions selected, after an explicit user request or proactively in the presence of interesting objects.

DISCOVR [15] is a system which consists of three different components (see Figure 6) in the form of separate services: Sensor, recommendation and utility services. A sensor is a service that acquires information about the user's context (eg by introducing the IP address of the user in the "www.mon-ip.com/localiser-adresse-ip.php" site we can deduce the city where he lives), Recommendation services use the input in order to generate recommendations (context-aware ones, if the output of sensors is being incorporated). The last kind of services is utility services. Such a service might filter its input according to specific criteria, like selecting only resources that have a specific type, or limit the amount of results. The advantage of this method (Discover) is to provide an easy extensible architecture to integrate into recommender systems and prototype a quick way through the treatment that can be done in parallel.

```
<Statement id="16">
<content> <subject><UbisWorld:Dora /></subject>
<predicate><UserOL:eating</predicate>
<predicate-range><UserOL: restaurant,fast-food,pizzeria/>
</predicate-range><object>restaurant</object>
<predicate><UserOL:table/></predicate>
<predicate-range><UserOL:open-air,inside..../>
</predicate-range><object>open-air</object>
</content>
<restriction><season>summer</season></restriction>
<meta>
<owner><UbisWorld:Dora /></owner>
<privacy><UbisWorld:friends /></privacy>
<purpose><UbisWorld:commercial /></purpose>
<retention><UbisWorld:short /></retention>
<viewer><UbisWorld:MyMap /></viewer>
<explanation confidence="0,75"  creator="Dora" evidence="
Interface input "method="acquire_pref" />
</meta>
</Statement>
```

Figure 5 MUP fragment of a person named Dora.

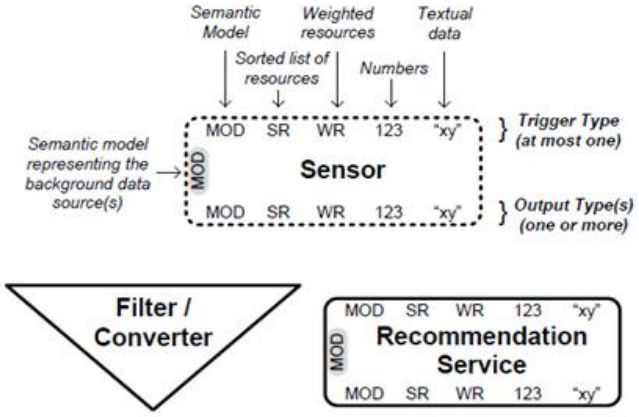

Figure 6 Types components for the DISCOVER architecture.

MOD: Semantic model.

SR: Resources list.

WR: The resource weight includes between 0 (for the lowest) and 1 (for the most important).

123: Numbers.

'' XY'': Text data types.

Sourcetone [24] (Malcolm Goodman and Dr. Jeff Berger, 2004) is a system that recommends the music depending on the mood of users. It classifies music into 21 different areas designed to help improve the listener's mood, activity and overall health. This was setup through research done in collaboration with places like Harvard Medical Center which collected data from test groups of people

in an effort to find out how different music made each individual feel. Figure 7 shows a recommendation of the singer "Lady Gaga" for a user who has an "agitated" mood.

Figure 7 Screenshot of the www.sourcetone.com website

Amazon [5] is a recommendation system sensitive to context which was created by Jeff Bezos in July 1995, and the French subsidiary was opened in 2000. Amazon.com is an American e-commerce company based in Seattle. His best-known specialty is selling books, but it has diversified into other products, including the sale of all types of cultural products: CD, music download, DVD, digital cameras, hardware and household appliances, etc. The system requires users to connect via their name (see Figure 8.A) to create their profile and provides a "find a gift" button (see Figure 8.B) for each user to distinguish between specific user preferences and preferences of the person to whom it will offer this gift.

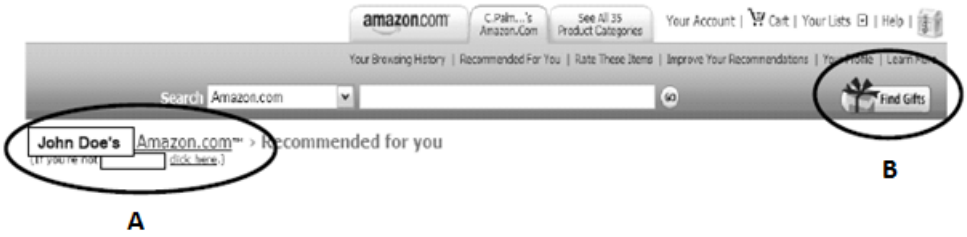

Figure 8 Toolbar of www.amazon.com website.

The personalized access model (PAM) [39] is a system is to explore the possibilities of integrating contextual information in conventional systems recommendation following the work initiated by G.Adomavicius et al. [3]. PAM provides a set of services, concepts and personalization techniques

that are adaptable to different applications. This platform meets the following requirements:

- Dispose of meta models profile and context that allows it to be adaptable to all kinds of applications.
- Be independent.
- Provide a set of services that can customize existing applications (partial use), or even build new applications customization (full use).
- Make reusable information ensuring their sustainability.

Figure 9 shows the architecture of this platform. It is divided into three layers: (i) a persistence layer, (ii) a functional layer and (iii) a communication layer.

- **The persistence layer:** Concerns storing information in the meta model profile and context in a database, ensure their sustainability, and provides an API that allows access to this information.
- **The functional layer:** Includes all services offered by the platform. These services are: instantiation, updating, contextualization profile, matching profiles, filtering profiles relative to the context, finally, the queries reformulation.
- **The communication layer:** Is the graphical interface. It makes the connection between the personalized access model and the user, and allows access to the database of profiles and contexts, and also run the services offered by the platform.

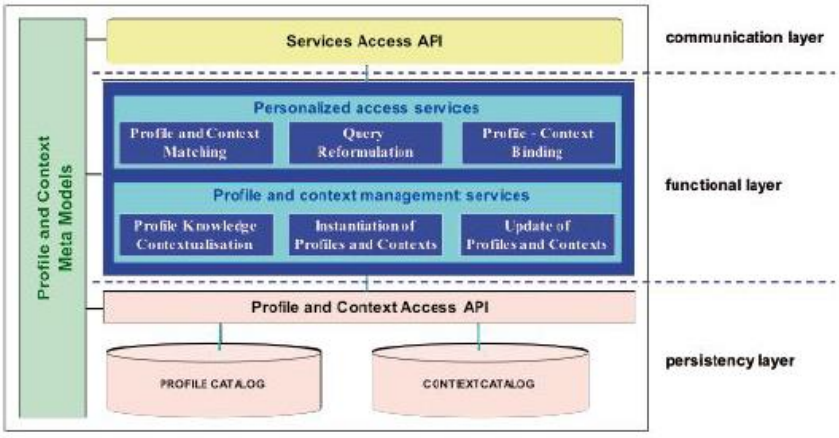

Figure 9 Personalized Access Model architecture.

The recommendation system PAM combines the content-based and collaborative filtering method. The content-based method provides us the information about the user profile by analyzing the content of their preferences, these profiles are then compared to determine which users are similar and perform the collaborative recommendations. As for collaborative filtering, it uses the votes of the similar k neighbors to an active user in order to infer the missing votes using aggregate functions.

In [27], a context-aware rating predictor based on Factorization Machines (FM) [26] is proposed. The authors show how FMs can be applied to a wide variety of context domains including categorical, set categorical or real-valued domains. For learning the model parameters of FMs, a new algorithm was proposed that is based on alternating least squares (ALS). This algorithm directly finds the optimal solution for one model parameter given all the other ones and a joint optimum is found within a few iterations.

The main advantage of the new ALS algorithm over the stochastic gradient descent SGD is that no learning rate has to be determined. This is very important in practice, because the quality of SGD learning relies largely on a good learning rate and thus an expensive search has to be done. This is not necessary for ALS.

Tensor factorization approach (TF) [28] captures the intrinsic multi-way interactions between users, items, and aspects, and to predict the unknown ratings on items. They propose a new CF framework that integrates multi-faceted opinions in the reviews into the CF process, in order to tap the rich sentiment information embedded in the reviews, and to alleviate the cold start/data sparsity problems. In particular, the framework of the authors consists of two components, namely (1) opinion mining, and (2) rating inference.

The first component extracts and summarizes the multiple aspects of opinions expressed in the reviews, and generates numerical ratings on the different aspects. Generally, an opinionated statement consists of two parts: the opinion word (like "excellent" or "bad") and the opinion aspect (the target object that is being evaluated). For the purpose of mining and summarizing opinions at the aspect level, they first employ a double propagation approach to expand opinion words and extract the aspect terms. Latent Dirichlet Allocation [10] is used to cluster those aspect terms into latent aspects. The corresponding opinions can then be aggregated to get a user's ratings on each of these aspects. Since each review contains multiple opinion aspects, the result of the opinion mining component is a set of rating matrices, each corresponding to one of the aspects.

The second component uses tensor factorization to infer the overall rating a user may give to an item, forming the basis of item recommendation. They focus on exploring optimization techniques for rating estimation. The rating matrices for different aspects, together with the overall ratings, constitute a tensor, i.e., a 3-dimensional array. They explore the use of tensor factorization to capture the underlying latent structure of the tensor, and the result of the factorization can be used for inferring the unknown ratings. The method can be seen as an extension of matrix factorization techniques widely applied in collaborative filtering; it can preserve the multi-dimensional nature of the data and extract the latent factors along each dimension.

6.2 Comparison criteria

Table 8 shows a comparison between the different approaches presented above according to the approach type used, data collection type, definition and context type, as well as the evaluation domain and the flexibility of the approach.

6.2.1 Approach recommendation

There are three types of recommender systems: The content-based filtering systems (CBF), collaborative filtering systems (CF) and hybrid systems. In CBF systems, content descriptors are

directly matched with user profiles to assess their usefulness. Only the content of an important score will be recommended to users. CF systems are based on the observation that users with the same behavior (in the past) probably have the same interests. These systems require no content description. Their main idea is to recommend to a user, content consumed (appreciated) by users who are similar to him. Finally, the hybrid systems combine the techniques of content-based filtering and those of collaborative filtering to improve the relevance of the recommendations. An example of such systems is the collaboration with the content in which the CBF technique is used to learn user profiles. These profiles are then used to calculate the recommendations using the CF technique.

6.2.2 Data collection type

Data collection can be explicit or implicit. It is explicit when the user enters his preferences manually (case of Sourcetone system), or implicitly when the system infers the user profile according to his behavior (case of Discover system). Most of these systems have resorted to the implicit data collection, which requires no effort from the user, even if it is quite difficult to obtain such data.

6.2.3 Context type

The pre-filtering technique is the most used because it is much simpler to implement and less expensive than the post-filtering method. For instance, The MyMap system uses pre-filtering technology unlike Davi system which opts for the post-filtering technique.

6.2.4 Context definition

Some systems have defined the context based on the time factor as in Micro-profiling approach. Other systems have used the state of mind (mood) of the person as contextual information to recommend music (case of Sourcetone system). The context for other systems can be any entity, according to user needs. This is the case of MyMap approach where the context is not defined (specified by "any" in Table 8).

6.2.5 Evaluation domain

It specifies the area in which the systems were evaluated. The News @ hand system is evaluated in the domain of the news, whereas the Davi system is evaluated in any field "any".

6.2.6 Flexibility

This criterion indicates if the system can be used in several domains. For example, the use of micro-profiling system is not flexible while the Discover system is flexible.

Table 8 Summary table of existing contextual recommendation systems.

Criteria \ Approach	**Micro-Profiling**	**DaVI**	**DISCOVR**	**News@hand**	**MyMap**	**Sourcetone**	**Amazon**	**PAM**	**FM**	**TF**
Recommendation approach	CF	CF+AR	Hybrid	Hybrid	CBF	-	CF+CBF	CF+CBF	-	CF
Data collection type	Implicit	Implicit	Implicit	Implicit	Implicit/ Explicit	Explicit	Implicit / Explicit	Implicit / Explicit	Explicit	Explicit
Context type	Pre-filtering	Post-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering	Pre-filtering
Context definition	Time	Access to a web page or item	Any	Built in real time	Any	State of mind	Any	Any	Any	opinions from reviews
Evaluation domain	Music	Any	Any	News article	Tourism	Music	e-commerce	Movies	Yahoo! webscope, Food and Movies	Movies
flexibility	No	Yes	Yes	No	No	No	No	Yes	Yes	Yes

7 Conclusion

The integration of context in traditional recommendation systems, improves their performance. We present in this paper, a new approach to take into account contextual information in recommender systems. It is general in the sense that it can be applied to the recommendation of contents in general (Book, Url, item, product, movie, song, restaurant etc.). In this approach, the context is defined in terms of objects, not users. It is intrinsic, that is to say, defined over the set of attributes for the object itself. We show, by combining it with the user's profile, how it can improve the accuracy of recommendation and better meet users' requirements.

The architecture of the system using this new method consists of three essential components, namely the creation of users' profiles, the context resolution and the profile / content matching. The first algorithm was realized to create automatically users' profiles based on the analysis of interaction logs. A second algorithm was developed to allow the extraction of the intrinsic context directly from the content identified as potential candidates for the recommendation. This second algorithm, based on the similarity between instances of context, solves even the problem of contexts that are not compatible with the user profile. As regards the matching profile / content, we developed and applied a new similarity measure which linearly combines semantic and vector measures.

To better understand the different aspects taken into account by our approach, we applied it in recommendation of movies to a user by choosing as context the director of a film. We have also presented and compared the main context-aware systems based on a number of properties that we had defined.

Finally, to confirm the effectiveness of our application we performed the tests on the data collected from two large movie databases namely MovieLens and IMDB, using the proposed recommendation approach. These tests have shown the importance of the context in a recommendation system. The

evaluation of the system with context showed better results in comparison with those of a conventional RS evaluation (without context).

This work is far from being finished. There are perspectives that should be considered, such as:

- Introduction of the context in RS by using the collaborative filtering and hybrid filtering,
- Integrate the extrinsic context, which is a set of attributes relating to the environment of user interaction, in the recommendation process.
- Combination of two types of contexts, extrinsic and intrinsic one, to refine the list of products to recommend.

References

1. Abbar, S., Personalized access model for content delivery platforms: A service oriented approach. PhD thesis, Versailles University. France, 2010.
2. Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. Towards a better understanding of context and contextawareness. In HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pages 304-307, London, UK, 1999. Springer.
3. Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A., Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Information System, 23(1):103–145, 2005.
4. Adomavicius, G., and Tuzhilin, A. 2008. Context-Aware Recommender Systems. Tutorial presented at the 2008 ACM Conference on Recommender systems, 335-336.
5. Amazon: www.amazon.com.
6. APMD: http://apmd.prism.uvsq.fr
7. Baba-Hamed, L., Soltani, R. et Sabri, K., Construction d'une ontologie pour la recommandation de films à un utilisateur. Actes des Ateliers des 21es Journées Francophones d'Ingénierie des Connaissances (IC 2010), Nîmes, France, juin 2010.
8. Baba-Hamed, L., Abbar, S., Soltani, R., et Bouzeghoub, M., Elaboration et Evaluation d'un Système de Recommandation Sémantique, in proc. 1st international Conference on Information Systems and Technologies (ICIST'11), PP.515-523, 24-26 April, 2011.
9. Baltrunas, L., Amatriain, X., Towards Time-Dependant Recommendation based on Implicit Feedback, cars 2009.
10. Blei, D., Ng, A., and Jordan, M., "Latent dirichlet allocation," The Journal of Machine Learning Research, vol. 3, pp. 993–1022, 2003.
11. Cantador, I., Castells, P., Semantic Contextualisation in a News Recommender System, Cars 2009.
12. Castagnos. S., Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information, thèse de doctorat de l'université Nancy 2. Novembre 2008.
13. De Carolis, B., Cozzolongo, G., Pizzutilo, S., and Silvestri, V. 2007. MyMap: Generating personalized tourist descriptions. Applied Intelligence 26, 2, 111-124.
14. Domingues, M., Mário Jorge, A., Soares, C., Using Contextual Information as Virtual Items on Top-N Recommender Systems, cars 2009.
15. Hussein, T., Linder, T., Gaulke, W., Ziegler. J., Context-aware recommendations on rails. Cars 2009.

16. Jiang, J., and Conrath, D., Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings of the 10th International Conference on Research in Computational Linguistics, Taiwan. 1998.
17. Kostadinov, D., Personnalisation de l'information et gestion de profils utilisateur. PhD Thesis, University of Versailles Saint-Quentin-en-Yvelines, Décembre 2007.
18. Markov, Kr., and Ivanova, Kr., An ontology-content-based filtering method. In Proceedings of the Fifth International Conference "Information Research and Applications", Varna, Bulgaria. June 2007.
19. MovieLens: http: movielens.umn.edu
20. Nguyen, A. T., COCoFil2: Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés. Thèse de Docteur. Université Joseph Fourier, Grenoble I. Novembre 2006.
21. Panniello, U., Tuzhilin, A., Gorgoglione, M., Experimental Comparison of Pre- vs. Post-filtering Approaches in Context-Aware Recommender Systems, 2009.
22. Pazzani, M. J., A framework for collaborative, content-based and demographic filtering. In technical report, University of California, Irvine., 1999.
23. Peralta, V., Extraction and Integration of MovieLens and IMDb Data, technical report, APMD project, Laboratoire PRiSM, Université de Versailles .July 2007.
24. Sourcetone : www.sourcetone.com.
25. Shoval, P., Maidel, V., Shapira, B., An Ontology- Content-Based Filtring Method, I.Tech-2007, Information Research and Applications, 2007.
26. Rendle, S., Factorization machines. In Proceedings of the 10th IEEE International Conference on Data Mining. IEEE Computer Society, 2010.
27. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L., Fast Context aware Recommendations with Factorization Machines, Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval 2011, pages 635-644, ISBN: 978-1-4503-0757-4.
28. Wang, Y., Liu, Y., Yu, X., Collaborative Filtering with Aspect-based Opinion Mining: A Tensor Factorization Approach, 2012 IEEE 12th International Conference on Data Mining, pages 1152 – 1157, ISBN: 978-1-4673-4649-8.