Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 35 (2008) 790–804

Expert Systems with Applications

www.elsevier.com/locate/eswa

Evaluation of recommender systems: A new approach

Félix Hernández del Olmo *, Elena Gaudioso

Universidad Nacional de Educación a Distancia, Departamento de Inteligencia Artificial, C/Juan del Rosal, 16, 28040 Madrid, Spain

Abstract

It is difficult to deny that comparison between recommender systems requires a common way for evaluating them. Nevertheless, at present, they have been evaluated in many, often incompatible, ways. We affirm this problem is mainly due to the lack of a common framework for recommender systems, a framework general enough so that we may include the whole range of recommender systems to date, but specific enough so that we can obtain solid results. In this paper, we propose such a framework, attempting to extract the essential features of recommender systems. In this framework, the most essential feature is the *objective* of the recommender system. What is more, in this paper, recommender systems are viewed as applications with the following essential objective. Recommender systems must: (i) choose *which* (of the items) should be shown to the user, (ii) decide *when* and *how* the recommendations must be shown. Next, we will show that a new metric emerges naturally from this framework. Finally, we will conclude by comparing the properties of this new metric with the traditional ones. Among other things, we will show that we may evaluate the whole range of recommender systems with this single metric.

Keywords: Evaluation; Recommender systems; Interactive systems; Metrics

1. Introduction

Recommender systems were originally defined as ones in which "people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients" (Resnick & Varian, 1997). Now, a broader and more general definition is taking place in the field, referring to recommender systems as those systems that "have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" (Burke, 2002). The last implies that current recommender systems have a clear main objective: to *guide* the user to *useful/interesting* objects. As a result, evaluation of recommender systems implies assessing how much of this goal has been achieved.

Although the definition (and so the goal) of these systems has evolved through the years, we can hardly affirm the same about their metrics. From the early first systems to date, a majority of the published empirical evaluations are focused on measuring only how close a recommender system predictions are to the user's true preferences.

Although popular, these measures do not match the complete and general goal of recommender systems stated above at the beginning. Moreover, some metrics are enforcing recommender systems to follow particular policies, limiting their possibilities. Indeed, instead of following a common goal for any recommender system, we claim that, at present, unfortunately, a fuzzy objective exists, which is supplied *de facto* by a diversity of current metrics. To illustrate this, we quote some appropriate paragraphs from Herlocker, Konstan, Terveen, and Riedl (2004):

> The challenge of selecting an appropriate metric is compounded by the large diversity of published metrics that have been used to quantitatively evaluate the accuracy of recommender systems. This lack of standardization is damaging to the progress of knowledge related to collaborative filtering recommender systems. With no standardized metrics within the field, researchers have continued to introduce new metrics when they evaluate

* Corresponding author.
E-mail address: felixh@dia.uned.es (F. Hernández del Olmo).

0957-4174/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2007.07.047

their systems. With a large diversity of evaluation metrics in use, it becomes difficult to compare results from one publication to the results in another publication. As a result, it becomes hard to integrate these diverse publications into a coherent body of knowledge regarding the quality of recommender system algorithms.

Additionally, these metrics have certain limitations.

Researches who want to quantitatively compare the accuracy of different recommender systems must first select one or more metrics. In selecting a metric, researchers face a range of questions. Will a given metric measure the effectiveness of a system with respect to the user tasks for which it was designed? Are results with the chosen metric comparable to other published research work in the field? Are the assumptions that a metric is based on true? Will a metric be sensitive enough to detect real differences that exist? How large a difference does there have to be in the value of a metric for a statistically significant difference to exist? Complete answers to these questions have not yet been substantially addressed in the published literature.

Despite these problems, we should not overlook the final objective of any metric for recommender systems. A good metric must have the capacity to compute the "good behavior" of recommender systems and keep similar values whenever these systems are found in similar environments (i.e. dataset of users' interactions).

In addition, it must be clearly defined what "good behavior" means. In any case, it should match the real objective of the general recommender system stated above at the beginning: to *guide* the user to *useful/interesting* objects. It is very noticeable that this objective is composed of two different tasks: (i) to *generate suggestions* to be accepted by the user and (ii) to *filter* useful/interesting objects. The first task has to do with the most *external* and *interactive* behavior that any recommender directly reveals to the user. The second task is related to the known task "find good items", with a more *internal* and less *interactive* behavior. However, in spite of the large number of metrics published to date, it is difficult to identify these two tasks together, as a whole objective, on them. Moreover, a certain research bias towards the second part of the objective could be noticed, while frequently losing the first part. In this article we will set off towards the development of a new metric which, among other things, gathers both of these pieces together.

The main contributions of this paper are: (1) A presentation of the explicit problems we find when we attempt to apply the commonly accepted framework for recommender systems, and try to measure, *in a common way*, the performance of *any* current type of these systems. (2) As a solution to the latter, we propose a new logical framework, general enough to incorporate in it every recommender system to date, while keeping it specific enough to obtain important and solid results. (3) A new metric that measures the performance of any recommender system defined into the last framework limits. (4) A comparison between this new metric and the traditional ones.

The article's layout is arranged as follows. In Section 2 we review the evaluation of recommender systems in the past, focusing on the most frequent metrics employed for it. Also, in the same section, we will highlight some common assumptions related to these metrics, and the problems derived from them. In Section 3, we will propose a new logical framework. We will create it so as to gather the whole range of recommender systems in the new single framework. In Section 4, we will reformulate the traditional metrics in the terms of this framework. Afterwards, we will define a new metric that emerges naturally from the framework. Finally, we will enumerate the advantages of this new metric with regard to the traditional ones. Conclusions and future work are in Section 5.

2. Previous metrics for evaluating recommender systems

Whenever a recommender system is evaluated, the metric used for it is built upon certain assumptions. Therefore, we will devote this section to a review of the most common metrics, highlighting the assumptions on which they are based and which we consider most significant. To this end, first of all, we start describing the commonly accepted framework followed in the field to define the general recommendation process.

In this current framework, a recommender system is embedded in another system, which contains a number I of items available to be recommended. In order to start the recommendation process, some of those items must be rated. In most of the recommender systems these ratings are obtained explicitly. In some other cases, the ratings are inferred from other users' interactions, then they are called implicit ratings.

Afterwards, once the recommender system has enough ratings, it can start the process. For each recommendation, a number $N \leqslant I$ of objects are chosen by the recommender, and shown to the target user. Additionally, some recommender systems also rank the marked-out objects in order to show them as an *ordered list*. Next, the user *presumably* will investigate these items starting at the top of this list.

Finally, in order to evaluate the performance of the recommender system, for each object shown to a particular user we must measure how *close* the utility of the *shown* object is with respect to the preferences of the user. In the case of an ordered list, additionally, we should take into account the place that each recommended object has in this list. Now, we will have a quick view on how this evaluation has been carried out to date.

2.1. First considerations

In order to measure the *closeness* of predictions to users' real preferences, a numerical representation is normally used. In addition, for reasons of clarity, we will use the same notation along the current section. To this end, let us call $P(u,i)$ the predictions of a recommender system for every particular user u and item i, and $p(u,i)$ the real preferences.

Clearly, the function $p(u,i)$ can never be known with absolute precision. Therefore, the values of this function are most usually *estimated* by means of the users' previous ratings. As we said above, these ratings can be obtained explicitly or implicitly.

In some cases, both functions $p(u,i)$ and $P(u,i)$ will offer only two values 1 or 0, which means that a particular item i is considered *useful* or *useless*, respectively, for a particular user u. For this singular case, we will say that p and P are *binary* functions.

2.2. Accuracy metrics

Accuracy metrics measure the quality of nearness to the truth or the true value achieved by a system. Perhaps, *accuracy* is the most used and well-known metric into the field of Artificial Intelligence, and, in general terms, it can be formulated as in (1).

Particularizing the metric to the recommender system's field, it can be formulated as in (2). Under this form, *accuracy* can be found in the evaluation of many cases (Armstrong, Freitag, Joachims, & Mitchell, 1995; Billsus & Pazzani, 1998; Burke, 2002; Pazzani, Muramatsu, & Billsus, 1997; Rogers, Flechter, & Langley, 1999).

$$accuracy = \frac{number\ of\ good\ cases}{number\ of\ cases} \tag{1}$$

$$accuracy = \frac{number\ of\ successful\ recommendations}{number\ of\ recommendations} \tag{2}$$

Now, assuming that a "successful recommendation" is equivalent to "the usefulness of the recommended object is *close* to the user's real preferences", and using the functions p and P introduced previously, we may be more formal and reformulate *accuracy* as in (3). In this equation, we consider that p and P are *binary* functions. Additionally, $r(u,i)$ is 1 if the recommender showed the item i to the user u, and 0 otherwise. Finally, $R = \sum_{u,i} r(u,i)$ is the number of recommended items shown to the users.

$$accuracy = \frac{\sum_{(\forall u,i/r(u,i)=1)} 1 - |p(u,i) - P(u,i)|}{R} \tag{3}$$

Also common in the recommender systems' field is the metric *mean absolute error* (*MAE*). This metric measures the average absolute deviation between each predicted rating $P(u,i)$ and each user's real ones $p(u,i)$. Then, due to the fact that only rated items can show us each user's real preferences, we may derive the (4), where i must have been rated by u (to obtain $p(u,i)$). In this, we consider N as the number of observations available, which obviously depends on the number of items properly rated. Of course, the higher this number, the better the estimate.

$$MAE = \frac{\sum_{u,i} |p(u,i) - P(u,i)|}{N} \tag{4}$$

Several recommender systems make use of this metric for the evaluation (Breese, Heckerman, & Kadie, 1998; Herlocker, Konstan, Borchers, & Riedl, 1999; Shardanand & Maes, 1995). Also, there are some direct variations of *MAE*. For instance, *mean squared error*, *root mean squared error*, or *normalized mean absolute error* (Goldberg, Roeder, Gupta, & Perkins, 2001).

Finally, notice the similarity between *accuracy* and *MAE* metrics. In fact, if we consider that p and P are binary functions, and also consider that the (*MAE*) number of recommender *predictions* is the same as the (*accuracy*) number of recommender *recommendations* (thus, $N = R$), then we can collapse both formulas into one: $MAE = 1 - accuracy$.

However, notice a subtle but important difference between them. To be computed, *MAE* requires rating *observations* (to estimate p) plus recommender *predictions* (to get P). However, the computation of *accuracy* requires the value of the nearness to the truth of the recommender system's *decisions* (=recommendations). Consequently, the similarity between these two metrics mainly exists because it is widely assumed that the "nearness to the truth" that is presumed in the *accuracy* metric may only be computed by $|p(u,i) - P(u,i)|$. Further discussion about the use of *predictions* or *decisions* in the computation of a metric will put forward in Section 2.6. Presently, consider it a significant idea which will repeatedly appear during the presentation of the following metrics.

2.3. Information retrieval measures

Information Retrieval (IR) is a consolidated discipline whose objectives are somehow related to the ones of the recommender systems (RS) field. Moreover, IR research is focused on the *retrieval* of *relevant* documents from a pool, which is not far from the related RS task of recommending *useful/interesting* items from a pool. Therefore, it is not a surprise that the IR field is a good supplier of tools for the RS field. Among these tools, we find its metrics, whose key ones are: *precision* & *recall* (Baeza-Yates and Ribeiro-Neto, 1999). Also, we can find the related ROC analysis. In fact, several recommender systems have been evaluated by them (Billsus & Pazzani, 1998, 2000; Herlocker et al., 1999).

To compute these metrics, *precision*, see (5), and *recall*, see (6), a confusion matrix is expected such as the one in Table 1. This table reflects the four possibilities of any *retrieval decision*. In order to work with *recommendation decisions*, to use them into the RS field, firstly we must switch the IR terms "retrieved" and "relevant" to the RS terms "recommended" and "successful recommendation" respectively. Again, notice we are working with recommender's *decisions*, thus in principle no ratings are needed.

Table 1
Confusion matrix of two classes when considering the retrieval of documents

	Relevant	Non-relevant
Retrieved	*a*	*b*
Not retrieved	*c*	*d*

Diagonal numbers *a* and *d* count the correct *decisions*: retrieve a document when it is relevant, do not retrieve it when it is non-relevant. The numbers *b* and *c* count the incorrect cases.

Even though, we can always translate non-binary (rating) functions $p(u,i)$ to binary ones (by means of thresholds).

$$precision = \frac{a}{a+b} \tag{5}$$

$$recall = \frac{a}{a+c} \tag{6}$$

The meaning of each measure is really intuitive. Thus, *recall* represents the *coverage* of useful items the recommender system can obtain. In other words, this metrics measures the capacity of obtaining all the useful items present in the pool. On the other hand, *precision* shows the recommender's capacity for showing only useful items, while trying to minimize the mixture of them with useless ones.

Observe that we can always improve one of the metrics by declining the other (see Fig. 1). For instance, a recommender system could recommend a large number of items to the user, then the coverage would be maximal (almost all the useful items would be shown), though the precision would be as bad as the proportion of useful items present in the pool. As a result, we look for the optimization of *recall* and *precision*, both at the same time (see Fig. 2).

An alternative to the last metrics is ROC (Receiver Operating Characteristic) analysis (Haley & Mcneil, 1982). A ROC curve represents *recall* against *fallout* (7). The objectives of ROC analysis are to return all of the relevant documents without returning the irrelevant ones. It does so (see Fig. 3) by maximizing *recall* (called the true positive rate) while minimizing the *fallout* (false positive rate).

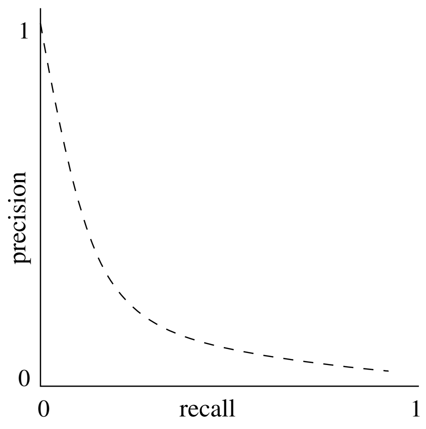


Fig. 1. A common curve representing *precision* against *recall*. Notice that the higher is one of the metrics, the less is the other one.

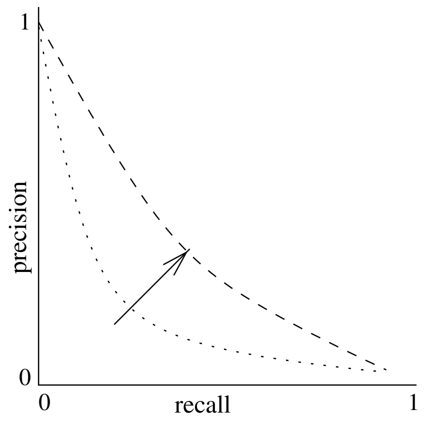


Fig. 2. The simultaneous optimization of *recall* and *precision* can be graphically represented as pushing the peak of the curve towards the right-top corner (towards the point *recall* = 1, *precision* = 1).

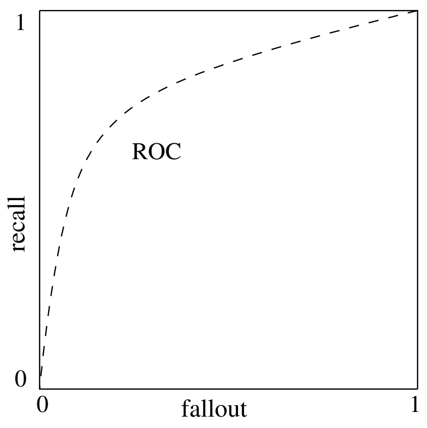


Fig. 3. Common ROC curve representing *recall* against *fallout*. Notice that ROC curve is a monotonically increasing function. Thus, the earlier it gets the top value of *recall*, the better the ROC result. In fact, the Area Under this Curve (AUC) is usually used as a measure of this result.

$$fallout = \frac{b}{b+d} \tag{7}$$

Notice that the optimization of a ROC curve is similar to the optimization of *precision/recall* curves (see Fig. 4). What is more, methodologically, optimizing *ROC* curves is equivalent to optimizing *precision/recall* curves (Davis & Goadrich, 2006; Fisher, Fieldsend, & Everson, 2004). As a result, we can focus the evaluation on whatever of both analysis.

Some other measures derived from *precision/recall* are *F-measures* (8), which try to grasp in a single value the behavior of both *precision* and *recall* metrics. Thus, varying β, the value of F_β weights one metric over the other. However, the most usual of the *F-measures* is $F1$ $(\beta = \frac{1}{2})$, see (9), which is the harmonic mean of *precision* and *recall*.

$$F_\beta = \frac{precision\ recall}{(1-\beta)\ precision + \beta\ recall} \tag{8}$$

$$F1 = \frac{2\ precision\ recall}{precision + recall} \tag{9}$$

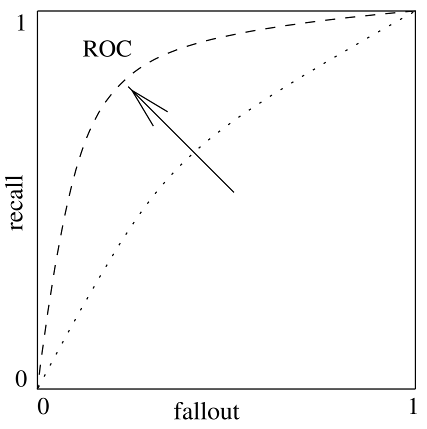


Fig. 4. ROC optimization can be graphically represented as pushing the peak of the curve towards the left-top corner (towards the point *recall* = 1, *fallout* = 0).

2.4. Rank metrics

These metrics are used in the case of recommenders based on the display of an ordered list of elements. These systems provide a ranked list of recommendations where those that rank highest are predicted to be the most preferred.

In the spirit of quantifying the *closeness* of a recommender's predictions to users' real preferences, some rank metrics will measure the *correlation* of the rank of predictions $P(u,i_1) \geqslant P(u,i_2) \geqslant P(u,i_3)\cdots$ to the rank of real preferences $p(u,i_1) \geqslant p(u,i_2) \geqslant p(u,i_3)\cdots$ Examples of systems that apply these techniques are (Hill, Stead, Rosenstein, & Furnas, 1995) which applies the well-known Pearson's product-moment correlation (Sheskin, 2003) or Fab (Balabanovic et al., 1997) which applies NDPM (Normalized Distance-based Performance Measure) (Yao, 1995).

Alternatively, other rank metrics as Half-life utility (Breese et al., 1998; Heckerman, Chickering, Meek, Rounthwaite, & Kadie, 2001) weight decreasingly this predicted/real-preference closeness. To this end, they postulate that each successive item in the ordered list is likely to be viewed by the user with an exponential decay.

2.5. Other metrics

In the life of the recommender systems field a lot of ad hoc measures have appeared. However, most of them are not far from the known accuracy metrics *accuracy* or *MAE* explained above. In fact, we will see that they are mostly related to one of both.

For instance, some systems, such as Billsus and Pazzani (1998), make use only of the first top n recommended items in order to compute *accuracy*. Other systems as INTRIGUE (Ardissono, Goy, Petrone, Segnan, & Torasso, 2003) or SETA (Ardissono & Goy, 2000) use a metric named *satisfaction score*, which measures the degree of matching between an item and a group of users by analyzing the preferences of a group of users and the properties of the item. The assumptions are similar to those made by *MAE*, but using groups of users (stereotypes) instead of single users.

2.6. Discussion

It cannot be denied that there is a lack of uniformity in the current metrics for the evaluation of recommender systems, which perhaps is due to the large number of them. However, we will attempt to classify them into one of the next three categories, depending on the way they quantify the good behavior of the recommender system.

(1) *Rating prediction*. These metrics are focused on measuring the capacity of the recommender system for predicting the rating a user will give to an item before she does it.
(2) *Ranking prediction*. These metrics are focused on measuring the capacity of the recommender system for predicting the rank a user will set on a set of items before she does it.
(3) *Successful Decision Making Capacity* (*SDMC*). These metrics are focused on measuring the capacity of the recommender system for making successful decisions (recommendations).

Bearing this classification in mind, we should classify *MAE* (and related metrics) into the first class, ranking metrics into the second class, and *accuracy* (and related metrics) and IR metrics into the third class.

Now, if we bring forward the main goal of a recommender system (stated in the beginning of Section 1) and we observe what the first and second class of metrics try to measure, we could think of some kind of "over-particularized" metrics. In fact, we should not make more assumptions than the ones actually required. However, there is no mention to any rating or rank in the definition of a recommender system's goal. Moreover, even though assuming that a useful item might be one whose $p(u,i)$ is high enough, it is really arguable that we can derive an exact $p(u,i)$ function only by means of users' ratings (Cosley, Shyong, Istvan, Konstan, & Riedl, 2003). In conclusion, if our desire is to be strict while building up a metric that measures just the real objective of *any* recommender system, we must be cautious while making assumptions that could set apart some proper recommender systems from being measured.

Therefore, if we want to keep the methodology general enough to include as recommender whatever system with the objective already stated, we must bear in mind that this objective is expressed in terms of the recommender system's decisions. Thus, *SDMC* metrics appear as the most appropriate for this task.

However, note that when we refer to a useful recommendation in an SDMC metric, it is widely considered that a *successful recommendation* is one whose *recommended item* interest corresponds to the target user's real interest. In other words, if we have available the users' binary preference functions, *successful recommendations* are the ones that comply with the next: $|P(u,i) - p(u,i)| = 0$. At this point, we want to extract two important assumptions that stand behind this popular belief:

(1) It is widely assumed that a recommendation is successful if and only if the recommended item is useful. However, a recommendation destination is also to *guide* the user. Moreover, if a recommendation is not "opportune" nor "attractive" enough for guiding the user to the recommended item, the whole recommendation will be of no use in spite of the usefulness of the item. In other words, it is a requirement to choose *when* and *how* to recommend, apart from the common decision of *what* to recommend.

(2) It is widely assumed that a recommended item is useful if and only if this item preference matches the target user's preferences. However, the last is not always true. For instance, many e-commerce systems consider that a recommended item is useful whenever it provokes a transaction. Naturally, the latter could have nothing to do with user's preferences. In fact, it could be a necessity. Therefore, the usefulness of a recommended item must be reconsidered and generalized.

For the above mentioned reasons, we claim there is an "over-specification" in the current metrics for recommender systems. As a solution, we will provide a framework general enough to liberate recommender systems from following stipulated policies, but particular enough to obtain important results. To this end, we will presume *only* the objective that features recommender systems: to *guide* the users to *interesting/useful* objects.

3. A general framework for recommender systems

In this section we will develop a general framework for recommender systems. We call it *general* because this framework will introduce no more assumptions than the common objective of recommender systems. However, in spite of its abstraction, as we will verify at the end of this section, the framework will be particular enough so that we can obtain solid results. The goal consists of achieving a logical framework with a common terminology. Once all recommenders use the same terms, the creation of common metrics will be straightforward.

In order to build up such a framework we cannot expect to work with individual recommenders. Instead, we will work with categories of recommender systems. Through the years, several initiatives for classifying these systems have arisen. The most familiar one consists of categorizing them into two classes: *collaborative filtering* and *content-based filtering* (Adomavicius & Tuzhilin, 2005). In addition, hybrids of these can also be found (Balabanovic et al., 1997; Burke, 2002).

Despite its popularity, this classification is extremely dependant on the type of algorithm used by the recommender system, and it could leave proper recommender systems away from consideration. In fact, Burke had to extend this categorization by adding two more classes: *demographic* and *knowledge-based* classes (Burke, 2002). Furthermore, this categorization could need to be extended again in the future.

In order to grasp the whole range of recommender systems in a categorization from the first time, we must focus on their most common and essential feature: again, their objective. Focusing on this objective, as first noticed in Section 1, we may separate it into two sub-objectives: (i) to *guide*, (ii) to *filter* useful/interesting items. The first part has to do essentially with an interactive, dynamic and very temporal behavior; while the second part has to do with a somewhat opposite behavior, more permanent and less directly interactive.

This encourages us to introduce a new categorization based on which sub-objective a particular recommender system is most centered on. In fact, in the next section we will see that most current recommender systems are extremely biased over one of the two sub-objectives, leaving almost unattended the other one. Because of that, as a first and strict division, we will classify all possible recommender systems as *interactive* or *non-interactive* respectively, depending on which part of the objective they are more focused on. We will extend this concept in the following subsection.

3.1. Interactive and non-interactive recommender systems

We will proceed to develop the concept in an inductive way. To this end, we will use two characteristic examples of *interactive* and *non-interactive* recommender systems. Two opposite systems in terms of interaction are WebWatcher (Armstrong et al., 1995; Mladenic, 1996) and Syskill&Webert (Pazzani, Muramatsu, & Billsus, 1996, 1997), even though they both are usually included together into a same category as content-based recommenders.

Content-based recommenders try to suggest items similar to those considered *interesting/useful* for a given user in the past. As a result, they need to create a user profile for solving the question. In both cases, the items considered are web hyperlinks. In the first case, Syskill&Webert builds its user profile by means of the collected (explicit) ratings given by each user for some visited web pages. In the second case, WebWatcher *assists* each user "looking over her shoulder" by highlighting as *recommendations* some of the hyperlinks present in every shown page. This method is the so-called *annotation in context*. In WebWatcher, the user may click on the recommendation or not, but, whatever is done, the user's action is logged and its user profile updated.

It must be strongly pointed out that Syskill&Webert could have obtained the users' feedback before any recommendation had taken place. In fact, nothing forbids the users to rate objects in applications which include no recommender system at all. This could be the case, for instance, in applications used only for analysis of user data. By no means, WebWatcher can be considered the same case. In fact, in this recommender system the collected data depend strongly on the previous activity of the recommender over the user. Clearly, only recommendations which have been previously shown to the user do have the possibility of being followed. Therefore, to distinguish between these two types of recommender systems, we propose calling *interactive recommenders* those systems similar to WebWatcher, leaving the name *non-interactive recommenders* to the rest. Note that we do not mean that *non-interactive* recommenders do not need users' interactions, because any adaptive system does require them. Instead, what we mean is that, in the second case, the users' interaction data are collected from an external system that is not part of the recommender system at all. In other words, in *non-interactive* recommender systems, users' interaction data can be collected before any user's interaction with the recommender system has ever taken place.

Now, a significative question arises: can *interactive* and *non-interactive* recommender systems be evaluated by means of the same metrics? At a first sight, both of the examples, Syskill&Webert and WebWatcher, were evaluated by the same metric *accuracy*. However, if we go over the way they compute their values, we will notice that, even though they both have the same name, the metrics are not that similar (we will come back to these ideas in Section 4). In fact, we can see that in the case of Syskill&Webert, the evaluation objective is to "determine whether it is possible to learn *user preferences* accurately" (Pazzani et al., 1996). Yet, in the case of WebWatcher, the evaluation objective is "How well (accurate) can WebWatcher learn to *advise the user*?" (Armstrong et al., 1995). Again, we must remark that in the case of Syskill&Webert, evaluation may be executed with stored/static data, while in the second case interactive/dynamic data are required.

We could be tempted to say that these two systems are different, because they have different objectives. However, we claim they only have one common objective, even though each system pays attention to a different part of it. Thus, in the case of Syskill&Webert, its attention is on the second part of the common objective: *determine user preferences accurately* to provide *useful/interesting* items to the user. Though, in the case of WebWatcher, its attention is on the first part of the common objective: *learn to advise the user* to *guide* her correctly.

In conclusion, we purpose a framework in which any recommender system is formed by two different subsystems: an interactive and a non-interactive subsystem. Each subsystem will be in charge of its own sub-objective respectively: (i) to guide the user and (ii) to provide useful/interesting items. Usually, the recommender system will have one of the two subsystems more active than the other, and its class (interactive or non-interactive) will depend on this. Finally, notice that the closer both subsystems are to their own sub-objective at the same time, the closer the whole recommender system is to its global objective. The next section is devoted to explain these two subsystems in detail.

3.2. The guide and the filter subsystems

In our framework we propose that any recommender system is composed of two subsystems. We will start by a first definition of them.

- The *interactive* subsystem is in charge of *guiding* the user. Thus, we will call it *the guide*. In other words, *the guide* must answer *when* and *how* each recommendation must be shown to the user.
- The *non-interactive* subsystem is in charge of *choosing* the interesting/useful items among the large quantity of them. Thus, we will call it *the filter*. In other words, *the filter* must answer *which* of the items are useful/interesting candidates to become recommended items.

Note the subtle use of the terms "recommendation" and "recommended items" in the last two definitions. The main point is this: a full *recommendation* contains more information than a simple *recommended item*. In fact, a recommendation is composed of two parts: (a) the item to be recommended (*what/which*), plus (b) the way of and the context in which the recommender must recommend (*how* and *when* respectively). From this point of view, each recommendation might be seen as a kind of wrapper which envelops the recommended items (see Fig. 5). Also, the process to achieve a recommendation or *recommendation process* can be seen as a two steps policy: (1) some items are chosen from a (large) pool of them (which answers *what* to recommend), (2) each item is collected into a wrapper which must be molded (answering *how*) and promoted or displayed (answering *when*) to the user in a personalized way.

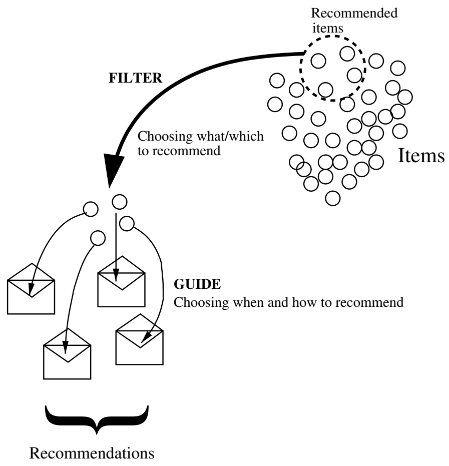

Recommended items
FILTER
Choosing what/which to recommend
Items
GUIDE
Choosing when and how to recommend
Recommendations

Fig. 5. Description of the process to obtain a recommendation. First, there is a filtering procedure, where the subsystem called *the filter* chooses a bunch of *useful/interesting* items for the guide. Second, the subsystem called *the guide* must wrap the received item (from the hands of the filter) to display them as recommendations.

Note, now metaphorically and in commercial terms, that the shape and the promotion date of a product is, at least, as important as the product itself. In addition, marketing and production people usually do not work together in the same department, because they are responsible for solving their own and different problems by themselves. Similarly, *the guide* (marketing staff) and *the filter* (production staff) subsystems of a recommender system are differentiated parts of it, and they are responsible for reaching their own objectives by themselves.

Additionally, the feedback of each department comes from different places. Thus, the quality of the marketing department is measured by the number of sales, though the quality of the production department is measured by the quality of its products. In a similar way, the guide and the filter subsystems should be measured by different metrics because their feedback come from different places. Firstly, we will measure the capacity of the filter by the number of useful/interesting items presented to the guide. Secondly, the capacity of the guide will be appreciated by the number of recommendations that have been *followed*.[1] Consequently, the guide must choose the most "attractive" and "magnetic" envelopes among the available to achieve the higher number of *followed recommendations*.

In conclusion, the recommendation process is defined as follows. First, the filter collects a number of items from the pool. This subsystem is the main responsible for the *usefulness* of the items. Second, the guide shows the recommendations to the user. As a result, this subsystem is the main responsible for the recommendations being *followed*. In the next section we will define this process and its components in a more formal way.

3.3. A formalized recommendation process for recommender systems

In this section we will formalize the elements that are part of the framework. We will divide the exposition in three parts: basic elements, subsystems goals and recommendation process.

3.3.1. Basic elements

What follows is a formal definition of the most basic elements of the general recommendation process.

- An *event* is a call to the system provoked by an action performed by the user. For instance, in the context of the web, every click on a hyperlink generates a new event.
- A *session* $s(u)$ is a *set of close events* provoked by a user u. Each *event* has associated an order in this set, thus we will be able to talk about *chains of events* of a session if required.
- A *recommendation process* is the sequence of actions that a recommender execute to produce a set of recommendations. In addition, we call *set of recommendation events* $ev_r(S)$, for one or more sessions S, to the subset of those single events that provoke the launch of a new *recommendation process*. In fact, we call *recommendation event* $e \in ev_r(S)$ to those events that always provoke one, but only one, *recommendation process*.

Now, for each recommendation event (or recommendation process), we define the following elements:

- I: the set of items available to be recommended.
- *recommendation window* $rw(e)$: a subset of I created for each new *recommendation event* (and, thus, for each *recommendation process*). In addition, every item that belongs to a recommendation window will be called *item to be recommended* $i \in rw(e)$.
- *filter*: the subsystem in charge of *creating* and *filling* a new recommendation window $rw(e)$ for each new *recommendation event* e.
- *guide*: the subsystem in charge of *wrapping* every *item to be recommended* $i \in rw(e)$ to constitute a *set of recommendations*. Additionally, at the end of the recommendation process, the guide must display this *set of recommendations* to the user. The display properties of each recommendation will depend on the assigned envelope. This will be discussed in more detail below.

3.3.2. Subsystems goals

Definition 1. A recommendation is called a followed recommendation if and only if a user has made use of it. This fact is independent of the usefulness or interest provided by this recommendation to this user or any other agent.

As we first said in Section 3.2, in some aspects, a recommendation is analogous to selling a product. In fact, it can be produced or provoked in spite of its usefulness or interest. Following this analogy, marketing departments and guide subsystems will have a similar objective: to sell or to push followed recommendations the more the better. Let us formalize this idea in the next definition.

Definition 2. The goal of the guide subsystem of a recommender system consists of (a) displaying the highest possible number of recommendations, but at the same time (b) suggesting only recommendations that are going to be followed.

Note that in the last definition there is no reference to the *usefulness* of the recommendation. Actually, it is not the matter of the *guide* subsystem. Instead, the usefulness of the recommendation is an issue to be solved by the *filter* subsystem.

[1] Informally, by *following a recommendation* we mean the act of opening the envelope by the user which directs her to the recommended item. However, note that this act does not imply that the user will ever make use of the recommended item.

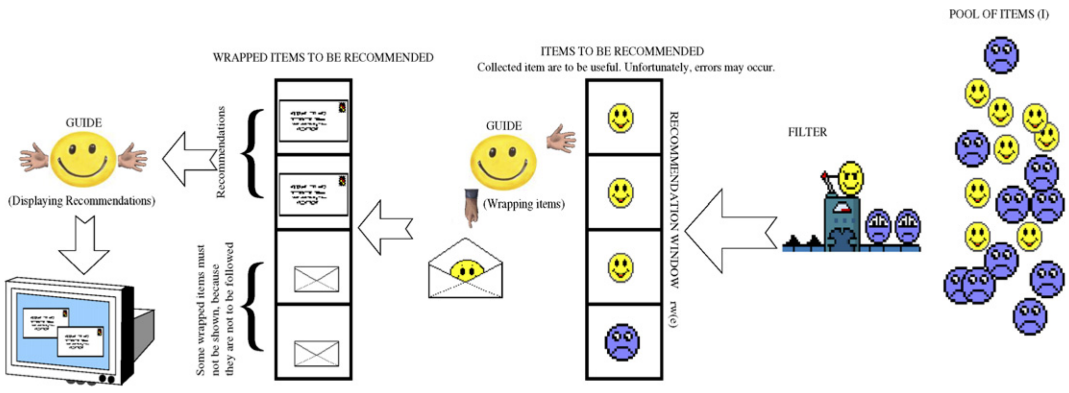

POOL OF ITEMS (I)
FILTER
ITEMS TO BE RECOMMENDED
Collected item are to be useful. Unfortunately, errors may occur.
RECOMMENDATION WINDOW rw(e)
GUIDE
(Wrapping items)
WRAPPED ITEMS TO BE RECOMMENDED
Recommendations
Some wrapped items must not be shown, because they are not to be followed
GUIDE
(Displaying Recommendations)

Fig. 6. From right to left, the steps followed by a recommender system during a *recommendation process* for each *recommendation event*. First, the *filter* gets the useful items from the pool and collect them into the *recommendation window*. Unfortunately, some errors may occur, i.e. the lower element of the recommendation window in the example. Afterwards, the guide must wrap the items in envelopes. Some envelopes will provoke that its item will not be displayed. Notice that the latter will only have to do with the *follow-ability* of the items and never with their *usefulness*. In fact, every item into the recommendation window must be considered *useful* by the guide, because this subsystem knows nothing about that term.

Definition 3. The goal of the filter subsystem consists of achieving that most/all of the followed recommendations are useful/interesting.

Once the basic elements of the common recommendation process have been introduced, we may define its mechanism.

3.3.3. The recommendation process

Formally, the steps of the recommendation process are enumerated as follows:

For each *recommendation event e* (see Fig. 6):

(1) From the set I of all objects available, the *filter* subsystem chooses a subset of them. We defined this subset as *recommendation window* $rw(e)$.
(2) The *guide* takes every *item to be recommended* which belongs to the *recommendation window* and wraps it in an envelope, transforming every *item to be recommended* into a *wrapped item to be recommended*. Note that some of the envelopes make its own *item to be recommended* "invisible". These cases are due to the fact that the guide has decided the item is not "attractive"[2] enough for being recommended.
(3) Finally, the guide displays the *recommendations* (*visible* wrapped items to be recommended). Also, observe that the position of each recommendation on the screen is given by its *envelope* properties.

At this point, we must highlight that the *recommendation window size* $|rw(e)|$ may change at each event. Actually, $|rw(e)|$ is chosen by the filter each time. Note that this subsystem is the main responsible for having filled the recommendation window with useful items. Therefore, the filter is the best candidate for choosing which $|rw(e)|$ works better.

[2] Note that this has nothing to do with "useful". The guide is only expected to show "follow-able" items, in spite of their *usefulness*.

In fact, a large size will allow the guide to have more freedom for transforming items into good recommendations. Note that, without such a liberty, the guide could find not enough "attractive" items to be shown as recommendations. On the other hand, the filter always has a limit in its capacity for filtering. Moreover, any filter, even a perfect one, always has a limit in the size of $rw(e)$: the number of useful items. In addition, the larger $|rw(e)|$, the higher the possibilities of filter failures. Therefore, the filter must balance these two opposite effects by finding an optimal $|rw(e)|$. Next, we will show an example in order to obtain a wider and clearer vision of the framework introduced.

3.4. The framework in action: an example

So as not to keep all the discussion in abstract terms, in this section we will illustrate the framework by means of two concrete instances. To this end, we have chosen examples of the two types of recommender system: interactive and non-interactive. Among other things, we will verify that both of the recommender systems are covered by the current framework without any difficulty.

As non-interactive recommender system we have chosen the one built into Amazon (Linden, Smith, & York, 2003; amazon.com, 2007). There are good reasons for this election. Firstly, Amazon is a very popular web store. Secondly, its recommender system is a known state-of-the-art one (Schafer, Konstan, & Riedl, 1999). As interactive recommender system we have chosen WebWatcher. As we have seen in Section 3.1, this early system still represents very well and concisely these kind of recommender systems. Note that both recommender systems are accessed by web browsers, which is a general pattern of the field.

We will use Amazon as our first example. First of all, we look at the system from the user's point of view. Once we have our web browser open, we decide to get into this web store. In the first place, this action brings forth a new *session* $s(u)$. After this, whenever we are shopping around, we are provoking *events* into the system. However, it is not until we reach the hyperlink called "My store" that we provoke the first *recommendation event* $e \in ev_r(s(u))$. Also, this event provokes the creation of a new *recommendation process*, which finishes once recommendations have been displayed (see Fig. 7).

Amazon has a large number of items I available (books, music, etc.). However, only a few of them will be shown to us. To this end, *the filter subsystem* is in charge of creating a *recommendation window* $rw(e)$ with those items that are expected to be useful/interesting. In the case of Amazon, to a certain degree, a *collaborative filtering* mechanism is used (Schafer, Konstan, & Riedl, 2000). In fact, after the recommendations have been displayed, we are invited to rate the *recommended items* that we have on our screen. This fact will allow to know the *usefulness/interest* of the item for improving the algorithm and also for evaluation reasons.

However, before the actions of *the guide subsystem*, no recommendation could have been displayed on my screen. Actually, as we saw in Section 3.3, the guide subsystem is in charge of choosing an envelope to wrap every item present in the recommendation window, and it is also in charge of displaying the recommendations. In this case each envelope is like the one which appears in Fig. 8. A click on one of the several buttons present in this envelope will indicate to the guide subsystem that the recommendation has been *attended* and *followed*.

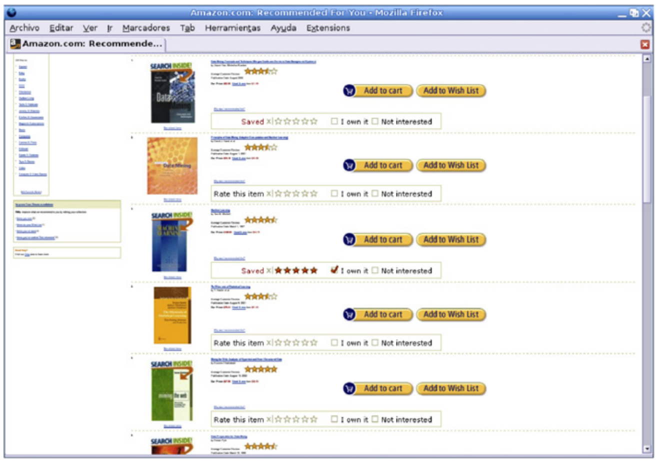

Amazon.com: Recommended For You - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Tab Herramientas Ayuda Extensions
Amazon.com: Recommende...
Add to cart
Add to Wish List
Saved
Rate this item
I own it
Not interested

Fig. 7. Recommendations displayed by Amazon (non-interactive recommender) after a recommendation event has been provoked. Notice the difference between the recommended items (the books themselves) and the envelopes to present the items (the icon of the book, the hyperlinks and the buttons).

Some features characterize this recommender system as a *non-interactive* one. Firstly, the envelope is almost the same for each item and user (see Figs. 7 and 8). Secondly, to a certain degree, the policies of *followed* recommendations are not strongly taken into account so that the guide subsystem could widely adapt the *envelopes* to each different user. Thirdly, as one of the consequences of the last point, all the items present in the recommendation window are to be displayed, regardless of its *follow-ability*.

Due to this fact, always in the context of the traditional evaluation of recommender systems, we could say that the *guide* subsystem of a *non-interactive* system is considered *perfect*, in the sense that currently it is considered that no guide intervenes in the recommendation process. Thus, following the Definition 2 (see Section 3.3), (i) the recommender recommends every item present in $rw(e)$ and (ii) every recommendation is (abstractly) assumed as being followed. By means of this idea, during the traditional evaluation of non-interactive recommender systems in our framework, we will only need to focus on the *usefulness* of the recommended items, because all of the available recommendations are always *displayed* and *followed*.

WebWatcher is a recommender system which assists us as we follow hyperlinks forward through the web. Concretely, its job consists of highlighting (as recommendations) some of the hyperlinks present in each visited page. Once we have started using WebWatcher, it keeps an open session $s(u)$ for us. Afterwards, each time we click on any hyperlink of the web page that WebWatcher is showing, we are provoking a new *recommendation event* $e \in ev_r(s(u))$. As a result, a new *recommendation process* begins.

The number of items $|I|$ to be recommended by WebWatcher is extremely large. Actually, they are the set of hyperlinks of every page on the Internet. In spite of this, the *filter subsystem* of WebWatcher is extremely simple, on the grounds that each recommendation window $rw(e)$ only consists of those hyperlinks (items) present in the web page accessed by the user at that precise *recommendation event* e.

In fact, all the important matter of WebWatcher can be found in the *guide subsystem*. This subsystem is in charge of choosing the most *follow-able* items from the recommendation window and wrapping them with an envelope which distinguishes the recommendations (hyperlinks surrounded by "eyes") from the rest of the hyperlinks of the page (see

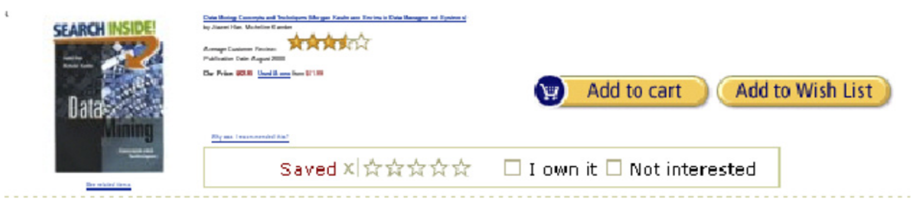
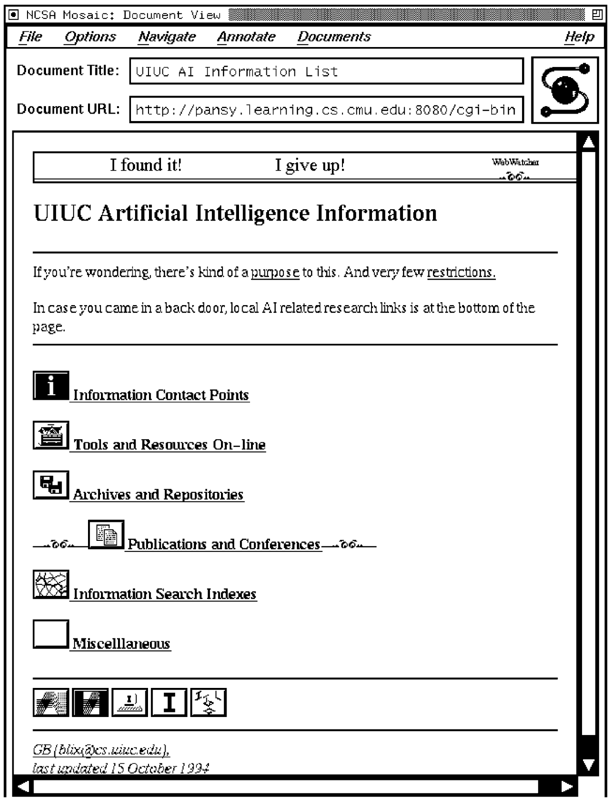


Fig. 8. Example of one of the envelopes that Amazon uses for each recommendation. The item is a book in this particular case.

Fig. 9). Notice that the recommender system only displays *enveloped hyperlinks* (or recommendations). The rest of hyperlinks are displayed by other parts of the application which are not in charge of recommending at all. In other words, following the terms of our framework, the unwrapped hyperlinks are not shown by the recommender system of WebWatcher. Instead, they are shown by another part of WebWatcher. To summarize, the mission of the guide subsystem in WebWatcher consists of: (a) taking the *set of hyperlinks* that belong to the page to be shown (that belong to the recommendation window $rw(e)$) and (b) wrapping them in one of the next two envelopes: "recommend it" or "do not recommend it".

Some features characterize this recommender system as an *interactive* one. Firstly, almost all the work of the recommender system falls on the side of the *guide subsystem*, leaving the *filter subsystem* as an abstract entity. Secondly, the *guide subsystem* applies two extremely different envelopes to each item present in the recommendation window, either "do recommend" or "do not recommend". Thirdly, the sequence of recommendations followed or not followed is totally taken into account for the improvement of the guide subsystem.



UIUC AI Information List

http://pansy.learning.cs.cmu.edu:8080/cgi-bin

UIUC Artificial Intelligence Information

If you're wondering, there's kind of a purpose to this. And very few restrictions.

In case you came in a back door, local AI related research links is at the bottom of the page.

Information Contact Points

Tools and Resources On-line

Archives and Repositories

Publications and Conferences

Information Search Indexes

Miscelllaneous

GB (blix@cs.uiuc.edu),
last updated 15 October 1994

Fig. 9. Recommendations displayed by WebWatcher (interactive recommender). Also, see the eyes that WebWatcher uses to distinguish the recommendation from the rest of hyperlinks. To put it in another way, the recommender system part of WebWatcher wraps the recommended hyperlink in an envelope patterned as the space between the "eyes" (figure extracted from Armstrong et al. (1995)).

Notice that, always in the context of traditional evaluation of recommender systems, and in the context of *interactive* recommender systems, a followed recommendation is always assumed useful. In fact, in this type of recommender systems, the meaning of the terms *followed* and *useful* collapse into one. Also, due to this fact, and always in the context of our framework, we will (abstractly) say that the *filter* subsystem of an interactive recommender system is *perfect*.

In the next section we will reformulate the traditional metrics so as to use them into this framework.

4. Measures for recommender systems

Once we have obtained the formal structures, we can safely start building on top of them. In this section, we will study several metrics that can be applied into this framework. We begin by refactoring the traditional metrics so that we can apply them over the framework developed in the previous section. Afterwards, we will develop a new metric. We will end the section by studying the advantages of this new metric over the traditional ones.

4.1. Traditional metrics

First of all, we must notice the difference of considering both interactive and non-interactive subsystems while constructing the metric. Actually, it is not a usual thing, and present metrics have not been constructed with this idea in mind. Thus, as we introduced in Section 3.4, whenever we are working with a non-interactive recommender system, we will say that its *guide* subsystem is *perfect*, so that we can avoid this system from consideration. In a similar way, we will avoid the *filter* subsystem in an *interactive* recommender system by defining it as perfect as well.

Firstly, take the case of a non-interactive recommender system. Due to the fact that the guide is perfect, we are free to focus on the usefulness/interest of the recommended items. Then, we can arrive at a confusion matrix like the one introduced in Section 2.3 in Table 1.

On the other hand, an interactive recommender system will consider that following any recommendation is totally successful, despite the final usefulness/interest of the recommended item present in the recommendation. However, in this specific case, it is harder to use a confusion matrix as the one in Table 1. Indeed, it is difficult (if not impossible) to detect the good or bad behavior of a recommendation never displayed. Thus, these recommender systems usually consider only the accuracy/precision of the displayed recommendations.

For the above mentioned reasons, we must reformulate the SDMC metrics (see Section 2) for each type of recommender system in terms of the formal elements of the new framework.

In the case of a non-interactive recommender system, the perfect guide will display every recommended item present in the recommendation window $rw(e)$. Therefore, for each recommendation event e, we will obtain a confusion matrix as the one in Table 2. Note the similarity with Table 1. Consequently, for non-interactive recommender systems, we arrive at the next traditional SDMC metrics $recall_e$, (10), $precision_e$, (11), and $accuracy_e$, (12).

$$recall_e = \frac{a_e}{a_e + c_e} \tag{10}$$

$$precision_e = \frac{a_e}{a_e + b_e} \tag{11}$$

$$accuracy_e = \frac{a_e + d_e}{a_e + b_e + c_e + d_e} \tag{12}$$

Nevertheless, keeping a lot of values, one for each recommendation event, could not be acceptable as a "normalized" metric. Then, some kind of average is required. In the case of *accuracy* there exist multiple ways of computing its average: mean, mean squared root, root mean squared error, etc. With regard to the average of the IR metrics, there are two usual ways (van Rijsbergen, 1979): the *macro-evaluation* and the *micro-evaluation*. The kind of average chosen depends on the system. Nevertheless, in order not to lose the objectives of this paper, we will not go further into these studies. On the other hand, in other cases, it would be more useful to dispose of a large number of values so that we can draw smooth graphs (in principle, the higher the number of values, the smoother the graph). This is the case when IR metrics are to draw precision/recall or ROC graphs rather than using their averaged quantities.

Now, we will jump to the case of an interactive system. Embodied into this recommender system we will find a perfect filter which only presents useful/interesting items to the guide. As a result, only the follow-ability of the recommendations is to be considered. Thus, as we saw in Section 3.3, for each recommendation event e and item i, the guide is in charge of choosing the envelope for the (always useful) item. Note that the most important property of the envelope is always its display-ability. In other words, the most important decision the guide must determine is whether the recommendation will or will not be displayed to the user. Therefore, we may see it as if the guide had two main decisions: (i) to display the item as a recommendation, (ii) not to display the recommendation. Next, the user has two possibilities: (i) to follow the recommendation, (ii) not to follow it. All this facts can be summarized in a confusion matrix as the one in Table 3.

Now, if we compare the Table 3 with the Table 2 we will observe some similarities. However, in this case, a noticeable fact appears: $c = 0$. In other words, a never displayed recommendation can never be followed. Due to this singular fact, it is usual to measure these recommenders taking into account just the first row of the table. This leads us to a metric as the one in (13). Notice the similarity of this equation with (11). In fact, it is not a surprise that in the context of interactive recommender systems "accuracy" and "precision" terms are considered almost synonyms.

$$accuracy = \frac{a}{a + b} \tag{13}$$

Observe that even when interactive and non-interactive systems can use similar metrics for their evaluation, the terms involved in their equations are completely different. Particularly, as an example, a quantity as c will never be the same for interactive and non-interactive recommender systems.

In conclusion, in this section we have reformulated the SDMC traditional metrics in the terms of the framework introduced in Section 3. Finally, different confusion matrixes appear, and consequently, different metrics must be applied to these two types of recommender systems. In the next section, we will develop a new metric which takes into account these differences and will allow to measure all types of recommender systems by the same metric.

4.2. Performance quantification for recommender systems

In this section we will develop a new metric to measure the quantity of the objective achieved by recommender systems. We will call this metric *performance* $\mathcal{P}$ of recommender systems. One of the main reasons is that this name expresses an evaluation of the *global* execution of the recommender system. In other words, the nearness to its main goal.

As we have mentioned repeatedly, the main goal of a recommender system consists of *guiding* the user to *useful* items. Also, as we said in Section 3, we confirm that the goal is achieved in every single recommendation whenever it complies with the next two rules at the same time: (i) the recommendation is *followed*, (ii) the item *enveloped* into the recommendation is *useful/interesting*.

With all these previous ideas in mind, we are ready to develop the new metric $\mathcal{P}$. To this end, we will take similar steps as those we took to reformulate the traditional metrics in the previous section. In fact, we will base the new measure on a similar confusion matrix as those in Tables 2 and 3. However, in this case, the quantities of this matrix

Table 2
Confusion matrix for *non-interactive* recommender systems

	Useful/interesting	Useless/uninteresting
Belong to $rw(e)$	a_e	b_e
Out of $rw(e)$	c_e	d_e

Notice that there must be a new confusion matrix for each recommendation event e, because the filter creates a new $rw(e)$ at each event.

Table 3
Confusion matrix for *interactive* recommender systems

	Followed	Not followed
Displayed	a	b
Not displayed	c	d

Note in this case we are *not* evaluating the accuracy/performance of the elements present in each recommendation window $rw(e)$, therefore we can consider the whole number of recommendations (displayed or not) in a set of sessions S.

will be primarily computed by counting the number of *followed* and *useful* recommendations that the recommender system has had to consider in a set of sessions S. This confusion matrix can be seen in Table 4. Note the extreme similarity with the confusion matrix of Table 3. This fact looks natural if we bear in mind that the guide is the external face of the recommender system.

As a first try, a good metric could consist of just counting the number of *followed* and *useful* recommendations that the recommender system has displayed. Why not, at a first sight it seems clear that the higher this number, the better the recommender. However, the problem appears whenever we try to compare between recommender systems that have recommended during different periods of time. In other words, we need to compare recommender systems regardless of the number of recommendation events they have experienced. To this end, we define the metric $\mathscr{P}(S)$, (14), as the quantification of the final *performance* of a recommender system over a set of sessions S.

$$\mathscr{P}(S) = \frac{\alpha}{|ev_r(S)|} \tag{14}$$

Note that this metric is not only an average, but an estimator of the number of *good* recommendations we *expect* to find in each one of the following recommendation events. Obviously, the higher the number of the recommendation events $|ev_r(S)|$ along the whole set of sessions S considered, the better will be the estimation. In addition to this single but significative advantage, we will enumerate several others.

(1) This metric has been directly derived from the general objective of recommender systems. As a result, this metric is a *SDMC* metric (see Section 2.6). In addition, there is no assumption about the source of knowledge about the usefulness/interest of the items enveloped in any recommendation. Needless to say, this usefulness/interest can still be derived from previous ratings over the item, however it is not a demand anymore.
(2) Unlike traditional metrics for the evaluation of *non-interactive* recommender systems, $\mathscr{P}(S)$ do not require the knowledge of the usefulness of any other item than those that belong to the *followed* recommendation. This is a very important issue. For instance, imagine an e-commerce system whose main source of knowledge about the usefulness/interest of its items come from the transactions they generate for a particular user. In this case, the usefulness/interest of most of the items present in the system will be unknown. Even more, with high probability the usefulness/interest of many items into this system will change over time. In fact, there are several collaborative filtering based recommender systems whose ratings can evolve over time (Cosley et al., 2003). Consequently, some recommender systems require to detect and to take into account the usefulness/interest of the item just in the precise instant in which the recommendation has been displayed and followed by the user.
(3) Unlike traditional metrics, $\mathscr{P}(S)$ does not require any additional computation of averages.
(4) Unlike traditional metrics, $\mathscr{P}(S)$ does not have to be "translated" whenever we evaluate different recommender systems as an *interactive* or a *non-interactive* recommender system. Moreover, this metric, like the framework introduced in Section 3, is applicable regardless of the recommender system type, because they are as general as the main objective of the recommender systems. As a result, it allows, among other things, to evaluate the whole range of recommender systems by means of a single framework/metric.
(5) As a consequence of the last point, $\mathscr{P}(S)$ gathers the performance quantification of both internal subsystems (the guide and the filter) together in a single value. Naturally, unlike traditional metrics, this fact occurs even if the other subsystem is not perfect.

Table 4
Confusion matrix for *general* recommender systems

	Followed and useful/interesting	Rest of the cases
Displayed	α	β
Not displayed	–	δ

In this case, note that there is only a single confusion matrix for the whole set of sessions S. Also, notice that we do not consider the quantity $c = 0$ anymore. As a mathematical detail, notice that $\alpha + \beta + \delta = \sum_{e \in ev_r(S)} |rw(e)|$.

Finally, someone could claim that $\mathscr{P}(S)$ does not treat usual problems that traditional metrics do. As an example, with $\mathscr{P}(S)$, it seems that if the guide displayed (in one session event) a million recommendations of which the user followed 10 (which were indeed useful), this is as good as if the guide displayed 10 recommendations all of which the user followed (and all of which were useful).

To clarify this question note that $\mathscr{P}(S)$, unlike traditional ones, is focused on measuring the *final objective* of any recommender system (of course, those recommenders placed into our framework limits). Therefore, the metric cannot measure *how* this objective must be achieved, but only if it is really achieved. Up to a point, displaying a million of recommendations could be fine, depending on the context and the kind of application considered. In spite of the latter, we will discuss about this topic in the next section. All things considered, we claim that $\mathscr{P}(S)$ does not get involved in the details of *how*, but just focus on *what* must be obtained by a general recommender system.

5. Conclusions and future work

Evaluation of recommender systems is a challenging task due to the many possible scenarios in which such systems may be deployed. Traditional evaluation metrics for recommenders are biased towards the particular techniques used to select the items to be shown, and they do not take into account the main goal of any recommender: to guide the user to useful/interesting objects.

The metric $\mathscr{P}$ presented in this paper can be considered an step forward towards this direction, since it considers the follow-ability of the recommendations, apart from the usefulness/interest of the recommended items.

In fact, to provide a common ground for evaluating recommender systems and taking into account the main goal of any recommender, we have presented a new general framework that considers each recommender as being composed of a guide subsystem and a filter subsystem. While the filter subsystem is in charge of selecting useful/interesting items, the guide subsystem is the responsible for wrapping and showing only those items that are most likely to be followed by the user. To illustrate both subsystems, we have shown how two very different and well-known recommenders can be defined in terms of our proposed framework.

We are now working on obtaining metrics derived from the metric $\mathscr{P}$ for evaluating each one of the subsystems individually. It would provide us with a knowledge of the recommender system inside, which we expect to be very useful for future improvements.

Related to the latter, although obvious and very often neglected, it must be noticed that the function of the guide subsystem is essential for the good performance of a recommender system. For instance, when the user obtains a lot of items on the screen, she can feel overwhelmed in spite of the usefulness of them (this is usually called *intrusion*). Thus, the quality of the *guide* subsystem is closely related to the intrusion cost of the act of recommending (Hernandez del Olmo, Gaudioso, & Boticario, 2005). This point appears really promising for obtaining better guides in future recommender systems.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward 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), 734–749.

http://www.amazon.com, January 2007.

Ardissono, L., & Goy, A. (2000). Tailoring the interaction with users in web stores. *User Modeling and User-Adapted Interaction, 10*(4), 251–303.

Ardissono, L., Goy, A., Petrone, G., Segnan, M., & Torasso, P. (2003). Intrigue: Personalized recommendation of tourist attractions for desktop and handset devices. *Applied Artificial Intelligence, 8–9*(17), 687–714.

Armstrong, R., Freitag, D., Joachims, T., & Mitchell, T. (1995). Webwatcher: A learning apprentice for the world wide web. In *AAAI spring symposium on information gathering* (pp. 6–12).

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval.* Essex: Addison Wesley.

Balabanovic, M., & Shoham, Y. (1997). Fab: Content-based collaborative recommendation. *Communications of the ACM, 40*(3), 66–72.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. In *ICML'98: Proceedings of the fifteenth international conference on machine learning* (pp. 46–54). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Billsus, D., & Pazzani, M. J. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction, 10*, 147–180.

Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Uncertainty in artificial intelligence, proceedings of the fourteenth conference* (pp. 43–52). Morgan Kaufman.

Burke, R. (2002). Hybrid recommender systems. *User Modeling and User-Adapted Interaction, 12*(4), 331–370.

Cosley, D., Shyong, K., Istvan, A., Konstan, A., & Riedl, J. (2003). Is seeing believing? How recommender interfaces affect users' opinions. In *Proceedings of CHI 2003, Florida, USA*.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on machine learning (ICML)*.

Fisher, M. J., Fieldsend, J. E., & Everson, R. M. (2004). Precision and recall optimisation for information access tasks. In *First workshop on roc analysis in AI. European conference on artificial intelligence (ECAI'2004), Valencia, Spain, August*.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133–151.

Haley, J. A., & Mcneil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology, 143*, 29–36.

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., & Kadie, C. (2001). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research, 1*, 49–75.

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *SIGIR'99: Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 230–237). New York, NY, USA: ACM Press.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems, 22*(1), 5–53.

Hernandez del Olmo, F., Gaudioso, E., & Boticario, J. G. (2005). Evaluating the intrusion cost of recommending in recommender systems. In *User modeling 2005, 10th international conference* (pp. 342–346).

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of ACM CHI'95 conference on human factors in computing systems* (pp. 194–201). ACM Press.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 07*(1), 76–80.

Mladenic, D. (1996). Personal webwatcher: Implementation and design. Technical Report IJS-DP-7472, Department of Intelligent Systems, J. Stefan Institute, Slovenia.

Pazzani, M. J., Muramatsu, J., & Billsus, D. (1996). Syskill&Webert: Identifying interesting web sites. In *Proceedings of the national conference on artificial intelligence* (Vol. 1, pp. 54–61).

Pazzani, M. J., Muramatsu, J., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning, 27*, 313–331.

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM.*

Rogers, S., Flechter, C., & Langley, P. (1999). An adaptive interactive agent for route advice. In *Proceedings of the third international conference on autonomous agents (Agents'99)* (pp. 198–205). Seattle, WA, USA: ACM Press.

Schafer, J., Konstan, J., & Riedl, J. (1999). Recommender systems in e-commerce. In EC'99: Proceedings of the First ACM Conference on Electronic Commerce, Seattle, November.

Schafer, J., Konstan, J., & Riedl, J. (2000). Electronic commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 115–152.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating 'word of mouth'. In *CHI'95: Proceedings of the conference of human factors in computing systems*. ACM Press.

Sheskin, D. (2003). *The handbook of parametric and nonparametric statistical procedures*. CRC Press.

van Rijsbergen, C. J. (1979). *Information retrieval*. London: Butterworth.

Yao, Y. Y. (1995). Measuring retrieval effectiveness based on user preference of documents. *Journal of American Society in Information Sciences, 46*(2), 133–145.