# State of the Art Recommender System

**Chapter** · April 2009
DOI: 10.4018/978-1-60566-306-7.ch001



**4 authors**, including:

Laurent Candillier
Indépendant
**21** PUBLICATIONS **313** CITATIONS
SEE PROFILE

Françoise Fessant
Orange Labs
**53** PUBLICATIONS **407** CITATIONS
SEE PROFILE

Frank Meyer
France Télécom
**21** PUBLICATIONS **256** CITATIONS
SEE PROFILE

# State-of-the-Art Recommender Systems

Laurent Candillier, Kris Jack, Françoise Fessant, Frank Meyer
France Telecom R&D Lannion, France
{name.surname}@orange-ftgroup.com

## *Abstract*

The aim of *Recommender Systems* is to help users to find items that they should appreciate from huge catalogues. In that field, *collaborative filtering* approaches can be distinguished from *content-based* ones. The former is based on a set of user ratings on items while the latter uses item content descriptions and user thematic profiles.

While collaborative filtering systems often result in better predictive performance, content-based filtering offers solutions to the limits of collaborative filtering, as well as a natural way to interact with the users. These complementary approaches thus motivate the design of hybrid systems.

In this chapter, state-of-the-art recommender methods are presented. Then various approaches are compared using cross-validation on two real film rating datasets called MovieLens and Netflix. Results of tests conducted with real users are also presented.

**Keywords:** recommender systems, collaborative filtering, content-based filtering, user modelling, profiling, personalisation, user interaction.

## *1 Introduction*

There has been a growth in interest in *Recommender Systems* in the last two decades (Adomavicius & Tuzhilin, 2005), since the appearance of the first papers on this subject in the mid-1990s (Resnick et al., 1994). The aim of such systems is to help *users* to find *items* that they should appreciate from huge catalogues.

Items can be of any type, such as films, music, books, web pages, online news, jokes and restaurants. In this chapter, focus is made on film items. The goal, therefore, of recommender systems, in this context, is to help users to find films of interest, based on some information about their historical preferences. Three types of recommender systems are commonly proposed:

1. *collaborative filtering*;
2. *content-based filtering*;
3. and *hybrid filtering*.

In collaborative filtering, the input to the system is a set of user ratings on items. Users can be compared based upon their shared appreciation of items, creating the notion of user neighbourhoods. Similarly, items can be compared based upon the shared appreciation of users, rendering the notion of item neighbourhoods. The item

rating for a given user can then be predicted based upon the ratings given in her user neighbourhood and the item neighbourhood.

In the case of content-based filtering, however, item content descriptions are used to construct *user thematic profiles* that contain information about user preferences, such as, in the context of films, *"like comedy and dislike war"*. The user's predicted appreciation of a given item is then based on the proximity between the item description and the user profile.

Finally, in the case of hybrid filtering, both types of information, collaborative and content-based, are exploited.

Collaborative filtering techniques are more often implemented than the other two and often result in better predictive performance. In this chapter, many different options in the three general approaches will be presented and compared:

1. *user-based* approaches (Resnick et al., 1994) associate a set of nearest neighbours with each user and then predict the user's rating for unscored items using the ratings given by the neighbours on that item;

2. *item-based* approaches (Sarwar et al., 2001) associate an item with a set of nearest neighbours, and then predict the user's rating for an item using the ratings given by the user on the nearest neighbours of the target item;

3. *model-based* approaches, and more specifically those based on *clustering* (Ungar & Foster, 1998), tend to be more scalable, by constructing a set of user groups or item groups, and then predicting a user's rating for an item using the mean rating given by the group members.

The definition of similarity between users and items is a key problem in each approach. While traditional similarity measures can be used, bespoke ones, which are tailored to type of data that is typically available (i.e. very sparse), tend to lead to better results.

The input to content-based recommender systems is a set of item content descriptions, such as the genre, director and actors, in the context of films. Such techniques can also be divided into three general approaches:

1. *profiling* information can be obtained from users *explicitly*, through questionnaires about their preferences for the item descriptions;

2. *user profiles* may be built *implicitly* from user preferences for items, by searching for commonalities in liked and disliked item descriptions;

3. *user models* may be learned *implicitly* by an automatic learning method, using item descriptions as input to a supervised learning algorithm, and producing user appreciations of items as output.

User profiles are often represented as vectors of weights on item descriptions. Any other user model may be considered if an automatic learning method is used. If rule

induction algorithms are used, user models may be of the type *"if genre is action and actor is Stallone then film is liked"*. Finally, items that have a high degree of proximity to a given user's preferences would be recommended.

To be efficient, content-based approaches need rich and complete descriptions of items and well-constructed user profiles. This is the main limitation of such systems. Since well-structured item descriptions are hard to come by in many domains, such approaches have mainly been applied in those where items are described by textual information, that can be parsed automatically, such as documents, web sites and Usenet news messages (Pazzani & Billsus, 1997; Mooney & Roy, 1999).

Besides, content-based approaches can also suffer from *overspecialisation* (Zhang et al., 2002). That is, they often recommend items with similar content to that of the items already considered, which can lead to a lack of originality. On the other hand, privacy issues (Lam et al., 2006), such as users who do not want to share their preferences with others, are avoided.

Collaborative filtering systems, however, do not require such difficult to come by, well-structured item descriptions. Instead, they are based on users' preferences for items, which can carry a more general meaning than is contained in an item description. Indeed, viewers generally select a film to watch based upon more elements than only its genre, director and actors.

On the other hand, collaborative filtering systems suffer from the *cold start* problem: they cannot make predictions on items that have not yet been rated by any user. Content-based approaches are able to handle such situations. These complementary approaches thus motivate the design of hybrid systems.

Collaborative filtering seems to be more suitable as the core method of the recommender system while content-based filtering offers solutions to the limits of collaborative filtering, as well as a natural way to interact with the users. Indeed, users should be allowed to exert control over the system, thus building a meaningful relationship, and leading to psychological benefits such as the increase of trust in recommendations.

This brings naturally to the issue of evaluating the performance of a recommender system. Two methods can be used (Herlocker et al., 2004):

1. error rate evaluation using *cross-validation*;
2. *user satisfaction* evaluation.

For the first approach, many measures can be used to compare the results of different recommender systems. The most widely used ones are:

1. *Mean Absolute Error* (MAE);
2. *Root Mean Squared Error* (RMSE);
3. *Precision* measures.

The first two measures evaluate the capability of a method to predict if a user will like or dislike an item, whereas the third measure evaluates its capacity to order a list of

items based on user tastes. These measures thus carry different meanings (McNee et al., 2006). In the first two cases, the method needs to be able to predict dislike, but there is no need to order items. In the last case, however, the method only focuses on items that users will like and the order in which these items are ranked is important.

Beyond the importance of the predictive performance of recommender systems, other elements may be taken into consideration in their evaluation. The *scalability* of the proposed system is for example an important characteristic that needs to be taken into account. The *coverage* of a method, that is the proportion of recommendations it can provide, can also be considered. Finally, the system's ability to provide a level of *confidence* in a recommendation (Basu et al., 1998) and to *explain* why a recommendation was made (Herlocker et al., 2000; Bilgic, 2004) can be used to define its potential interest to the user.

Evaluating a recommender system based on real users' opinions is also important because, in many cases, recommending the set of items that maximise their predicted ratings does not necessarily lead to user satisfaction. For instance, users may estimate that such recommendations lack originality, or they may think that the proposed list of recommendations is not varied enough (Ziegler et al., 2005). Users may also want to have some control over the system, rather than having little or no direct influence in the results.

The experiments presented in this chapter use two real rating datasets that are publicly available: *MovieLens* (www.grouplens.org) and *Netflix* (www.netflixprize.com). The first dataset contains 1,000,209 film ratings collected from 6,040 users on 3,706 films and the second contains 100,480,507 film ratings collected from 480,189 users on 17,770 films. These datasets are independent of one another.

To summarise, this chapter, on the state-of-the-art in recommender systems, will be structured as follows: an overview of the principal approaches for collaborative filtering is presented in section 2, alongside a study of similarity measures for comparing users and items; content-based and hybrid filtering methods are considered in section 3 for tackling the important issue of providing an efficient interaction with users; the results of extensive experiments are reported in section 4 in order to compare various recommender systems, using cross-validation on the MovieLens and Netflix datasets, as well as tests with real users; finally, section 5 concludes the chapter and topics for future research are considered.

## *2 Collaborative filtering*

Let $U$ be a set of $N$ users, $I$ a set of $M$ items, and $R$ a set of ratings $r_{ui}$ of users $u \in U$ on items $i \in I$. $S_u \subseteq I$ stands for the set of items that user $u$ has rated. Let $\min_r$ and $Max_r$ be the minimum and maximum ratings respectively. In the MovieLens and Netflix datasets for example, ratings are integers ranging from 1 (meaning dislike) to 5 (meaning like).

The goal of collaborative filtering approaches is then to be able to predict the rating $p_{ai}$ of a user $a$ on an item $i$. User $a$ is supposed to be *active*, meaning that she has

already rated some items, so $S_a \neq \varnothing$. The item to be predicted is not yet known to the user, making $i \notin S_a$.

## User-based approaches

For user-based approaches (Resnick et al., 1994; Shardanand & Maes, 1995), the prediction of a user rating on an item is based on the ratings, on that item, of the nearest neighbours. So a similarity measure between users needs to be defined before a set of nearest neighbours is selected. Also, a method for combining the ratings of those neighbours on the target item needs to be chosen.

The way in which the similarity between users is computed is discussed in the last subsection. For now, let $sim(a,u)$ be the similarity between users $a$ and $u$. The number of neighbours considered is often set by a system parameter, denoted by $K$. So the set of neighbours of a given user $a$, denoted by $T_a$, is made up of the $K$ users that maximise their similarity to user $a$.

A possible way to predict the rating of user $a$ on item $i$ is then to use the weighted sum of the ratings of the nearest neighbours $u \in T_a$ that have already rated item $i$:

$$p_{ai} = \frac{\sum_{\{u \in T_a | i \in S_u\}} sim(a,u) \times r_{ui}}{\sum_{\{u \in T_a | i \in S_u\}} \left| sim(a,u) \right|}$$

**Equation 1 : user-based prediction using weighted sum.**

In order to take into account the difference in use of the rating scale by different users, predictions based on deviations from the mean ratings have been proposed. $p_{ai}$ can be computed from the sum of the user's mean rating and the weighted sum of deviations from their mean rating of the neighbours that have rated item $i$:

$$p_{ai} = \overline{r_a} + \frac{\sum_{\{u \in T_a | i \in S_u\}} sim(a,u) \times (r_{ui} - \overline{r_u})}{\sum_{\{u \in T_a | i \in S_u\}} \left| sim(a,u) \right|}$$

**Equation 2 : user-based prediction using deviations from the mean.**

$\overline{r_u}$ represents the mean rating of user $u$:

$$\overline{r_u} = \frac{\sum_{\{i \in S_u\}} r_{ui}}{\left| S_u \right|}$$

**Equation 3 : mean user rating.**

Indeed, let's suppose that a user rates 4 a film that she likes and 1 to a film that she dislikes while another user rates 5 a film she likes and 2 a film that she dislikes. By using deviations from the mean rating, the individual user's semantics, with respect to their appreciation of the film, is better accounted for.

The time complexity of user-based approaches is $O(N^2 \times M \times K)$ for the neighbourhood model construction and $O(K)$ for one rating prediction. The space complexity is $O(N \times K)$.

## Item-based approaches

Recently, there has been a rising interest in the use of item-based approaches (Sarwar et al., 2001; Karypis, 2001; Linden et al., 2003; Deshpande & Karypis, 2004). Given a similarity measure between items, such approaches first define item neighbourhoods. The predicted rating for a user on an item is then derived by using the ratings of the user on the neighbours of the target item.

The possible choices of the similarity measure $sim(i, j)$ defined between items $i$ and $j$ are discussed later. Then, as for user-based approaches, the item neighbourhood size $K$ is a system parameter that needs to be defined. Given $T_i$, the neighbourhood of item $i$, two ways for predicting new user ratings can be considered:

1. using a weighted sum:

$$p_{ai} = \frac{\sum_{\{j \in S_a \cap T_i\}} sim(i, j) \times r_{aj}}{\sum_{\{j \in S_a \cap T_i\}} |sim(i, j)|}$$

**Equation 4 : item-based prediction using weighted sum.**

2. using a weighted sum of deviations from the mean item ratings:

$$p_{ai} = \bar{r}_i + \frac{\sum_{\{j \in S_a \cap T_i\}} sim(i, j) \times (r_{aj} - \bar{r}_j)}{\sum_{\{j \in S_a \cap T_i\}} |sim(i, j)|}$$

**Equation 5 : item-based prediction using deviations from the mean.**

$\bar{r}_i$ is the mean rating on item $i$:

$$\bar{r}_i = \frac{\sum_{\{u \in U | i \in S_u\}} r_{ui}}{|\{u \in U | i \in S_u\}|}$$

**Equation 6 : mean item rating.**

The time complexity of item-based approaches is $O(M^2 \times N \times K)$ for the neighbourhood model construction and $O(K)$ for one rating prediction. The space complexity is $O(M \times K)$.

## Model-based approaches

A quadratic complexity is too high for huge datasets and many real applications need predictions that can be made very quickly. These considerations are the starting points of model-based approaches (Breese et al., 1998). The general idea is to derive off-line a model of the data in order to predict on-line ratings as fast as possible.

The first types of models that have been proposed consist of grouping users using clustering and then predicting a user rating on an item using only the ratings of the users that belong to the same cluster.

Bayesian models have also been proposed to model dependencies between items. The clustering of items has been studied in (Ungar & Foster, 1998; O'Conner & Herlocker, 1999). Also, models based on association rules have been studied in (Sarwar et al., 2000; Lin et al., 2002).

Probabilistic clustering algorithms have also been used in order to allow users to belong, at some level, to different groups (Pennock et al., 2000; Kleinberg & Sandler, 2004). And hierarchies of clusters have been proposed, so that if a given cluster of users does not have an opinion on a particular item, then the super-cluster can be considered (Kelleher & Bridge, 2003).

In such approaches, the number of clusters considered is of key importance. In many cases, different numbers of clusters are tested, and the one that leads to the lowest error rate in cross-validation is kept. Clusters $C_k$ are then generally represented by their centroid $\overrightarrow{\mu_k}$:

$$\mu_{ki} = \frac{\sum_{\{u \in C_k | i \in S_u\}} r_{ui}}{\left|\{u \in C_k | i \in S_u\}\right|}$$

**Equation 7 : centroid value of a cluster.**

Then the predicted rating of a user for an item can be directly derived from the rating of its nearest centroid. If both user and item clustering are used, then the predicted rating is the mean rating, on the item's group members, of the user's group members.

In this chapter, the clustering algorithm considered is *Bisecting K-means* using *Euclidian* distance. *K-means* is the well-known full-space clustering algorithm based on the evolution of $K$ centroids that represent the $K$ clusters to be found, while Bisecting K-means is based on the recursive use of ($K = 2$)-means. At each step, the cluster that maximises its inertia is split.

This algorithm needs to be run many times with random initial solutions in order to avoid local minima. A parameter must therefore be introduced, say $L$, that represents the required number of runs. The time complexity of cluster-based approaches is then $O(N \times M \times K \times L)$ for the learning phase and $O(1)$ for one rating prediction. The space complexity, when both user and item clustering are considered, is $O((N + M) \times K)$.

## Similarity measures

The similarity defined between users or items is crucial in collaborative filtering. The first one proposed in (Resnick et al., 1994) is the *Pearson* correlation. It corresponds to the *Cosine* of deviations from the mean:

$$pearson(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a}) \times (r_{ui} - \overline{r_u})}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a})^2 \sum_{\{i \in S_a \cap S_u\}} (r_{ui} - \overline{r_u})^2}}$$

**Equation 8 : Pearson similarity.**

Simple *Cosine* can also be used:

$$\cos ine(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} r_{ai} \times r_{ui}}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} r_{ai}^2 \sum_{\{i \in S_a \cap S_u\}} r_{ui}^2}}$$

**Equation 9 : Cosine similarity.**

The simple case of *Manhattan* similarity can also be considered:

$$manhat(a,u) = 1 - \frac{1}{Max_r - \min_r} \times \frac{\sum_{\{i \in S_a \cap S_u\}} |r_{ai} - r_{ui}|}{|\{i \in S_a \cap S_u\}|}$$

**Equation 10 : Manhattan similarity.**

For these similarity measures, only the set of attributes in common between two vectors are considered. Thus two vectors may be completely similar even if they only share one appreciation on one attribute.

Such measures have drawbacks. For example, consider the case when one user is a fan of science fiction while another only watches comedies. Furthermore, these users haven't rated any film in common so their similarity is null. Now they both say that they like *"Men In Black"*, a science fiction comedy. These users thus become completely similar according to the previously presented measures, given that their only common reference point was equally rated.

The *Jaccard* similarity, however, doesn't suffer from this limitation since it measures the overlap that two vectors share with their attributes:

$$jaccard(a,u) = \frac{|\{S_a \cap S_u\}|}{|\{S_a \cup S_u\}|}$$

**Equation 11 : Jaccard similarity.**

On the other hand, such a measure doesn't take into account the difference of ratings between the vectors. In this case, if two users watch the same films but have completely opposite opinions on them, then they are considered to be similar anyway according to Jaccard similarity.

When Jaccard is combined with the other similarity measures, a system can benefit from their complementarity. For example, the product of Jaccard with another similarity measure produces a new result. In this case, Jaccard serves as a weight. *wPearson* can thus represent a weighted Pearson measure, produced by the product of Pearson and Jaccard. Similarly, *wCosine* and *wManhattan* denote the combination of Jaccard with Cosine and Manhattan respectively.

The values of Cosine-based similarity measures lie between −1 and 1 while the other similarity values lie between 0 and 1.

## *3 Content-based filtering*

When working in the domain of cinematography, it is possible to collect a rich and relatively complete set of descriptions about items. A film can be described by its genre, nationality, year, the directors responsible for it and the actors who star in it. Such information can be used directly to design recommender systems based on user content profiles. They can also be used to improve collaborative filtering systems, leading to new hybrid recommender systems. Finally, content data can be used to interact with users, in order to provide them with a level of control over the system.

### User profiles

The most widely-used set of content-based approaches consists of building user thematic profiles. Users are associated with a set of preferences that relate to item contents. A profile can be created explicitly by the user or automatically generated based upon her past actions. In this case, a recommendation system attempts to find the items that best match a given user profile.

Preferences indicate a relationship between a given user and data. In recommender system research, a preference must be both machine codable and carry useful information for making recommendations. For example, the monadic preference *"I like Jackie Chan as an actor"* can be coded as a high score for films with this actor. In turn, films with this higher score will be more recommended than films that are not promoted in this way. In addition, dyadic preferences can be asserted such as *"I like comedies more than dramas"*, allowing a wide number of films to be compared.

While these preferences can be used to improve recommendations, they suffer from certain drawbacks, the most important of these being their limited *coverage*. The coverage of a preference is directly related to the coverage of the attribute(s) to which it is applied. An attribute has a high coverage when it appears in many items and a low coverage when it appears in few items. The coverage of the preference *"I like Jackie Chan as an actor"* is extremely low in most film databases. As such, recommenders that rely solely upon content-based preferences often require a large amount of user details before good recommendations can be made.

It is possible, however, to extend the coverage of a preference by employing the notion of similarity to attributes. A preference for one attribute can also be inferred

for all other attributes that are very similar to that attribute. For example, if *Jackie Chan* and *Bruce Lee* are considered to be very similar, then the preference *"I like Jackie Chan as an actor"* can be extended to include *"I like Bruce Lee as an actor"*. This extension, assuming that it does not contradict other given preferences, extends the coverage of the preference.

Several approaches are commonly followed to determine the similarity between attributes. Most traditionally, this falls within the remit of a domain expert who can construct, by hand, a rich ontology of the domain. While this approach remains popular within smaller domains, recommendation systems are often employed in large domains where instantiation by hand is impractical. Alternatively, measures of similarity can be used that exploit the wealth of information present in the internet. One such similarity metric, the *Normalised Google Distance* (Cilibrasi & Vitanyi, 2007), infers similarities between textual terms using their co-occurrence on websites, as found by *Google*. This metric tends to perform well under diverse conditions and, since it employs the internet, is not domain specific.

The Normalised Google Distance between two items $i$ and $j$ is:

$$d(i,j) = \frac{\max\{\log f(i), \log f(j)\} - \log f(i,j)}{\log M - \min\{\log f(i), \log f(j)\}}$$

**Equation 12 : Normalised Google Distance.**

$M$ is the total number of Google pages searched, $f(i)$ and $f(j)$ are the number of hits for $i$ and $j$ respectively, and $f(i,j)$ is the number of hits for the co-occurrence of $i$ and $j$.

The Normalised Google Distance metric has proved useful in finding the similarity between attributes such as actors (Jack & Duclay, 2008). Unfortunately, it is difficult to determine complete similarity matrices for large scale databases due to some restrictions on the use of the Google API.

To evade such restrictions, similarity metrics that directly analyse the available recommendation system database are often preferred. For example, given two actors in a film database, a vector can be constructed for each one that describes their film history. The vector can contain information such as the genre of films in which they have played, the directors with whom they have worked and the actors with whom they have co-starred. A similarity measure such as wCosine can then be used to compare the two actor vectors.

Another set of possible approaches for content-based filtering consists of using a classifier, like *Naive Bayes*, having as input the item descriptions and as output the tastes of a user for a subset of items. The classifier is trained over the set of items already considered by the user. It is then able to predict if a new item will be liked or not by the user, according to its content description (Adomavicius & Tuzhilin, 2005).

These content-based methods are thus able to tackle some limitations of collaborative ones. They are able to provide recommendations on new items for which no rating is

available, and to atypical users having no or little neighbours. They can also handle situations where users do not consider the same items but consider similar items.

However, such approaches can be limited by the difficulty to obtain rich content descriptions. Recommendations may lack of originality because they stay too close to the users current tastes. And overall, items are often appreciated for reasons more general than is contained in their content descriptions. In the case of films for instance, viewers generally select a film to watch based upon more elements than only its genre, director and actors. On the contrary, collaborative methods possess this advantage of providing a *meta* view on the interest and quality of the items. These complementary approaches thus motivate the design of hybrid systems.

## Hybrid filtering

The first direct way to design a hybrid recommender system is to run independently a collaborative and a content-based one, and then combine their predictions using any voting scheme.

In (Balabanovic & Shoham, 1997), the combination is performed by forcing items to be, at the same time, close to the user thematic profile, and highly rated by her neighbours. In (Pazzani, 1999), users are compared according to their content profiles, and the generated similarity measures are then used in a collaborative filtering system.

In (Polcicova et al., 2000; Melville et al., 2002), the rating matrix is enriched with predictions based on the content, and then a collaborative filtering is run. In (Vozalis & Margaritis, 2004), the similarity between items is computed by using their content descriptions as well as their associated rating vectors. Then an item-based collaborative filtering is used.

In (Han & Karypis, 2005), it is proposed to extend the prediction list of a collaborative filtering method to the items whose content is close to the recommended items. Based on the same idea, a content-based similarity between items is used in (Wang et al., 2006) in order to compare users not only according to their shared appreciations for some items, but by considering also their shared appreciations for items whose contents are similar.

A hybrid system can also be designed that follows a content-based filtering strategy and uses the data produced from collaborative filtering to enrich item similarity descriptions. At its core, it is a content-based filtering system that makes use of attribute similarities, similar to a personalised information retrieval system where requests are null (Jack & Duclay, 2007). Items are recommended that are similar to the user's likes but not to their dislikes. The similarities between genres, nationalities and language attributes are defined by hand, and using the wCosine measure for directors and actors. Each film also contains a unique identification attribute. This identifier is compared to the films that have been previously noted by the user. The notion of similarity, for attributes of this type, is that embodied in the collaborative filtering algorithms. The more that a strictly collaborative filtering algorithm would recommend a film, the closer the film is to the user's profile. This type of hybrid system thus treats social data (found through collaborative filtering) as an attribute of

an item, like any other. By weighting the importance of each characteristic, the system can vary from being purely content-based through to purely collaborative.

## User interaction

The quality of recommendations is ultimately judged by the user of the recommendation system. Many recommendation systems suffer from the 'one-visit' problem where users login, use the system once and then never return. Even users who receive good recommendations can quit using a recommendation system because they become frustrated that they cannot express particular preferences or find that the system lacks flexibility and is not user friendly.

One of the most important aspects of user interaction is that of control. The user must feel like they are in control of their recommendations and that they can navigate themselves out of awkward situations. For example, when a system incorrectly learns a user's preferences, the user should be able to correct the error. While algorithms like collaborative filtering can predict a user's tastes well, they cannot take into account attribute-based preferences, for example, the possibility for the user to express that they do not like some genre of films and that they no longer want them to be recommended. Giving such control to the user will not only improve the recommendations but will also improve the user's interaction experience.

Enjoyable explicit preference entry (EPE) interfaces are designed to allow users to explicitly indicate a preference in the system. There are many examples of EPE interfaces, from questionnaire-based entry forms (Miller et al., 2003) to dialogue systems (Krulwich, 1997). Blog recommenders, such as MineKey (www.minekey.com), and website recommenders, such as StumbleUpon (www.stumbleupon.com), often ask users to indicate their preferences with respect to general topic (e.g. sports, hobbies and arts). Similarly, MovieLens asks users to rate films that are presented in a list format. Unfortunately such interfaces are often boring to use. Many systems use interactive data visualisation techniques in order to provide a fun and engaging setting for the user. For example, both Music Plasma (www.musicplasma.com) and Amaznode (amaznode.fladdict.net) have shown how recommendations can be attractively visualised to the user. EPE interfaces can make use of data visualisation techniques in order to guide users into finding attributes that they know and hence help them to find familiar points at which express their preferences (Jack & Duclay, 2008).

Another important issue regarding user interaction concerns the necessary diversification of the recommendations. One possible way proposed in (Ziegler et al., 2005) for recommendation diversification consists of selecting, among the list of items considered as the most appropriate to a user, a subset as diverse as possible. To do that, the item with highest predicted interest is first selected. Then the system chooses the item that optimises a criterion mixing its predicted interest and its difference with the first selected item. And this process is iterated until the desired number of items is reached.

Some information can also be useful for a user as the explanation of why a recommendation is given (Billsus & Pazzani, 2000). Explanations can be based on the

neighbours used for the recommendation in the case of collaborative filtering. They can be based on the elements in the user profile that were contained in the film attributes when a content-based filtering is used. In (Bilgic, 2004), this second type of information has been shown to be more expressive to the users. Explanation is useful to give confidence in the recommendation. It also helps the user to understand how the system works, so that she is then able to provide more relevant information when constructing her profile and interacting with the system.

## *4 Experiments*

Two complementary ways for evaluating recommender systems exist. The first one consists of evaluating the prediction performance of the system using some datasets and cross-validation. The second method consists of asking for their satisfaction to real users testing the system.

### Cross-validation

Experiments are conducted using the MovieLens and Netflix datasets. They are divided into two parts in order to perform cross-validation, training the chosen model using 90% of the datasets and testing it on the last 10%.

Given $T = \{(u,i,r)\}$ the set of (user, item, rating) triplets used for test, the Mean Absolute Error Rate (MAE) and Root Mean Squared Error (RMSE) are used to evaluate the performance of the algorithms:

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{ui} - r|$$

**Equation 13 : Mean Absolute Error.**

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i,r) \in T} (p_{ui} - r)^2}$$

**Equation 14 : Root Mean Squared Error.**

The predicted ratings are rounded when the MAE is reported. First of all, this improves the results. Besides, rounding ratings is natural in practice, since real users generally prefer rating scales based on natural numbers than on real numbers.

Then precision measures specifically designed for the current context can be used. $precision_5$ stands for the proportion of maximum ratings in the test dataset ($Max_r = 5$) that are retrieved as the best predicted ratings. Similarly, $precision_4$ stands for the proportion of test ratings higher than $Max_r - 1 = 4$ that are considered as the best ratings by the given recommender system.

Finally, the times spent to learn the models, and for predictions, are also reported. The computer used for these experiments has 32Go RAM and 64 bits 3.40GHz 2-cores CPU.

Considering only the principal collaborative filtering approaches already leads to a lot of choices and parameters. When implementing a user- or item-based approach, one may choose:

1. a similarity measure: Pearson (Equation 8), Cosine (9), Manhattan (10), Jaccard (11), or the proposed combinations of Jaccard with the others;
2. a neighbourhood size $K$;
3. how to compute predictions: using a weighted sum of rating values (Equations 1 and 4), or using a weighted sum of deviations from the mean (2 and 5).

For model-based approaches, the following parameters need to be defined:

1. clustering users and/or items;
2. the number of clusters.

Prediction scheme based on deviations from the mean has been shown to be more effective in (Candillier et al., 2007). So in the following, only the results using this scheme are reported.

One important aspect of collaborative filtering is the choice of the similarity measure used. The following figures 1 to 3 show the Mean Absolute Error Rate obtained using the presented measures, varying the neighbourhood size $K$ from 10 to the maximum number of possible neighbours, for both user- and item-based approaches, and on both MovieLens and Netflix datasets.

Figure 1 first shows the error rates of item-based approaches depending on the similarity measure used and the neighbourhood size. The optimum is reached with the weighted Pearson similarity and 100 neighbours. All similarity measures are improved when they are weighted with Jaccard, at least when few neighbours are considered. All these weighted similarity measures reach their optimum when 100 neighbours are selected. On the contrary, non-weighted similarity measures need much more neighbours to reach their optimum. 700 neighbours shall be selected when using simple Manhattan similarity, and 1500 when using simple Cosine or simple Pearson.

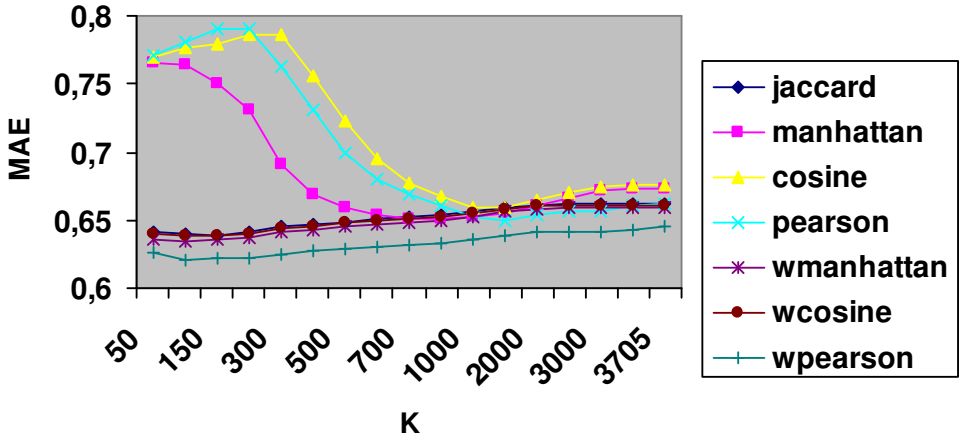


**Figure 1 : Comparing MAE on MovieLens when using item-based approaches with different similarity measures and neighbourhood sizes (K).**

Figures 2 and 3 show that the same conclusions hold when using user-based approaches, as well as when the Netflix dataset is used instead of MovieLens. Weighted Pearson similarity always leads to the best results. Weighting the similarity measures with Jaccard always improves the results. 300 neighbours shall be considered for a user-based approach on MovieLens, and 70 for an item-based approach on Netflix. On the contrary, 2000 to 4000 neighbours need to be selected to reach the minimum error rate with non-weighted similarity measures.

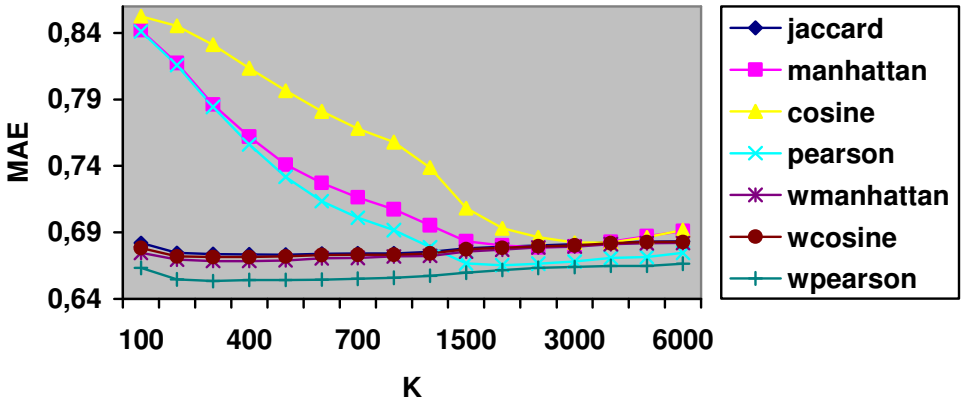


**Figure 2 : Comparing MAE on MovieLens when using user-based approaches with different similarity measures and neighbourhood sizes (K).**

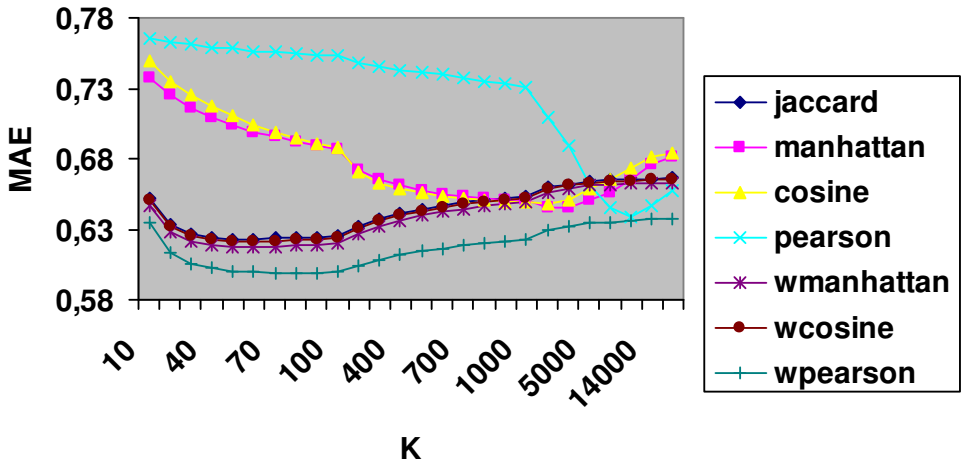


**Figure 3 : Comparing MAE on Netflix when using item-based approaches with different similarity measures and neighbourhood sizes (K).**

The results have been presented for the MAE. They are completely similar for the other performance measures: RMSE and precisions. Beyond the improvement of predictive performance when the proposed weighting scheme is used, another important advantage is that fewer neighbours need to be selected, so that the algorithms also gain in scalability.

In fact, when a non-weighted similarity measure is used, the nearest neighbours do not share many attributes. They often have only one attribute in common. On the contrary, by using Jaccard similarity, the selected neighbours are those that share a maximum number of attributes. Jaccard searches to optimise the number of common attributes between vectors, but this may not be the best solution for nearest neighbour selection since the values of the vectors on the shared attributes may differ. So weighted similarity measures offer an interesting compromise between Jaccard and the other non-weighted measures.

Figures 4 and 5 then show the results obtained by using model-based approaches on both MovieLens and Netflix datasets. The three possible approaches are compared: user clustering, item clustering and double clustering. On MovieLens, user and item clustering behave samely and both outperform the double clustering. Optimal results are reached by using 4 item clusters. On Netflix however, using 70 user clusters leads to the best results.

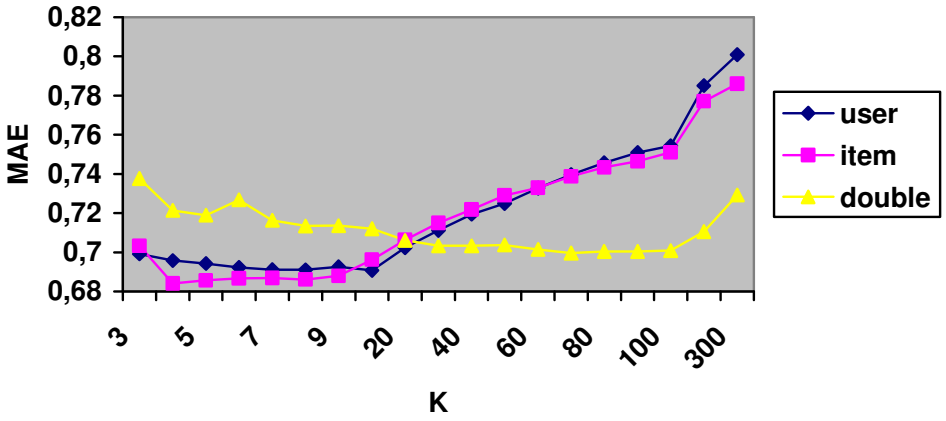


**Figure 4 : Comparing MAE on MovieLens when using model-based approaches with different options and numbers of clusters (K).**

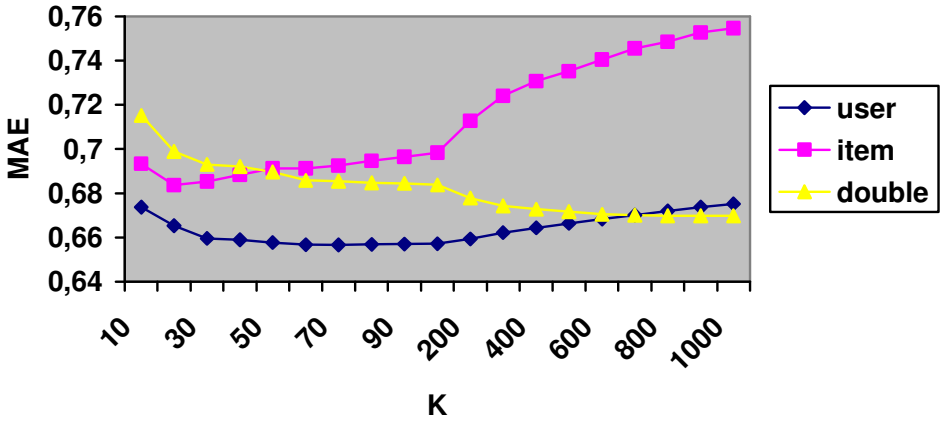


**Figure 5 : Comparing MAE on Netflix when using model-based approaches with different options and numbers of clusters (K).**

Tables 1 and 2 summarise the results of the best of each approach, including learning and prediction times, and precisions. Both user- and item-based approaches reach optimal results when the weighted Pearson similarity is used. On MovieLens, 300 neighbours are selected for the best user-based approach, and 100 for the best item-based one. On Netflix, considering 70 neighbours leads to the lowest error rate. User-based approaches, however, face scalability issues. It is too expensive to compute the entire user-user matrix. So instead, a clustering is first run, and then only the users that belong to the same cluster are considered as potential neighbours. Considering many neighbours improves the results. But a model based on 1,000 neighbours, selected starting from a clustering with 90 clusters, needs nine hours to learn, twenty eight minutes to predict, and 10Go RAM. The best overall results are reached using an item-based approach. It needs two hours and a half to learn the model on Netflix, and one minute to produce ten millions rating predictions. A $precision_5$ of 0.6216 means that 62.16% of the best rated items are captured by the system and proposed to the users.

| | *Parameter* | *Learning time* | *prediction time* | *MAE* | *RMSE* | $precision_5$ | $precision_4$ |
|---|---|---|---|---|---|---|---|
| *model-based* | 4 item clusters | **5 sec.** | **1 sec.** | 0.6841 | 0.9172 | 0.5041 | 0.7550 |
| *user-based* | 300 neighbours | 4 min. | 3 sec. | 0.6533 | 0.8902 | 0.5710 | 0.7810 |
| *item-based* | 100 neighbours | 2 min. | 1 sec. | **0.6213** | **0.8550** | **0.5864** | **0.7915** |

**Table 1: Summary of the best results on MovieLens depending on the type of approach.**

| | *Parameter* | *Learning time* | *prediction time* | *MAE* | *RMSE* | $precision_5$ | $precision_4$ |
|---|---|---|---|---|---|---|---|
| *model-based* | 70 user clusters | **24 min.** | **3 sec.** | 0.6566 | 0.8879 | 0.5777 | 0.7608 |
| *user-based* | 1,000 neighbours | 9 h | 28 min. | 0.6440 | 0.8811 | 0.5902 | 0.7655 |
| *item-based* | 70 neighbours | 2 h 30 | 1 min. | **0.5990** | **0.8436** | **0.6216** | **0.7827** |

**Table 2: Summary of the best results on Netflix depending on the type of approach.**

## User satisfaction

The recommendations from a semantically enriched content-based filtering algorithm, a collaborative filtering algorithm and a hybrid of these two algorithms have been compared in a study by human participants. As well considering the quality of the recommendations produced by the different algorithms, as judged by the participants, the participants' actions and comments are also analysed. In doing so, a number of conclusions can be drawn as to how user needs can be accounted for to produce more natural and satisfying interactions with recommendation systems.

The film database was generated from two sources. Film details (used for content-based filtering) came from an in-house Orange database while user ratings (used for collaborative filtering) came from Netflix. A database of 3,626 films was produced by taking the intersection of films from both data sources. Each film was described by five characteristics: actors, directors, genres, languages and nationalities.

A recommendation system interface was constructed with three primary screens: a login screen, a profile manager and a recommendation screen. The login screen allowed the participant to enter their username and be identified by the system. On first identification, a user profile is created. The participant could manage their profile using the profile manager, which allowed for both monadic and dyadic preferences to be expressed. Monadic preferences could be expressed for any of the five characteristics and films themselves (e.g. *"I like Charlie Chaplin as an actor"* and *"I dislike Scary Movie"*) on a 3-point scale (like, neutral, dislike). Dyadic preferences could be expressed for the relative importance of each of the five characteristics and films themselves (e.g. *"the genre is more important than the director"* and *"the actor is less important than the film"*).

The first recommendation algorithm was a content-based algorithm that interpreted monadic and dyadic preferences with respect to the five film characteristics. Item attributes were also semantically enriched with the notion of similarity. The similarities among directors and actors were derived using the wCosine measure and the complete in-house database. Genre, language and nationality similarities were constructed by hand, since there were a manageable number, by ontology experts. The second algorithm was an item-based collaborative filtering algorithm that used the wPearson similarity. Finally, a hybrid algorithm that acted as a content-based algorithm, where collaborative filtering data appeared as a single film attribute, was put in place.

Thirty participants were recruited. All were experienced computer users who had received a university level education. Six participants had already used a recommendation of some sort while the rest had not. Participants were given an instruction sheet that explained them the interface workings of the system. The study took around thirty minutes to complete per participant.

The participants were asked to enter some of their film preferences and then to request some recommendations. It was stressed that they should only enter as many or as few preferences as they would normally do so in the comfort of their own home. Neither should they feel pressured to enter any particular type of preference (film or film attribute).

The system showed a single list of recommendations produced by the three algorithms within. Each algorithm produced five recommendations. Recommendations for films that were the direct subject of preferences were not allowed. The fifteen total recommendations were randomly ordered in a list with duplicate recommendations being removed. The participant was then asked to score each of the recommendations. If they had already seen the film then they were asked to give a score as to how much they liked it, on a scale from 1-5 (1 being the least and 5 being the most). If they had not already seen the film, then they were asked how much they would like to see it, on the same scale from 1-5. As the semantics of the scale may differ depending upon the question, it is important to analyse the results separately.

After scoring all films, the participant was asked to return to the profile manager to enter more preferences and to ask for recommendations once more. The scores given to recommendations were not being used to influence their preferences nor recommendations and were for study use alone. On requesting a second list of recommendations, the system showed films that has already been scored in the first round with their previously given scores. The participant was asked to score all of the new recommendations. Once all films were scored, the participant then had the choice to quit the system or to continue the process. On choosing to quit the system, the participant was given a short questionnaire to complete.

The participants scored at least two recommendation lists. The difference between the average scores given to seen films in their first list of recommendations is compared with the average scores given to seen films in their last list of recommendations (Figure 6). On receiving the first recommendations, the collaborative filtering algorithm produces significantly better results (mean = 4.00; standard deviation SD =

0.91) than the content-based filtering algorithm (mean = 3.35; SD = 1.13). Given the final recommendation, the collaborative filtering algorithm (mean = 4.23; SD = 0.66) produces significantly better results than both the content-based filtering (mean = 3.79; SD = 0.89) and hybrid (mean = 3.70; SD = 0.88) algorithms. The content-based filtering algorithm is the only algorithm that significantly improves between the first and last recommendations given. The content-based and collaborative filtering algorithms tend to improve when more preferences are available while the hybrid filtering algorithm remains stable.

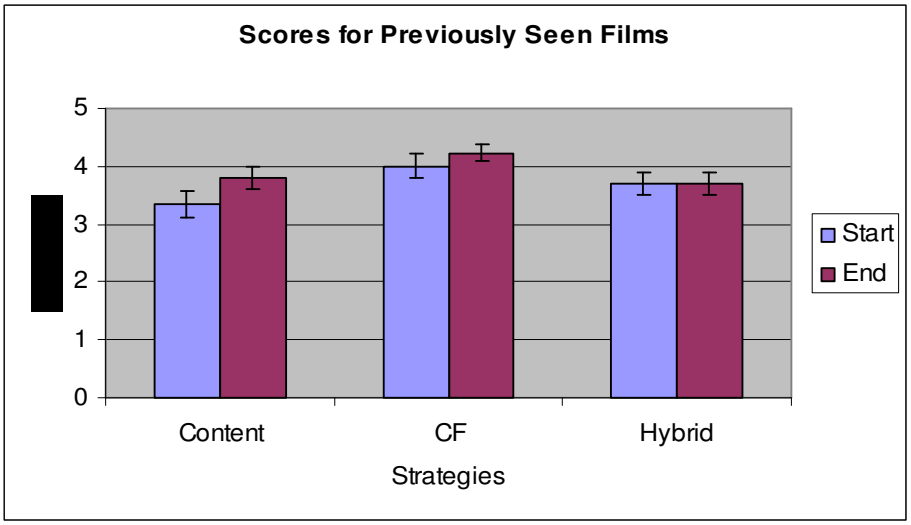


**Figure 6 : Average scores given by the participants to the recommended films that they had already seen before.**

Similarly, the same average scores can be found for unseen films. In this case, no significant differences were found between the scores produced by the three algorithms although there was a trend for each algorithm to produce better results when more preferences were available. Participants have reported a strong desire to watch the unseen films that were recommended (mean score of 4 on 5 to the statements *"I found new films that I want to watch"*, 5 being equivalent to strongly agree).

Participants were asked to enter as many preferences into their profile as they felt comfortable with. They were not encouraged to enter any particular type of preference. The average constitution of a user profile at the moments when the first and last lists of recommendations were requested tends to change in quantity alone (Figure 7). The number of preferences that are given for films is comparable to the number of preferences that are given for film attributes (the five characteristics). Out of the film attributes, participants tended to give their preferences for genres.

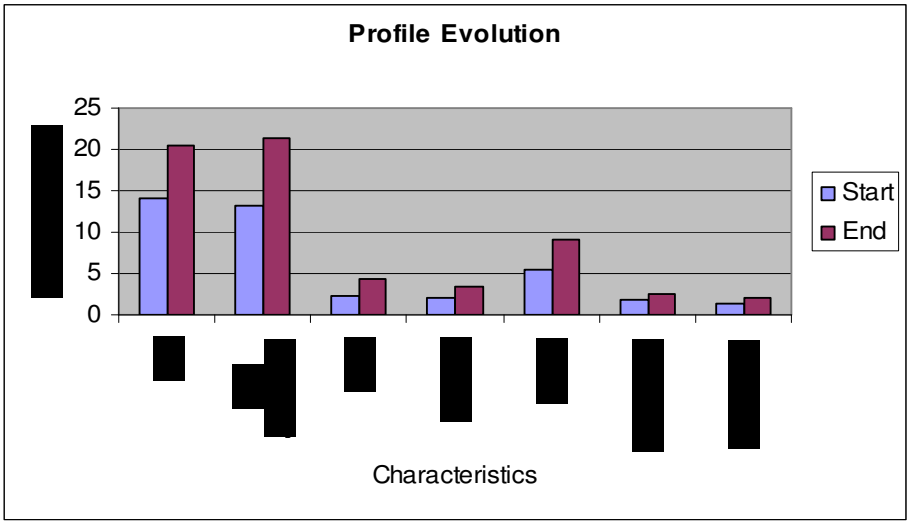


**Figure 7 : Evolution of the profile information given by the participants.**

In the questionnaire, participants were also asked to indicate which type of characteristics they were comfortable to give preferences for or against (Table 3). The majority of participants were comfortable giving preferences for films and substantial minorities indicated that they were comfortable giving preferences for actors, genres and directors. Only a few participants were comfortable giving preferences for nationalities or languages. In addition, a number of participants indicated that they would like to express preferences for the release dates of films, a characteristic that was not manageable in their profile.

| Characteristic | % of Participants |
|---|---|
| Films | 75 |
| Actors | 58 |
| Genres | 54 |
| Directors | 42 |
| Language | 17 |
| Nationality | 12 |

**Table 3 : Film characteristics declared as important by the participants.**

Participants also commented in the questionnaires that they would have preferred a preference rating scale that was richer, such as a 5-point scale that begins at “hate”, rising through “dislike”, “neutral” and “like” and finishing at “love”. As well as enriching the scale, they also expressed an interest in declaring complex preferences that were interdependent or contextual. For example, they would have liked to have expressed a preference for *“Charlie Chaplin”* but only in some of his films and not in all of them.

To summarise, these experiments with real users confirm that collaborative filtering systems have better predictive performance than content-based and hybrid systems. On the other hand, content data have been shown to be important to provide an efficient interaction to the users.

Collaborative filtering holds a very powerful kind of information that is not found in the data for items. It allows tastes to be aligned with one another even when items have no attributes in common. This is the main reason of its good performance.

The content strategy offers items because of the proximity of item attributes. The results for this strategy weren't too bad, which is probably thanks to the semantic similarity encoded between item attributes.

The hybrid strategy appears to have been pulled between the two. The user was allowed to control how much each of the characteristics were taken into account. Perhaps this shouldn't be in the control of the user but should be set at an optimal level for all users. It appears that the mixed information was not best handled by the current system. So further research into hybrid systems is necessary in order to understand the best way in which they can be put together.

Participants were also asked how they thought that the system operated after receiving all of their recommendations. It is very interesting to note that no participants were conscious of the working of the collaborative filtering algorithm. All participants who offered an explanation of the system's logic focused upon the correspondence between individual characteristics of films, such as actors and directors and the preferences that they had indicated. This shows that users are conscious of attribute-based decisions. So using content data to interact with the users seems to be a good technique for gaining the user's trust. It also encouraged them to try and influence the system by making profile modifications.

The participants entered as much information about film attributes as they did about films. Although the suitable algorithms were not yet in place in this study in order to exploit the both of them effectively, it is important to note that users want to express both types of information. An interesting effect was also witnessed. The more that users added their film attribute preferences, the more they began to see the system's logic. Seeing the logic of a system is extremely important when asserting control. In fact, users wanted to be guided by the system in order to give it the most useful information. There is also a very nice psychological effect that comes along with control and that is responsibility. Users become more willing to correct the mistakes of a system if they feel that they are able to. If not, then they often feel that they have hit a dead-end and don't know where to go next.

The fact that users felt lost in entering their profile is a problem that arises when the user has lots of control, but is not guided. A form of guidance could be to propose graphical interfaces that make use of similarity metrics in visualising data. They have been shown to aid the user in finding familiar attributes in (Jack & Duclay, 2008).

## *5 Conclusion*

In the field of recommender systems, collaborative filtering methods often result in better predictive performance than content-based ones, at least when enough rating data are available. This observation follows the intuition. Recommending items based on a set of users' preferences for items carry more meaning than when only item

content information are used. This is especially true with films, since viewers generally select a film to watch based upon more elements than only its genre, director and actors.

On the other hand, content-based filtering offers solutions to the limits of collaborative filtering, as well as a natural way to interact with the users. This issue of designing user-friendly interfaces should not be underestimated. Users must feel like they are in control of their recommendations and that they can navigate themselves out of awkward situations. Otherwise, even if the recommender system is accurate in its predictions, it can suffer from the 'one-visit' problem, if users become frustrated that they cannot express particular preferences or find that the system lacks flexibility. Creating a fun and enduring interaction experience is as essential as making good recommendations.

Focusing on collaborative filtering approaches, item-based ones have been shown to outperform user-based ones. Besides their very good results, item-based approaches have other advantages. Their learning and prediction times are lower, at least for datasets that contain more users than items. They are able to produce relevant predictions as soon as a user has rated one item. Moreover, such models are also appropriate for the navigation in item catalogues even when no information about the current user is available, since it can also present to a user the nearest neighbours of any item she is currently interested in. Finally, the learned neighbourhood matrix can be exported to fill a system that needs similarities between items, and user privacy is preserved.

In this chapter, many issues concerning recommender systems have been presented. The core of the systems is item appreciation predictions. Current research for improving such predictions concerns the combination of different approaches. In that field, many *ensemble methods* can be considered (Polikar, 2006). Specific combinations of different recommender systems are also proposed (Bell et al., 2007).

Experiments with real users suggest the design of systems that are able to propose actions to the users for their profile construction. This issue is opening a new area of research that is related to *Active Learning* (Cohn et al., 1996). Indeed, which items should be rated to optimise the accuracy of collaborative filtering systems, and which item attributes are more critical for optimal content-based recommendations, are questions that are worth to be answered. And this naturally raises the parallel issue of how to design an efficient user interface for such an interaction.

## *References*

(Adomavicius & Tuzhilin, 2005) 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.

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

(Basu et al., 1998) Basu, C., Hirsh, H., & Cohen, W. W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *15th National Conference on Artificial Intelligence* (pp. 714–720).

(Bell et al., 2007) Bell, R., Koren, Y., & Volinsky, C. (2007). Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 95–104). New York, NY, USA. ACM.

(Bilgic, 2004) Bilgic, M. (2004). *Explanation for Recommender Systems: Satisfaction vs. Promotion*. PhD thesis, University of Texas at Austin, Department of Computer Sciences.

(Billsus & Pazzani, 1999) Billsus, D., & Pazzani, M.J. (1999). A personal new agent that talks, learns and explains. In Etzioni, O., Müller, J.P., & Bradshaw, J.M. (Ed.), *3rd International Conference on Autonomous Agents* (pp. 268-275). ACM Press.

(Breese et al., 1998) Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *14th Conference on Uncertainty in Artificial Intelligence* (pp. 43–52). Morgan Kaufman.

(Candillier et al., 2007) Candillier, L., Meyer, F., & Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. In Perner, P. (Ed.), *5th International Conference on Machine Learning and Data Mining in Pattern Recognition* (pp. 548–562), Leipzig, Germany. Springer Verlag.

(Cilibrasi & Vitanyi, 2007) Cilibrasi, R., & Vitanyi, P.M.B. (2007). The Google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3), 370-383.

(Cohn et al., 1996) Cohn, D. A., Ghahramani, Z., & Jordan M. I. (1996). Active Learning with Statistical Models. In *Journal of Artificial Intelligence Research*, 4, 129-145.

(Deshpande & Karypis, 2004) Deshpande, M., & Karypis, G. (2004). Item-based top-N recommendation algorithms. In *ACM Transactions on Information Systems*, 22(1), 143–177.

(Han & Karypis, 2005) Han, E.-H. S., & Karypis, G. (2005). Feature-based recommendation system. In *14th Conference of Information and Knowledge Management* (pp. 446-452).

(Herlocker et al., 2000) Herlocker, J., Konstan, J., & Riedl, J. (2000). Explaining collaborative filtering recommendations. In *ACM Conference on Computer Supported Cooperative Work.*

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

(Jack & Duclay, 2007) Jack, K., & Duclay, F. (2007). Etude de la pertinence de critères de recherche en recherche d'informations sur des données structurées. In *PeCUSI, INFORSID* (pp. 285-297). Perros-Guirec, France.

(Jack & Duclay, 2008) Jack, K., & Duclaye, F. (2008). Improving Explicit Preference Entry by Visualising Data Similarities. In *Intelligent User Interfaces, International Workshop on Recommendation and Collaboration (ReColl).* Spain.

(Janowitz, 2002) Janowitz, M. F. (2002). *A combinatorial introduction to cluster analysis* (Tech. Rep.). Classification Society of North America.

(Karypis, 2001) Karypis, G. (2001). Evaluation of item-based top-N recommendation algorithms. In *10th International Conference on Information and Knowledge Management* (pp. 247–254).

(Kelleher & Bridge, 2003) Kelleher, J., & Bridge, D. (2003). Rectree centroid : An accurate, scalable collaborative recommender. In Cunningham, P., Fernando, T., & Vogel, C. (Ed.), *14th Irish Conference on Artificial Intelligence and Cognitive Science* (pp. 89–94).

(Kleinberg & Sandler, 2004) Kleinberg, J., & Sandler, M. (2004). Using mixture models for collaborative filtering. In *36th ACM Symposium on Theory of Computing* (pp. 569–578). ACM Press.

(Krulwich, 1997) Krulwich, B. (1997). LIFESTYLE FINDER: Intelligent User Profiling Using Large-Scale Demographic Data. *AI Magazine* (pp. 37-45).

(Lam et al., 2006) Lam, S. K., Frankowski, D., & Riedl, J. (2006). Do you trust your recommendations? An exploration of security and privacy issues in recommender systems. In *International Conference on Emerging Trends in Information and Communication Security*.

(Lin et al., 2002) Lin, W., Alvarez, S., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. In *Data Mining and Knowledge Discovery*, 6, 83–105.

(Linden et al., 2003) Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, 7(1), 76–80.

(McNee et al., 2006) McNee, S., Riedl, J., & Konstan, J. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems*.

(Melville et al., 2002) Melville, P., Mooney, R., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *18th National Conference on Artificial Intelligence* (pp. 187-192).

(Miller et al., 2003) Miller, B., Albert, I., Lam, S., Konstan, J., & Riedl, J. (2003). MovieLens unplugged: experiences with an occasionally connected recommender

system. In *8th international conference on Intelligent User Interfaces* (pp. 263-266). ACM.

(Mooney & Roy, 1999) Mooney, R., & Roy, L. (1999). Content-based book recommending using learning for text categorization. In *ACM SIGIR'99, Workshop on Recommender Systems: Algorithms and Evaluation*.

(O'Conner & Herlocker, 1999) O'Conner, M., & Herlocker, J. (1999). Clustering items for collaborative filtering. In *ACM SIGIR Workshop on Recommender Systems*.

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

(Pazzani, 1999) Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. In *Artificial Intelligence Review*, 13(5-6), 393–408.

(Pennock et al., 2000) Pennock, D., Horvitz, E., Lawrence, S., & Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *16th Conference on Uncertainty in Artificial Intelligence* (pp. 473–480).

(Polcicova et al., 2000) Polcicova, G., Slovak, R., & Navrat, P. (2000). Combining content-based and collaborative filtering. In *ADBIS-DASFAA Symposium* (pp. 118-127).

(Polikar, 2006) Polikar, R. (2006). Ensemble systems in decision making. In *IEEE Circuits & Systems Magazine*, 6(3), 21–45.

(Resnick et al., 1994) Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Conference on Computer Supported Cooperative Work* (pp. 175–186). ACM.

(Sarwar et al., 2000) Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce* (pp. 158–167).

(Sarwar et al., 2001) Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *10th International World Wide Web Conference*.

(Shardanand & Maes, 1995) Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *ACM Conference on Human Factors in Computing Systems*, 1, 210–217.

(Ungar & Foster, 1998) Ungar, L., & Foster, D. (1998). Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems*. AAAI Press.

(Vozalis & Margaritis, 2004) Vozalis, M., & Margaritis, K. G. (2004). Enhancing collaborative filtering with demographic data: The case of item-based filtering. In *4th*

*International Conference on Intelligent Systems Design and Applications* (pp. 361–366).

(Wang et al., 2006) Wang, J., de Vries, A.P., & Reinders, M.J. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *29th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 501-508).

(Zhang et al., 2002) Zhang, Y., Callan, J., & Minka, T. (2002). Novelty and redundancy detection in adaptive filtering. In *ACM SIGIR'02*.

(Ziegler et al., 2005) Ziegler, C.-N., McNee, S., Konstan, J., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *14th International World Wide Web Conference* (pp. 22–32).