Analyzing User Actions within a Web 2.0 Portal to Improve a Collaborative Filtering Recommendation System

Andrea Turati, Dario Cerizza, Irene Celino
CEFRIEL – ICT Institute Politecnico di Milano
Via Fucini 2, 20133 Milano, Italy
{*andrea.turati, dario.cerizza, irene.celino*}*@cefriel.it*

Emanuele Della Valle
Politecnico di Milano
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
emanuele.dellavalle@polimi.it

Abstract

The current Web manifests the problem of information overload, especially due to the success of the Web 2.0 paradigm, in which users provide new contents quickly. To help people find the most valuable information, many Web sites include a recommendation system based on a rating mechanism. However, such approach cannot be used when a rating mechanism is not present and, in addition, it does not take into account all the actions performed by the users. We propose an extension of the collaborative filtering approach to design a more effective recommendation system that overcomes those limitations.

1. Introduction

Recommendation systems are a specific type of *information filtering* technique that attempts to present information items (e.g. movies, songs, books, news, images, Web pages) that are likely of interest to the user.

The majority of the successful Web sites including recommendation systems – like Amazon.com, MovieLens, NetFlix, Pandora and Last.fm – usually implements a mechanism that allows users to explicitly assign ratings to the items. However, this method cannot be adopted in context where users are not allowed to leave an explicit evidence about their preference. Furthermore, even if such a mechanism exists, this approach does not take into account many aspects of the user behavior that might be relevant.

In this paper, after a brief description of the current state of the art of the recommendation systems (see Section 2), we inspect how it is possible to enrich the "standard" collaborative filtering technique analyzing the users' behavior in order to improve the quality of the recommendations and we describe a project named *Service-Finder* where we implemented the approach presented in this paper, which we aim to extend in the *SOA4All* project[1] (see Section 3).

1. http://www.soa4all.eu/

2. State of the Art

Information filtering systems decide to select or discard items taking into account the user that will get the results. This is done by comparing the user's *profile* (i.e. a representation of his interests or tastes) to some reference characteristics. A user's profile can be created and maintained either explicitly (i.e. the user specifies it by stating his preferences) or implicitly (i.e. the system monitors his behavior and makes deductions). The characteristics that are compared to the user's profile depend on the algorithm implemented into the information filtering system (or the recommendation system). In literature, two different approaches exist [1]: the *content-based* approach extracts such characteristics directly from the information items while the *collaborative filtering* approach derives them from the user's social environment.

Given a user, a content-based recommendation system suggests those items having the highest correlation between their contents and the user's profile. Therefore, given an item that a user liked very much in the past (i.e. the user assigned it a high rating), it suggests the most similar items being confident that the user will like them [4].

Given a user, a collaborative recommendation system suggests those items preferred by people with a profile most similar to the user's one (i.e. the user will be recommended items that people with similar tastes liked in the past).

Mathematically, the problem can be modeled through a matrix where users and items intersect (see equation 1). Let U be the set of all n users that use the system and I the set of all m items managed by the system. A generic user $u_j \in U$ can express his opinion about an item $i_k \in I$ by assigning a rating r_{u_j,i_k}, which is normally in a binary or numerical scale.

$$R = \begin{bmatrix} r_{u_1,i_1} & r_{u_1,i_2} & \cdots & r_{u_1,i_m} \\ r_{u_2,i_1} & r_{u_2,i_2} & \cdots & r_{u_2,i_m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{u_n,i_1} & r_{u_n,i_2} & \cdots & r_{u_n,i_m} \end{bmatrix} \tag{1}$$

The ratings that user u_j assigned to items represent his preferences, so the user profile used for computing recommendations can be formulated as $\{r_{u_j,i}|i \in I\}$, which is the row corresponding to user u_j in the matrix R.

Accordingly to [2], there are two general classes of collaborative filtering algorithms, which differ in the use of the matrix R: memory-based and model-based.

Given a user, the memory-based approach identifies his profile (i.e. a row in the matrix R) and compares it with all the other user profiles, e.g. using the Pearson correlation coefficient, or the mean squared difference, or the cosine-based algorithm [8]. In this way, user profiles that are most similar to the given one (i.e. their corresponding two rows in the matrix contain similar values, which means that the users gave similar ratings to the same items) are marked as *neighbors* of the given user. Finally, the ratings of the neighbors are used to estimate the rating that the target user would give to a specific unseen item.

A variation of the memory-based approach described so far has been named *item-based* or *item-to-item* collaborative filtering [3], [6], where the similarity is calculated for the items instead of users (i.e., it compares the columns of the matrix R instead of the rows).

The model-based approach aims at compiling a mathematical model reflecting user preferences, using the matrix R to learn some internal parameters and clustering techniques to group the users. This can be done by first compiling (off-line) the complete data set into a descriptive model of users, items and ratings and then computing recommendations by consulting the model [5], [7].

3. Designing a Recommendation System for a Web 2.0 portal

Service-Finder is a Web 2.0 portal[2] that allows users to look for Web services. Beside three standard search functionalities (i.e. free-text search string, a category tree and a tag cloud), it suggests Web services through a recommendation system. SOA4All is another project that goes beyond Service-Finder by providing users with the possibility to semantically describe and execute both services and processes that involve them. It also includes a recommendation system to suggest semantically annotated services and other entities.

In this section we provide an overview of the way we followed in Service-Finder (and that we are going to extend in SOA4All) to design and implement a system that exploits rich information to make recommendations.

3.1. The Approach for Making Recommendations

The ratings that users assign to services can be used by the recommendation process to evaluate user profiles and make recommendations. However, the fact that a user prefers a service can be inferred by the observation of the user behavior with respect to the service. Indeed, within the portal users perform many actions with respect to the services – and not only assigning ratings – and, for example, if a user views the details of a specific service many times and spends a lot of time in editing the service details, then it is possible to say with confidence that the user is addicted to that service.

We associated a weight to each action that a user may perform while visiting the portal (see Table 1). Higher positive weights are associated to actions giving a clear evidence of the high appreciation of the service, while lower positive weights are associated to actions where the appreciation is not so evident or is lower. On the other hand, the actions giving a clear evidence of the rejection of the service are associated with negative weights.

By properly combining all weighted actions that a user did related to a specific service, the system identifies a number reflecting the strength of the relation between the user and the service. A strong relation between a user and a service means that the user really appreciates the service and would recommend it. A weak relation means that the user is not interested in the service or dislikes it, so it is not worth recommending it. The numbers representing the strength of the relations correspond to the values included in the user-item matrix described in Section 2.

Action	Weight
Assign a high/positive rating with a comment to a service	10
Assign a high/positive rating without a comment to a service	8
Add into bookmarks	7
Assign a tag to a service	6
Edit a service	5
Try to invoke a service	4
Click a link related to a service to go to an external document	3
Compare a set of services	3
View the details of a service	2
Select a service (e.g. from the search results)	1
Remove the service from bookmarks	-2
Assign a low/negative rating with a comment to service	-5
Assign a low/negative rating without a comment to service	-10

Table 1. The initial weights of the user actions

Comparing the relations concerning two different users, it is possible to estimate the degree of similarity between the two users: if the set of services tightly connected to a user overlaps considerably the set of services tightly connected to the other one, then the two users are somehow similar. In this way, similarities between all users are computed. Then, given a specific user, it is possible to suggest services that he might be interested in, because those services are appreciated by other users that are similar to him.

3.2. Architecture and Implementation of the Recommendation Component

Figure 1 shows the internal architecture of the recommendation component (arrows represent data flows). The

2. http://demo.service-finder.eu/

Parser continuously monitors the file system and, after the appearance of new log files, it extract information from them. The extracted information is temporarily inserted into a database named *User History*, which represents the history of all user actions done in the portal. Then, the *User-Service Correlation Analyzer* accesses the information stored in the User History and calculates the relations between users and services. The result of the analysis is a User-Service matrix, which is used by the *Recommender* to make recommendations.

In particular, our implementation of the Recommender is based on Taste[3], an extensible framework that implements many collaborative filtering algorithms available in literature. Firstly, it compares the rows of the user-service matrix – representing the users' profiles – in order to calculate the similarity between users. The most similar users to the given one form the user's neighborhood. At run-time, given a specific user, the profiles of his neighbors are taken into account jointly in order to identify the most appreciated services (i.e. the matrix cells containing the highest values). From that set of services, the ones that the given user has not yet seen are recommended.

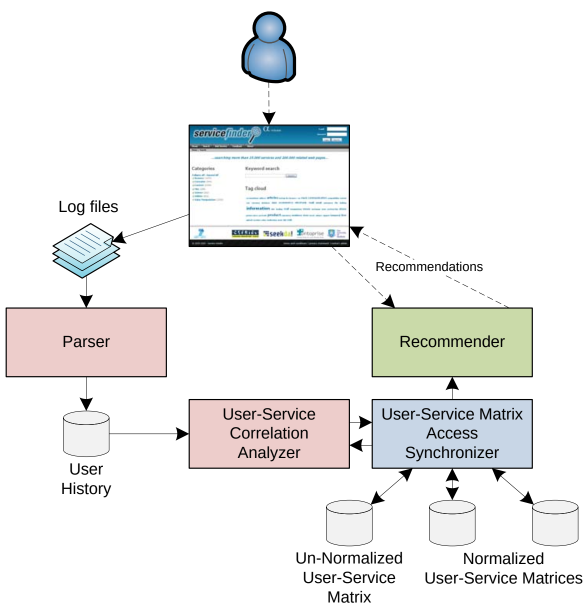

Figure 1. The refined internal architecture

Since on the one hand it takes a while for User-Service Correlation Analyzer to update all the matrix cells and on the other hand the Recommender has to provide recommendations on demand, a synchronization problem might arise due to concurrent accesses to the User-Service Matrix. For this reason, we use two copies of the User-Service Matrix. One matrix is used by Recommender to provide on-line recommendations while the other one is updated by User-Service Correlation Analyzer. When the User-Service Correlation Analyzer finishes to update the off-line matrix, then the two matrices are swapped. After each swap, the updates made on the "new" on-line matrix are copied into the "new" off-line matrix, to keep their content consistent.

The component named *User-Service Matrix Access Synchronizer* is indeed responsible for granting access to the right matrix. In Figure 1, a third matrix also appears beside the two copies of the User-Service Matrix: it differs from them by the fact that it is not normalized (this issue will be discussed later).

In the rest of the section we detail the implementation of the core of our approach.

3.2.1. User-Service Correlation Analyzer. It considers every user separately. Firstly, it collects all actions made by a specific user. Then, for each action, it gets the reference to the service related to that action and updates the cell of the matrix that corresponds to the intersection between the row of the user and the column of the service by adding the action weight. Formally, the User-Service Correlation Analyzer runs the algorithm in Figure 2[4].

The first time this algorithm is executed, the User-Service Matrix contains only null values. When the algorithm is executed, for each user it extracts from the User History all actions that the algorithm has not yet taken into consideration. Then, every action implies to update the value of a cell of the matrix: the weight of the action is summed to the value already stored in that cell.

The values contained in the matrix will vary on different scales due to the fact that users perform different numbers of actions. To allow comparison between any users (irrespective of the number of actions they performed), it is necessary to shift the values of every user to a common scale. For this reason, finally, a normalization step is executed.

After the normalization of all values, the values in all the rows of the matrix can be used to make recommendations. However, if the algorithm is executed once more (for example, because the User-Service Matrix needs to be updated taking into account new actions performed by the users within the portal), the action weights cannot be simply summed to the values included in the cells because they have been normalized while the weights have not.

For this reason, a User-Service Matrix containing values computed by the algorithm in Figure 2 without the normalization step is maintained and managed by the User-Service

3. http://taste.sourceforge.net

4. $max(u)$ and $min(u)$ are respectively the maximum and minimum values in the relations between user u and all services. MAX and MIN are respectively the maximum and minimum value used to express the strength of any relations between users and services in the matrix R. Given an action a, $a.weight$ denotes the weight of the action, $a.timestamp$ denotes the instant at which the action occurred, $a.user$ identifies the user that performed the action and $a.service$ identifies the service related to the action (e.g. if a user rated a service, then the service related to that action is the one rated by the user). $lastExecutionTimestamp$ stores the instant of the last execution of this algorithm.

```
for all user u do
  updatedStrengths ← ∅
  minOrMaxChanged ← false
  for all action a ∈ UserHistory such that a.user = u and a.timestamp > lastExecutionTimestamp do
    s ← a.service

    {Update the value of relation between u and s}
    R(u, s) ← R(u, s) + a.weight
    updatedStrengths ← updatedStrengths ∪ {s}

    if R(u, s) > max(u) then
      max(u) = R(u, s)
      minOrMaxChanged ← true
    end if
    if R(u, s) < min(u) then
      min(u) = R(u, s)
      minOrMaxChanged ← true
    end if
  end for

  {Normalization of the values of the user's relations}
  if minOrMaxChanged then
    for all service s do
      if R(u, s) ≠ NULL then
        R(u, s) ← (R(u, s) − min(u)) / (max(u) − min(u)) × (MAX − MIN) + MIN
      end if
    end for
  else
    for all service s ∈ updatedStrengths do
      if R(u, s) ≠ NULL then
        R(u, s) ← (R(u, s) − min(u)) / (max(u) − min(u)) × (MAX − MIN) + MIN
      end if
    end for
  end if
end for
```

Figure 2. Evaluation of the strength of the relations between users and services

Matrix Access Synchronizer (in Figure 1 it is named *Un-Normalized User-Service Matrix*).

4. Conclusion and Future Works

We proposed an extension of the collaborative filtering approach for making recommendations, designed to exploit all the information coming from a Web 2.0 portal – rather than merely the ratings explicitly assigned by users – in order to build more accurate users' profiles, so that more effective recommendations are provided. In addition, we described a real use-case (Service-Finder) where the approach has been implemented and we cited another project (SOA4All) where we are going to improve the approach, e.g. by adding a content-based module that exploits the semantic annotations available for services to identify those services that are similar to the ones the user interacted with (so that the user-service matrix sparsity [6] can be reduced). Finally, we are going to follow a twofold procedure in order to obtain both some data that demonstrate the quality of the recommendations provided with our approach (by analyzing implicit and explicit user feedbacks), and some guidelines that help developer in setting appropriate action weights.

Acknowledgment

This research has been partially supported by the EU co-funded projects named *Service-Finder* (FP7-IST-215876) and *SOA4All* (FP7-215219).

References

[1] G. Adomavicius and A. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] John S. Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. pages 43–52. Morgan Kaufmann, 1998.

[3] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[4] Raymond J. Mooney, Paul N. Bennett, and Loriene Roy. Book recommending using text categorization with extracted information. In *Proceedings, AAAI-98 Workshop on Recommender Systems*, pages 49–54, Madison, WI, July 1998. AAAI.

[5] Dmitry Y. Pavlov and David M. Pennock. A Maximum Entropy Approach to Collaborative Filtering in Dynamic, Sparse, High-Dimensional Domains. In *In Proceedings of Neural Information Processing Systems*, pages 1441–1448. MIT Press, 2002.

[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[7] Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-Based Recommender System. *J. Mach. Learn. Res.*, 6:1265–1295, 2005.

[8] Upendra Shardanand and Patti Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.