Capturing User Behavior in Multimedia Recommenders

M. Albanese
UMIACS - University of Maryland
College Park, MD 20742, USA
albanese@umiacs.umd.edu

A. d’Acierno
ISA CNR - Via Roma, 64
83100 Avellino, Italy
dacierno.a@isa.cnr.it

V. Moscato A. Picariello
DIS - Via Claudio, 21
80125 Napoli, Italy
{vmoscato, picus}@unina.it

Abstract

Despite the great amount of work done in the last decade, the design of intelligent systems for effective browsing of large multimedia repositories still remains an open field. In this paper, we propose an extension of our previous multimedia recommender, and introduce mechanisms to improve the performance of the system for registered users who explicitly login before starting a browsing session, by dynamically capturing their browsing behavior and adjusting the recommendations accordingly. In addition, we introduce tracking of time spent by users on each item, and take this information into account when computing recommendations. Preliminary results are presented and compared with our previous system.

1. Introduction

Recommender systems are intelligent applications which assist users in their information-seeking tasks, by suggesting useful information items on the basis of their needs and preferences. In order to reach this goal, a key task for the system is to understand the user’s preference. The problem becomes challenging when the recommender system operate on heterogeneous multimedia data: although a huge amount of work has been done in the field of content based multimedia retrieval, no significant effort has been devoted to the problem of intelligent multimedia recommender systems, despite the large availability of multimedia data.

To better illustrate the ideas behind our work, consider the case of a virtual museum, offering web-based access to a multimedia collection of digital reproductions of paintings, educational videos and text documents. In order to make the user’s experience in the museum more interesting and stimulating, access to information should be customized based on the specific profile of a visitor, which includes learning needs, level of expertise and personal preferences. Suppose that a visitor requests, at the beginning of her tour, paintings depicting imaginary landscapes, such as Peter Paul Rubens’ painting entitled Landscapes with the ruins on the Palatine Hill in Rome. It would be helpful if the system, based on these first interactions, could learn her preferences and predict her future needs, by suggesting other paintings representing the same or related subjects, depicted by the same or related artists, or any other multimedia object that has been requested by users with similar preferences. As an example, a user who is currently observing Rubens’ painting might be recommended to see a Nicolas Poussin’s painting Landscape in the Roman Campagna, that is quite similar to the current picture in terms of color and texture, and Italian landscape - Early seventeenth century by William Van Nieulandt, that is not similar in terms of low level features but is similar in terms of semantic content. From the user perspective there is the advantage of having a guide suggesting artifacts which the users might be interested in, whereas, from the system perspective, there is the undoubted advantage of using the suggestions for pre-fetching and caching the objects that are more likely to be requested.

In this paper, we present an extension of the multimedia recommender proposed in [2], which was based on a novel paradigm to combine analysis of user behavior with analysis of low and high level descriptors. Our previous system did not rely on explicit login to capture user preferences, therefore a returning user starting a new browsing session was considered as a new user. In this work, we introduce mechanisms to improve the performance of the system for registered users who explicitly login before starting a browsing session. However, we do not ask registered users to fill in surveys or otherwise provide static information about their preferences, in accordance with the principle that personalization is the process of customizing the content and the structure of an application in order to provide users with the information they are interested in, without asking for it explicitly [3]. In addition, we introduce tracking of time spent by users on each item, and use such information when computing recommendations.

The remainder of the paper is organized as follows. Section 2 briefly discusses related work. Section 3 provides preliminary definitions, and Section 4 introduces the core
aspects of our approach. Finally, preliminary experiments are discussed in Section 5, and concluding remarks are given in Section 6.

2. Related Works

Recommender systems for multimedia objects broadly fall into two classes, although many hybrid solutions also exist. Content-based recommender systems try to recommend items similar to those a given user has preferred in the past. The basic process consists in matching the attributes of a user profile, in which preferences and interests are stored, with the attributes of content items. In [7], the utility of an item \( s \) for a user is estimated based on the utility assigned by the same user to other items that are similar to \( s \). In our context, the drawback of these techniques – heavily based on information retrieval and information filtering – is that they do not benefit from the great amount of information that could be derived by analyzing the behavior of other users [1]. The key idea of the collaborative filtering approach is that users will prefer items that like-minded people prefer. Therefore, a collaborative filtering recommender system makes predictions for a user based on the similarity between the interest profile of that user and those of other users, i.e., data items are recommended based on the similarity between users, rather than on the similarity between data items themselves [10]. Content based filtering and collaborative filtering can be profitably combined to improve the effectiveness of the recommendation process [8].

Our work significantly differs from previous works, as it combines usage patterns, low-level object features and semantic descriptors in a novel approach to recommendation.

3. Preliminary Definitions

In [2], we introduced the notion of multichannel browser – a browser which allows concurrent browsing of \( h \) channels, each assigned to a specific media type (image, video, audio, text). We also introduced the notion of multichannel object – a snapshot of what is being displayed on a channel browser at a given time – and defined the whole recommendation strategy with respect to these notions. In this paper, for reasons of space and without loss of generality, we will consider the case of a single channel multimedia browser \( B \), and deal exclusively with images\(^1\). Instead, we will focus on the core aspects of the system, highlighting the novel features introduced by this paper.

The basic intuition behind our approach is that, if we can predict what objects a user is likely to request next and use such predictions as recommendations, it is likely that the user will accept one of the recommendations, rather than jumping to entirely unrelated objects or starting a new browsing session altogether. Experimental results have confirmed this intuition.

In [2], we introduced the concept of usage pattern as a means to keep track of user behavior. Let \( O \) denote the set of objects to be made accessible through the browsing system. We now introduce the concept of temporal usage pattern, which explicitly factors in the temporal dimension.

**Definition 3.1 (Temporal Usage Pattern)** Given a multimedia browser \( B \), a usage pattern \( P_i \) of length \( k \) is the ordered sequence of \( k \) pairs defined as:

\[
P_i = ((O_{i1}, \tau_{i1}), (O_{i2}, \tau_{i2}), \ldots, (O_{ik}, \tau_{ik}))
\]

where \( O_{ij} \in O \) is the \( j \)-th object visualized by the user during browsing session \( i \), and \( \tau_{ij} \) is the fraction of the total temporal duration of session \( i \) spent on \( O_{ij} \).

Let \( \mathcal{P} \) denote the set of all temporal usage patterns associated with past browser sessions. We will refer to this set as the usage log. As we will discuss in Section 4, our approach to providing a user with valid recommendations is based on finding the patterns in \( \mathcal{P} \) that best match the current usage pattern of the user and making suggestions based on the behavior of the users who generated those patterns. Comparing usage patterns will require the ability of comparing objects, specifically images in our case. In fact, the definition of similarity metrics to compare multimedia objects is a key element in the design of an effective multimedia recommender. Our approach is to combine low-level features and high-level semantic descriptors. In the literature, images have been usually characterized through three fundamental low-level features, namely color, texture, and shape [9]. Image processing algorithms can automatically extract these features and compute the distance between two images as the distance between their feature vectors in the feature space. In the prototypical implementation of our recommender system, we adopted the distance function \( \delta_F \) included in the Oracle Intermedia extension of the Oracle DBMS, which exploits color, texture, shape and spatial information of images.

As for the high-level descriptors, different solutions have been proposed to compare different sets of annotations using some form of background knowledge, represented for example through an ontology. Without loss of generality, we will reasonably assume that semantic annotations of objects have been manually generated by human experts based on taxonomies. A taxonomy \( T = (\mathcal{N}, \mathcal{E}) \) is a hierarchical concept network, where a node \( n \in \mathcal{N} \) in the hierarchy represents a concept and an edge \( e \in \mathcal{E} \) represents a parent/child relationship between two concepts. We now recall the notion of semantic annotation formalized in [2] and extend the proposed metric for comparing objects based on

\(^1\)However, we will often use the term object instead of image to emphasize that the proposed approach can be applied to any media type.
their annotations, in order to incorporate browsing preferences captured from historical data of registered users.

**Definition 3.2 (Annotation Schema)** Given a taxonomy \( T = (\mathcal{N}, \mathcal{E}) \), an Annotation Schema is a tuple 
\[
\Lambda_T = (A_1, \ldots, A_n, B_1, \ldots, B_m) 
\]
where \( \{A_i\} \) are attributes s.t. \( \forall i \in [1, n] \ dom(A_i) \subseteq \mathcal{N} \), and \( \{B_j\} \) are attributes s.t. \( \forall j \in [1, m] \ dom(B_j) \not\subseteq \mathcal{N} \).

In other words, only values of attributes \( A_i \) correspond to nodes of the taxonomy. For example, in our virtual museum scenario, we assume the existence of a taxonomy that models the concepts of painters, pictorial genres and depicted subjects. Thus, we can assume \( n = 3, m = 1, \) and \( \Lambda_T = (A_1, A_2, A_3, B_1) = (Author, Genre, Subject, Title) \).

**Definition 3.3 (Semantic Annotation)** Given a taxonomy \( T \), an annotation schema \( \Lambda_T \) and an object \( O \), a Semantic Annotation of \( O \) is a tuple 
\[
\Lambda_T(O) = (v_1^A, \ldots, v_n^A, v_1^B, \ldots, v_m^B) 
\]
where \( \forall i \in [1, n] \ v_i^A \in dom(A_i) \) and \( \forall j \in [1, m] \ v_j^B \in dom(B_j) \).

The following definition extends the metric proposed in [2] by incorporating dynamic browsing preferences. We start from the assumption that, given a taxonomic attribute \( A_k \), the similarity between objects \( O_i \) and \( O_j \) is inversely proportional to the length of the path from the respective values of \( A_k \) and directly proportional to the depth into the hierarchy of their subsumer [6], (e.g., the closer authors, genres and subjects are, the more similar two paintings are).

**Definition 3.4 (Taxonomic Distance)** Given a taxonomy \( T \) and an annotation schema \( \Lambda_T = (A_1, \ldots, A_n, B_1, \ldots, B_m) \), the Taxonomic Distance between two objects \( O_i \) and \( O_j \) is defined as
\[
\delta_T(O_i, O_j) = 1 - \sum_{k=1}^{n} \alpha_k \cdot e^{-\alpha \cdot l(v_i^k, v_j^k)} \cdot \left(1 - e^{-\beta \cdot d(v_i^k, v_j^k)}\right)
\]
where \( v_i^k \) and \( v_j^k \) are the values of attribute \( A_k \) for \( O_i \) and \( O_j \) respectively; \( l(v_i^k, v_j^k) \) is the length of the shortest path between \( v_i^k \) and \( v_j^k \); \( d(v_i^k, v_j^k) \) is the depth in the hierarchy of their subsumer; \( \alpha \) and \( \beta \) are parameters scaling the contribution of shortest path length and depth respectively; and \( \alpha_k \), with \( k \in [1, n] \) are weights - s.t. \( \sum_{k=1}^{n} \alpha_k = 1 \) - measuring the relative importance of the fact that objects \( O_i \) and \( O_j \) have equal or similar values of attribute \( A_k \).

The weighting factor \( \alpha_k \) represents how important is for the user that two objects have equal or similar values of attribute \( A_k \) in order to be considered similar, or rather relevant to each other. For registered users, such weights can be learned by observing usage behavior over time, while they can be simply set to \( \frac{1}{n} \) for guests. Given a user \( u \), consider the set \( P_u \subseteq P \) of usage patterns generated by \( u \), and the set \( P_u^* \) of all the consecutive pairs of objects in patterns \( P \). Then \( \alpha_k \) can be computed as follows:
\[
\alpha_k = \frac{|\{(O_{i_j}, O_{i_{j+1}}) \in P_u^* | \delta_T(O_{i_j}, O_{i_{j+1}}) = \delta(O_{i_j}, O_{i_{j+1}})|\}}{|\{(O_{i_j}, O_{i_{j+1}}) \in P_u | \delta_T(O_{i_j}, O_{i_{j+1}}) = \delta(O_{i_j}, O_{i_{j+1}})|\}}
\]

In other words, we can estimate \( \alpha_k \) by counting the number of cases in which the object selected by the user at one step of the browsing session had the same value of attribute \( A_k \) as the previous object. Note that, in order to make this mechanism more general, the equality \( \delta_T = \delta \) in Equation 4 can be replaced by the inequality \( e^{-\alpha \cdot l(v_i^k, v_j^k)} \cdot \left(1 - e^{-\beta \cdot d(v_i^k, v_j^k)}\right) \leq \sigma, \sigma \) being a threshold. In addition, to make the choice of \( \alpha_k \) adaptive to drastic changes in user behavior, the set \( P_u \) may be limited to include only the most recent usage patterns.

The distance we adopt in our system is a combination of feature-based and taxonomic distances, as defined below.

**Definition 3.5 (Distance)** The Distance between two objects \( O_i \) and \( O_j \) is defined as:
\[
\delta(O_i, O_j) = \alpha_F \cdot \delta_F(O_i, O_j) + \alpha_T \cdot \delta_T(O_i, O_j)
\]
where \( \alpha_F \) and \( \alpha_T \) are two weighting factors.

Note that \( \alpha_F \) and \( \alpha_T \) can be learned from historical data with a similar procedure to the one described for \( \alpha_k \).

### 4. Recommendation Algorithm

As anticipated, our approach to recommendation is based on finding the patterns in \( P \) that best match the current usage pattern. Therefore, we are interested in the notion of similarity between usage patterns. A well-known algorithm in this field is the Levenshtein algorithm [5], that was designed to evaluate the distance between two words as the total cost of the basic operations (insertions, deletions and substitutions) needed to transform a string into the other. The Levenshtein distance gives a measure of how much two sequences of symbols differ in terms of alignment, without taking into account the nature of the symbols themselves: the cost of substituting a symbol \( a \) with a symbol \( b \neq a \) is fixed and does not depend on the specific nature of \( a \) and \( b \).

The idea behind our approach is to evaluate the similarity between patterns based on the similarity between the objects in the patterns. To this aim, we use the metric defined in Section 3 and we adopt an indexing strategy to guarantee fast access to objects and patterns of interest. We remind the
reader that users have the option not to login, therefore, in this case, we need to learn their preferences as they browse the collection. The length of a usage pattern starts at zero and then increases by one unit every time the user requests a new item from the collection. For this reason, comparing the current usage pattern with full patterns in the usage log might not be effective. Instead, a measure of local similarity between patterns has proved to provide better results [2]. In other words, we are interested in finding patterns containing subsequences that match the current pattern in an optimal way and then make suggestions based on them. Starting from the Levenshtein theory, we have designed an algorithm that evaluates the local similarity between usage patterns, taking into account the features of the objects in them, and the time users spent on each item. Given two usage patterns \( P_k \) and \( P_l \), the algorithm computes a matrix \( \Omega \) whose \((i, j)\) element represents the maximum local similarity between two patterns, respectively containing the first \( i \) elements of \( P_k \) and the first \( j \) elements of \( P_l \). The highest value in \( \Omega \) is the overall local similarity between \( P_k \) and \( P_l \) and corresponds to the best local alignment between those patterns. The following definition introduces the functions used to compute the cost of an alignment in terms of substitutions, insertions and deletions.

**Definition 4.1 (Cost Functions)** Consider two temporal usage patterns \( P_k = \langle (O_{k1}, \tau_{k1}), \ldots, (O_{km}, \tau_{km}) \rangle \) and \( P_l = \langle (O_{l1}, \tau_{l1}), \ldots, (O_{ln}, \tau_{ln}) \rangle \) of length \( m \) and \( n \) respectively. We define the substitution, insertion and deletion cost functions as follows:

\[
Sub(O_{ki}, O_{lj}) = \frac{\tau - \chi(O_{ki}, O_{lj})}{1 - \tau} \tag{6}
\]

\[
Ins(O_{lj}, O_{ki}) = \frac{\tau - \min\{\chi(O_{ki}, O_{lj}), \chi(O_{ki+1}, O_{lj})\}}{1 - \tau} \tag{7}
\]

\[
Del(O_{ki}, O_{lj}) = Ins(O_{ki}, O_{lj}) \tag{8}
\]

where \( \tau \in [0, 1] \) is a threshold and \( \chi(O_{ki}, O_{lj}) = (1 - \delta(O_{ki}, O_{lj})) \cdot e^{-\gamma |\tau_{ki} - \tau_{lj}|} \) is a similarity metric based on the distance \( \delta \) defined by Equation 5, with \( \gamma \) a parameter to scale the contribution of the difference between the fractions of time \( \tau_{ki} \) and \( \tau_{lj} \) spent on \( O_{ki} \) and \( O_{lj} \) respectively.

\( Sub(O_{ki}, O_{lj}) \) is the cost of replacing the \( i \)-th element of \( P_k \) with the \( j \)-th element of \( P_l \). Note that the similarity between two objects is penalized by a factor \( e^{-\gamma |\tau_{ki} - \tau_{lj}|} \) which takes into account differences in the relative amount of time spent on \( O_{ki} \) and \( O_{lj} \) in their respective browsing sessions. Also note that, when the similarity between \( O_{ki} \) and \( O_{lj} \) is equal to the threshold, the cost is 0; when the similarity is above the threshold, the cost is negative, meaning that it reduces the overall cost, actually rewarding the substitution. Similarly, \( Ins(O_{lj}, O_{ki}) \) is the cost of inserting the \( j \)-th element of \( P_l \) after the \( i \)-th element of \( P_k \) and \( Del(O_{ki}, O_{lj}) \) is the cost of deleting the \( i \)-th element of \( P_k \) and \( j \)-th element of \( P_l \) being the position of the element in \( P_l \) aligned with \( O_{ki} \). The threshold \( \tau \) has been defined as a function of the size of the collection, by posing \( \tau = (1/\log|O| - 0.4)/\log|O| \), thus making the threshold more restrictive as the size of the log increases. For example, \( \tau = 0.8 \) when \( |O| = 100 \) and \( \tau = 0.9 \) when \( |O| = 10000 \).

Figure 1 lists the algorithm used for evaluating local similarity between temporal usage patterns. Given an alignment, the algorithm assigns a positive score (negative cost) to each substitution of an element \( O_{ki} \) of \( P_k \) with an element \( O_{lj} \) of \( P_l \) that is similar to \( O_{ki} \), with similarity above the threshold \( \tau \). Vice versa a negative score is assigned to each substitution where the similarity is below the threshold. In both cases the absolute value of the score is proportional to the similarity measure between the two objects. In a similar way the insertion of an element \( O_{lj} \) of \( P_l \) between elements \( O_{ki} \) and \( O_{ki+1} \) of \( P_k \) is penalized by an amount that is greater when the new element is dissimilar from both \( O_{ki} \) and \( O_{ki+1} \). In the following we define a measure of similarity between objects that is latent in the usage patterns, in the sense that usage patterns capture choices made by users based on the perceived relationship (visual or semantic) between different objects. To this aim we need to define the following sets:

\[
P_{\gamma} = \{ P \in P \mid \text{local-similarity}(P, P_c) \geq \gamma \} \tag{9}
\]

\[
O_{\gamma} = \{ O \in O \mid \exists P \in P_{\gamma}, \text{next}_P(P_c) = O \} \tag{10}
\]

\( P \) is the set of all the patterns in the log that are similar to the current pattern \( P_c \) within a threshold \( \gamma \), while \( O \) is the set of objects that, in browsing sessions corresponding to the patterns in \( P_{\gamma} \), were seen after the subsequence aligned with \( P_c \). The procedure described in the following equally applies to the case of registered users and to the case
of guests. The main difference is in the way the set \( P_\gamma \) is constructed. First, as we have previously discussed, for any given user the similarity function \( \chi \) depends on some parameters \( - \alpha, \beta \) and \( \alpha_k \), with \( k \in [1, n] \) – which are learned based on the browsing history of that specific user. In addition, the set \( P_\gamma \) of most similar patterns can be forced to only include patterns generated by the same user, or at least a given portion of patterns generated by the same user. Let us now define the following sets:

\[
O_c = O_\gamma \cup \text{NN}(O_c, k) \tag{11}
\]

\[
P_i = \{ P \in P_\gamma \mid \text{next}_P(P_c) = O_i \}, \forall O_i \in O_c \tag{12}
\]

where \( \text{NN}(O_c) \) selects the \( k \) nearest neighbors of the current object \( O_c \) being visualized by the user. \( O_c \) is the set of candidate objects for inclusion in the recommendation list, while \( P_i \) is the subset of \( P_\gamma \) containing those patterns having \( O_i \) as the first element following the subsequence aligned to \( P_c \). The threshold \( \gamma \) is needed because we want to base recommendations on patterns that are highly similar to the current pattern. Moreover, considering only a subset of \( P \) reduces the complexity of the algorithm. The threshold \( \gamma \) should be close enough to 1 in order to get high precision results and it should be higher when the size of the log increases. We have chosen \( \gamma = (|P| - 0.2)/|P| \).

**Definition 4.2 (Latent Similarity)** The Latent Similarity \( \chi_P \) between an object \( O_i \) and a current usage pattern \( P_c \) is defined as

\[
\chi_P(O_i, P_c) = \frac{\sum_{P \in P_i} \text{local-similarity}(P, P_c)}{\max \{ \sum_{P \in P_i} \text{local-similarity}(P, P_c) \}} \tag{13}
\]

\[
\max \{ \sum_{P \in P_i} \text{local-similarity}(P, P_c) \} \text{ being a normalization factor.}
\]

We can finally define how to build a ranked list of recommendations. The idea is to weight both the similarity w.r.t. the last requested object and the similarity in terms of usage patterns. In fact, when a user starts browsing the collection, her current pattern is too short to make useful recommendations based on usage patterns only. In this case, it would be useful to take into account the features of the last requested object and recommend the objects most similar to it. We can finally introduce the definition of recommendation grade.

**Definition 4.3 (Recommendation grade)** Given the current pattern \( P_c \) and the last element \( O_c \) in \( P_c \), the recommendation grade \( \rho \) of an object \( O_i \) is defined as:

\[
\rho(O_i) = \alpha_c \cdot \chi(O_i, O_c) + \alpha_P \cdot \chi_P(O_i, P_c) \tag{14}
\]

\( \alpha_c \) and \( \alpha_P \) being two weighting factors.

In conclusion, the system will recommend the \( k \) objects in \( O_c \) exhibiting the higher values of \( \rho \).

### 5. Preliminary Experimental Results

We tested our recommender system on a collection including 1000 paintings (20 genres, 100 authors and 80 different subjects) by repeating the experiments described in [2]. A user who is just starting her tour of the virtual museum can select any of the paintings in the exhibition by means of standard search methods. As she makes the first request for a painting, the system begins to assist her visit. At this time, the suggestions offered by the system are exclusively based on the retrieval of the most similar images w.r.t. the metric \( \delta \) defined by Equation 5. At this stage, in the case of a registered user, the parameters \( \alpha_k \) capturing the user’s browsing preferences will play a key role in bootstrapping the recommendation algorithm. If the current image is not the first of the session, the system attempts to propose both paintings similar to the current one and paintings accessed within similar browsing sessions².

Initially, we asked a group of 25 people to use the system for some days, in order to collect a significant amount of usage patterns (several hundreds). Then we asked a subset of them (12 people) to browse the image collection and complete several browsing tasks (20 tasks per user) of different complexity³ (5 tasks for each complexity level), using the well-known image database system Picasa (taxonomy is implemented as albums, folders and descriptions). After this test, we asked them to browse once again the same collection with the assistance of our recommender system, and complete other 20 tasks of the same complexity. Two strategies were used to evaluate the results of this experiment: empirical measurements of access complexity in terms of mouse clicks and time, and TLX (NASA Task Load Index factors) [4]. With respect to the first strategy, we measured the Access Time (\( t_a \)) – the average time spent by the users to request and access all the objects for a given class of tasks – and the Number of Clicks (\( n_c \)) – the average number of clicks necessary to collect all the requested objects for a given class of tasks. Table 1 reports the average values of \( t_a \) and \( n_c \), for both Picasa and the two versions of our system, for each of the four task complexity levels defined. We then asked the users to express their opinion about the capability of Picasa and our system respectively to provide an effective user experience in completing the assigned browsing tasks. To this end, we used the TLX evaluation form, which allows to assess the workload on operators of various human-machine systems. Specifically, TLX is a multi-dimensional rating procedure that provides an overall workload score based on a weighted average of ratings on six sub-scales: mental demand, physical demand, temporal demand, own

²The user is not required to accept one of the recommended items, but she can jump, at any time, to any other image in the collection.

³Query complexity (low, medium, high, very high) depends on several factors: number of objects to browse, type of features involved in the task specification and number of search constraints.
of browsing, leveraging the intuition that an object observed for a substantial portion of a browsing session is likely to be more relevant to the user than an object she watched only for a negligible amount of time. Preliminary results obtained in a virtual museum scenario, show that the proposed novel features of the system actually improve performance.

References


