Combined Proceedings of the Social Network Analysis and Norms for MAS Symposium - SN-MAS2010 Section,
Giulia Andrighetto, Guido Boella, Ugo Pagallo, and Serena Villata (Ed.), at the AISB 2010 convention, 29 March – 1 April 2010, De Montfort University, Leicester, UK

Social Aspects of Video Recording

Alessandro Basso, Marco Milanesio, André Panisson[1]

Abstract. Recently, Personal Video Recorders (PVRs) are gaining success among web applications. In this paper we give some preliminary results on a framework for identifying communities on the basis of the explicit usage of a PVR. Users programming their recordings are clustered and aggregated together and social interactions among users come out from the grouping of common interests, for sake of recommendation purposes. Items recorded by similars users (within a certain measure) are ordered for recommending unseen broadcasted programs. We exploit these interactions in the Facebook social network, with an application where direct communications between users contribute to our knowledge gathering. A detailed analysis of one-year observations is given, along with some considerations about the implicit user profiling, showing that it is possible to identify groups of users with similar interests, despite the lack of explicit information about their likes or dislikes.

1 Introduction and Motivation

In the last years, we have been seeing to an always increasing convergence trend between traditional media, such as the Television, and the Internet, which resulted in a huge amount of available contents to the end-user. Common examples of such a phenomena are identifiable in services such as IP-TV, VideoOnDemand or YouTube, which have undergone a noticeable diffusion due to the high bandwidth availability granted by the last generation of Internet connection technologies.

In such a context, the need of being able to access a large amount of multimedia contents, without requiring the user to be constantly in front of the media device, has enabled the emergence of a new generation of Personal Video Recorders (PVRs) [1]. Differently from a common video recorder, a PVR is basically a software package allowing users to record audio and video contents by properly setting their time intervals (e.g., MythTV[2], SageTV[3], MediaPortal[4]). This feature, though, comes commonly without the possibility to involve the user in a more interesting and enjoyable experience, like other web applications do, for example, by implementing user personalization through a *recommender system*, and, thus, by suggesting the user contents specifically tailored to her interest [8].

However, the intrinsic high dynamism of the PVR context, where contents may be registered and viewed by users with a daily frequency, presents a series of difficulties to the direct application of common recommendations algorithms. Aware of the known unreliability of Electronic Programming Guides (EPGs), we rely only on users' ability in precisely determine the timings of each recording. As we do not have a reliable source of information to know that there is a specific movie on a specific channel, we only know that several recordings have been set within a certain range of timings on that channel. In this perspective, we can not deal with predefined *discrete* and *permanent* objects. Instead, we use events characterized by a specific temporal validity - the users' recordings - to extract information about which events are being recorded, on a bottom-up approach to determine the discrete events.

Our goal in this paper is to provide a social dimension to the very dynamic PVR-services domain, which is typically focused on the single user. By using events discretization combined with a collaborative filtering (e.g., [8]) approach to make good predictions of user's recordings, we aim to aggregate users with similar interests with the purpose of improving their overall multimedia experience. Our approach is based on the analysis of real data generated by the Faucet PVR system [5], integrated in a web-based podcasting service named VCast. Faucet allows users to record favorite TV and Radio (Italian) programs, that can be further downloaded into their devices (e.g., iPod, PC, notebook).

The user can set up her own programming and see or download her recordings by the use of a simple web interface. Bringing the ability to record and group into a single feed public and private channels (such as radio and TV recorded programs), Faucet PVR offers a single framework for creating and aggregating personal podcast compilations. This system produces a very rich and dynamic data set, populated by real users expressing their preferences through the recorded programs. If such a collaborative filtering approach is able to make good predictions of the recordings, it could be immediately applied in a recommender system tool for implementing user personalization. We start here from our previous analysis given in [3], in which a detailed description of the events' definition is given.

This paper is organized as follows. Section 2 gives a brief introduction on how to extract social behavioral patterns from the explicit usage of a PVR application. Next, it is explained (Section 3) how these information can be used for aggregating items and exploited (Section 4) for recommendation purposes. Section 5 presents some early results of our analysis. Finally, Section 6 traces the route for our future works in this subject.

2 Context

The goal of a recommendation system in the PVR context is to suggest a customized set of transmissions to the users [9]. However, data coming from the Faucet PVR are not immediately usable to identify permanent and discrete events (i.e., transmissions), but assume the form of unstructured information, which has to be properly processed. In particular, let T be the set of transmissions during a day and t_i be a specific transmission broadcasted on channel c_{t_i}, starting

[1] Department of Computer Science, University of Turin, Italy, email: alessandro.basso@di.unito.it, marco.milanesio@di.unito.it, andre.panisson@di.unito.it

[2] `http://www.mythtv.org/`, Last Visited 10 Feb. 2010

[3] `http://www.sagetv.com/`, Last Visited 10 Feb. 2010

[4] `http://www.team-mediaportal.com/`, Last Visited 10 Feb. 2010

[5] `http://www.vcast.it/faucetpvr`, Last Visited 10 Feb. 2010

at time b_{t_i} and ending at time e_{t_i}. Then, t_i can be directly used in the recommendation engine and this property holds $\forall t \in T$.

On the contrary, data collected by the Faucet system (i.e., users' recordings) differ from t_i in the sense that they are a set R of several different recordings r_i with a temporal validity, each referring to a specific content. However, recordings with different properties (e.g., start time, end time, title and so on) may refer to the same content: $r_i \neq r_j$ even if r_i, r_j refer to the same transmission. Clearly, this property does not hold with discrete and well defined events such as the transmissions: $\forall t \in T, t_i \neq t_j$.

Respect to other works in this domain (e.g., [6]) we have to face the periodicity of a recommendable item. This means that, for example, given a weekly broadcasted program, we can not recommend the next-week event to the user, since the programmed recording explicitly contains this information.

A noticeable property characterizing the context in which we operate is the lack of a well defined programming for recorded contents. Despite several EPG sources do exist, we can not consider them reliable enough to be used to extract the input information of the recommender engine. Thus, we opted for a different approach: exploiting the so-called *wisdom of the crowd*, we rely on users' knowledge to define: *what* contents are broadcasted on the major (Italian) TVs and radios, *where* (the channel) and *when* (the timings).

The task of defining the parameters related to every recorded transmission, such as channel, title, timings and periodicity, is therefore demanded to single users. Since it is their interest to make sure that the information inserted in the Faucet PVR are as much precise as possible, we can consider such data reliable enough to be used in the recommendation process. As result of this users activity, we can obtain a series of recordings, defined as follows:

$$r_i =< u_{r_i}, c_{r_i}, tl_{r_i}, b_{r_i}, e_{r_i}, p_{r_i} >$$

where u_{r_i} is the user who set the recording, c_{r_i} is the channel, tl_{r_i} is the title, b_{r_i} and e_{r_i} are the the starting and ending time respectively, and p_{r_i} is the periodicity.

One of the main tasks of the proposed framework is the *discovery* of events from the amount of unstructured data obtained through the recordings. A *discrete event* can be defined as the representative of a broadcasted transmission, obtained as the result of the aggregation of several different recordings. More formally, given the set R of recordings $r_1, \ldots, r_n$, we cluster every recording possessing the same attributes for channel and periodicity but similar timings, i.e., $\forall r_i, r_j \in R$ so that $c_{r_i} = c_{r_j} \wedge p_{r_i} = p_{r_j} \wedge |b_{r_i} - b_{r_j}| > d_b \wedge |e_{r_i} - e_{r_j}| > d_e$, where d_b, d_e are the clustering diameters for the starting and end times respectively. The result is a series of clusters, whose centroid is chosen as representative. Finally, through the aggregations of similar clusters, we can obtain a number of discrete elements, defined as follows:

$$d_i =< \{u_{d_i}\}, c_{d_i}, tl_{d_i}, b_{d_i}, e_{d_i}, p_{d_i} >$$

where $\{u_{d_i}\}$ is the list of users who set a recording referring to that transmission, c_{d_i} is the channel, tl_{d_i} is the title chosen among those given by users, b_{d_i} and e_{d_i} are the the starting and ending times respectively, and p_{d_i} is the periodicity. For more details, please refer to [3].

An example of the highly dynamic environment of PVRs is given in Figure 1, where we show all the starting times referring to those recordings which contribute to define a single discrete event (i.e., a transmission).

It is important to underline that no information on the content of recordings is used to define a discrete event. Indeed, the Faucet PVR does not provide any reference to the type of transmission recorded (e.g., sport program, news, or comedy-movie), nor we can rely on information inserted by users in titles, being characterized by a high variability. Thus, we can only discriminate between two programs (on the same channel) by clustering on top of timings, plus the periodicity of the recording. In addition, the lack of information about the content of the recordings does not allow us to apply content based recommenders, which focus on the description of item and user profiles in order to recommend items [7]. On the contrary, our approach only relies on the behavior and characteristics of large interconnected networks, by exploiting similarities between their users.

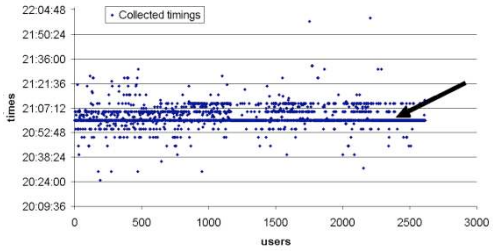

Figure 1. More than 2600 users programmed the recording of this specific transmission. As it turns out, several start times were given. The arrow indicates the representative timing we choose.

3 Aggregating recordings

As told in the previous section, each recording is characterized by a *channel*, a *timing*, a *periodicity* and a *title*, a completely uncontrolled and free user generated tag. Recordings are grouped together by channel and clustered on the timing: if two recordings on the same channel present similar start and end times, then we cluster them together. Neither the size of the cluster, nor the final number of clusters is fixed *a-priori*. At the end of the clustering process, the recordings minimizing the timing differences within the computed clusters are taken as centroids of the clusters.

The aim of the clustering phase is the transformation of the recordings domain to a domain of *discrete events*: the set of individual recordings must be transformed into a set of distinct *broadcasted programs*, i.e., from the continuous domain of timings to the discrete domain of programs. Still the clustering itself is not enough. After the clustering, attributes of the discrete events are computed from the centroid of each cluster. Specifically, timings are computed by considering the median of the timings extracted from every element of the cluster, whilst the most frequent title in the cluster is chosen as a representative of that discrete element.

The aim of the aggregation procedure is to correct possible inaccuracies in the clustering phase. In fact, as in old VCR programming, most of recordings are also loosely timed in the PVR domain, meaning that the user usually started the recording a little bit earlier to stop it a little bit after, to ensure the completeness of the desired TV show. For instance, the reader may think of a TV program starting at 8 pm. It is likely to observe a large number of recordings starting in the interval $[7.50, 8.00]$, whilst others might start in the interval $[7.40, 8.10]$. Due to this fact, it is problematic to directly treat both underestimation and overestimation of the timings in the clustering phase, as two possible cases can occur:

1. by choosing a larger clustering diameter (e.g., 60 minutes), we allow a more complete aggregation of recordings referring to the same content, with the side effect of possibly *skipping* the discretization of very short transmissions (e.g., whose duration is around 15 minutes).
2. By defining a shorter clustering diameter that barely takes into account underestimation and overestimation (e.g., 10 minutes), we end up with a larger number of clusters, which are likely to generate new discrete elements rather than being merged with existing ones.

In the current implementation of the aggregation algorithm, we opted for the first case, due to the low probability of occurrence of very short transmissions. The final creation of a discrete element is obviously constrained by the possible past insertion of a very similar discrete event, from which the parameters of the currently computed discrete event are updated.

Two main problems were faced in the clustering and aggregation procedure. The first, as said before, is the fact that the end user has an almost complete control on the system, thus she can set any timing and any title she wants. The channel is the only field that is given. This leads to a wide variety of titles and timings for recording the same program, but, while for the timings this was solved with the introduction of various thresholds, for the string representing the title it was necessary to compute the most frequent title of the cluster. The second problem is the periodicity of the recordings. As long as five options are available in VCast (i.e., daily, weekly, mon-fri, mon-sat, no-repeat), it was necessary to develop five different aggregating functions for managing the different behavior of the recordings.

4 Recommending events to users

The first outcome of the aggregation procedure is the daily top hits chart, namely the *most popular* events, computed with respect to the periodicity of the associated recordings. We collect all the recordings, cluster them together into discrete events and order them on the number of distinct users aggregated on the discrete events; that is, for each periodicity, we compute the most programmed discrete events, and we propose them to the users. The introduction of the most popular events appreciably increased the total number of recordings (see Figure 5), as users have no more to search on EPGs or on newspapers the broadcasted programs: most of them are included in the given chart.

The collaborative filtering approach used in grouping users can be exploited to build a valuable dataset for the setting up of a recommendation engine [2]. That is, as users are aggregated together into groups with similar interests (i.e., users have recorded the same events) then the basic idea on which layering the recommendation is: what programs have been recorded and programmed in the future by users who share some interests?

In order to define such a similarity notion between users, we introduce a *similarity function* $S(u, v)$, that returns a similarity evaluation for the given couple of users (u, v), estimated on top of past recording activities (i.e., resources in common), programmed recordings (in the future) and similar recording behaviors. The *similarity network* among users is represented by a graph where nodes are users and there exists a weighted edge between u and v iff $S(u, v) > 0$.

Remember that we do not have any information about the proposed content, but the collaborative filtering approach allows us to group users on the basis of their personal interests, even without this knowledge. Of course this process can be improved with a feedback mechanism that allows users to rate the suggested programs, specifying if it was satisfactory or not.

4.1 Social Networking

We exploit our framework in the Facebook social network, with the development of a VCast application[6]. This application acts both as a front-end to the PVR and as the presentation layer for our recommendation algorithm (see Figure 2) [4].

Figure 2. Recommendation front-end in Facebook.

Moreover, exploiting the inner relationships built on the social network, we can make use of a more *personalized* recommendation service. We can thus distinguish from two distinct recommendation approaches, both followed by our framework. It is important to underline that (possibly) there is no relation between the friendship on the social network and the similarity in the VCast's users network.

Direct recommendation is made on top of the friendship on Facebook and is totally unbound from the similarity in the VCast network. Users who were previously *friends* on Facebook and share the VCast application, can openly recommend each other TV shows they like.

On the other hand, *indirect recommendation* is built on top of the VCast similarity network. User's interests and past recordings are collected and computed, in order to build a neighborhood set with users with similar interests. From that set, we can then extract shows that should be of any interest for the user.

5 Results

What we have reached so far can be summarized as follows:

- Search & discovery process;
- VCast community investigation by means of network analysis;
- Recommendations based both on the community and on user's interests starting from the output of the discretization process.

VCast's users activity is studied through the number of programmed recordings, the number of downloaded files and the number of active users. Figure 3 shows the activity on a system in a year over a weekly basis. Each point in the graph represents the number of recordings (square-dotted line) and the number of downloaded events

[6] `http://apps.facebook.com/vcastapp/`, Last Visited 19 Jan 2010.

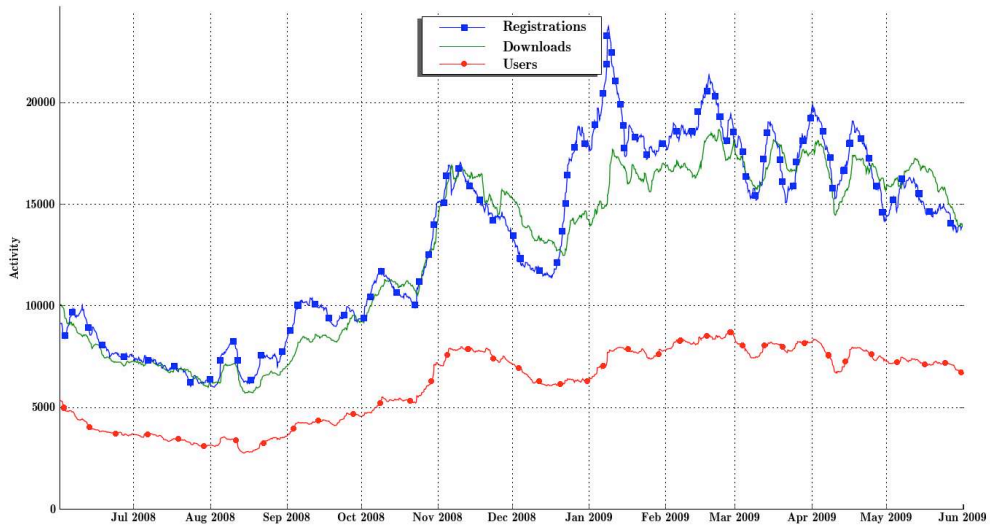

Figure 3. The activity of the system in the period: June 2008 - June 2009, expressed by means of the number of registrations, downloads and users.

occurred in the system in a 1-week interval. Just the same, each point of the circle-dotted line represents the number of active users on a 1-week interval.

As it turns out, this activity follows a constant behavior, alternating phases of increasing and decreasing programmed recordings (i.e., the peaks in Figure 3). Quite obviously, after a local maximum peak in recordings, follows a local maximum peak in the downloading activity. It is also interesting to notice that the curves decrease in the summer period, according to the reduced activity during the vacation time.

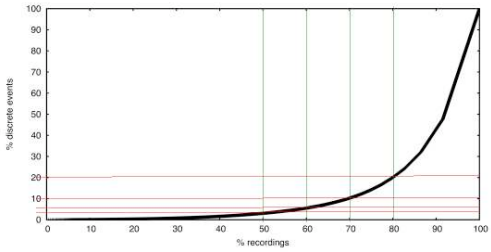

Figure 4. The long-tail effect (from 01/24/2008 to 05/18/2009: we identified 111.441 distinct events from 682.019 distinct recordings.)

As we expected, the relationship between the recordings and the identified discrete elements follows the so-called "long-tail" behavior (see Figure 4). Most recordings (about 80%) are relative to a well defined subset of the overall events (20%).

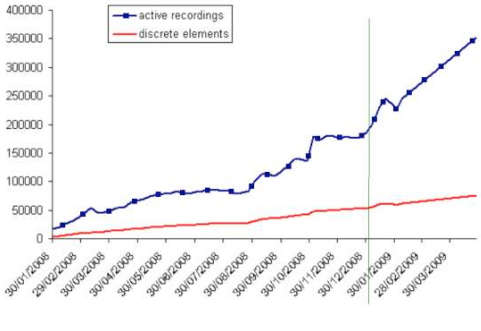

Figure 5. Since the most populars were showed on the web site, there is a significant increase in the number of distinct recordings.

This behavior is much more noticeable considering the timing in which we started showing the *most popular* recordings on the VCast web site. As Figure 5 shows, the number of recordings increased since the most popular introduction: it is an easy way to get all the relevant information on the most successful broadcasted shows, i.e., the most popular events.

The set of discrete events obtained by the discretization process is used as input for two different algorithms, devised to recommend

contents to the users of the system. In particular, the set of the *most frequently recorded* events is presented to the users as a community-based recommendation, by means of the *Most Popular* algorithm.

Similarly, a further recommendation scheme can also be devised from the discrete events, with the purpose of suggesting contents specifically tailored to users' interests. The algorithm, namely *Recommendation for Recordings* (Rec^2), exploits the well-known *collaborative filtering* approach of the *nearest neighbors* by identifing the k users $v_1, \dots v_n$ who can be considered most similar (with respect to a chosen similarity metric) to a specific user u, i.e., neighbors of u. From the elements of interest of such users, we can then identify those which can potentially be of any interest for u, thus recommendable to her [5].

In our tests, we chose the *Dice's similarity coefficient* to measure the similarity among the sets of items E_u, E_v watched by users u and v. Other similarity measures were also tried, however the obtained results are comparable with the Dice's one, which is defined as follows:

$$S(u,v) = \frac{2|E_u \cap E_v|}{|E_u| + |E_v|}$$

In the kNN algorithm, the weight of an element e_i for an user u_k can be defined as:

$$\mathrm{w}(u_k, e_i) = \sum_{u_a \in N(u_k)} \mathrm{r}(u_a, e_i) \cdot S(u_k, u_a),$$

$$\text{where} \quad \mathrm{r}(u_a, e_i) = \begin{cases} 1 & \text{if } e_i \in E_{u_a} \\ 0 & \text{if } e_i \notin E_{u_a} \end{cases}$$

E_{u_a} is the set of elements recorded by user u_a, whilst $N(u_k)$ is the neighborhood of user u_k, limited by considering only the top-N neighbors ordered by user similarity. We explored different variants of the scheme and the value $N = 300$ yelded to the best results.

In order to get an evaluation of our discretization process, we compute precision and recall of the Most Popular and Rec^2 algorithms. Starting from the Most Popular algorithm, we have to consider two sets: $V(u,t)$, i.e., the set of events registered by user u from time t onward, and $MP(n,t)$, i.e., the set of the n most popular events computed by our algorithm at time t. It must be clear that the Most Popular algorithm calculates $MP(n,t)$ based only on the events registered before time t, without any knowledge about the events $V(u,t)$ registered after time t. In this scenario, precision and recall for each user $u \in U$ are computed as:

$$Precision(u,n,t) = \frac{|V(u,t) \cap MP(n,t)|}{|MP(n,t)|}$$

$$Recall(u,n,t) = \frac{|V(u,t) \cap MP(n,t)|}{|V(u,t)|}$$

The precision and recall at time t are computed as the average of all users' precision and recall values. $Precision(n,t)$ and $Recall(n,t)$ represent the precision and recall at time t using the n most popular events. Finally, we compute the system precision and recall at different times, and calculate the system overall precision and recall as the average of it. $Precision(n)$ and $Recall(n)$ represent the system overall precision and recall using the n most popular events. At each run, our algorithm computes n of the most popular events, with n varying from 1 to 30, and the associated $Precision(n)$ and $Recall(n)$.

The way we compute precision and the recall for the Rec^2 algorithm is similar to computing precision and recall with the Most Popular algorithm. The main difference is that, while most popular events are given to all users, recommendation is personalized to each user. To this aim, thus, we consider the sets $V(u,t)$ as the set of events registered by user u from time t onward, and $Rec(u,n,t)$ as the set of n recommended items to user u at time t. It's important to notice that all recommended items are *in the future*, and also that the Rec^2 algorithm calculates $Rec(u,n,t)$ based only on the events registered before time t, without any knowledge about the events $V(u,t)$ registered after time t.

$$Precision(u,n,t) = \frac{|V(u,t) \cap Rec(u,n,t)|}{|Rec(u,n,t)|}$$

$$Recall(u,n,t) = \frac{|V(u,t) \cap Rec(u,n,t)|}{|V(u,t)|}$$

The way we calculate the system overall precision and recall is the same as the Most Popular algorithm, and $Precision(n)$ and $Recall(n)$ represent the system overall precision and recall using the n recommended events.

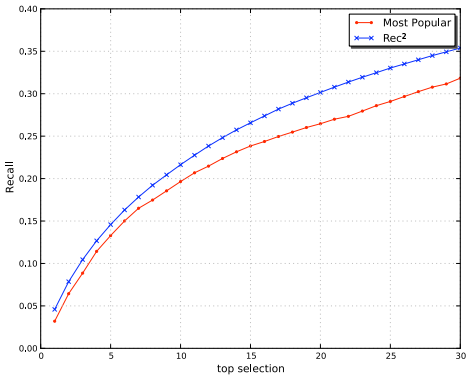

Figure 6. Recall for the recommendation algorithms.

In Figure 6 and 7, results are given. Recommending 30 events gives us more than 30% of recall with respect to the total recordings made by a single user. Ragarding precision, both the recommendation algorithms score better than 11% considering the first 5 elements, which can be considereded an adequate number of recommendable elements.

As it turns out, results are quite similar with the Most Popular algorithm and the Rec^2 engine, with the latter performing better in both precision and recall. This can be explained because of the lack of recommendable items due to what is depicted in Figure 4: most of users watch the same shows and nothing else. In order to retrieve previously unseen shows to be recommended we should navigate the similarity graph for farther nodes, but this lead to the widening of the set of considered items and thus increasing the risk of picking undesired content for the user.

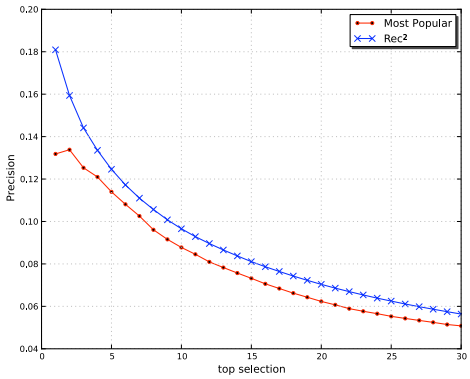

Figure 7. Precision for the recommendation algorithms.

6 Conclusion and Future Work

In this work, we present a framework for investigating the social aspects involved in the usage of a PVR application. Starting from a network analysis approach, we identify several clusters of users and sets of discrete events distinguishing the communities. Most of the users, however, tends to behave in a quite similar way, as the experience with the most popular recordings underlines.

The collaborative filtering approach gives us the opportunity to exploit an affinity function to group users with similar interests and try to predict and recommend what will be programmed for recording by the users. Still there are users who our recommendation engine is unable to recommend something to, meaning that there are no programmed recordings not already seen or programmed by that user in its neighborhood. This could lead to the identification of most hidden and isolated communities of users, unbiased from the most populars and very adherent to their *native* tendencies and interests.

Moving from the network analysis approach, it could be interesting to model this domain with a multiagent system. In such a viewpoint, users could be considered as agents which possess some *beliefs*, in the form of contents seen in the past and personal interests, and are trying to achieve specific *goals*, i.e., what they are going to program for the future. It is our intentions to investigate whether the existence of social relations among agents, modeled by means of direct (i.e., the social network) or indirect (i.e., the similarity network) interactions, could be exploited to support the agents in their activities, thus improving the overall user experience.

REFERENCES

[1] 'Free-to-air television and other PVR challenges in europe', Tech 3314, EBU-UER, (Jan 2006). `http://www.ebu.ch/CMSimages/en/tec_doc_t3314_tcm6-42262.pdf`.

[2] Gediminas Adomavicius and Alexander Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Trans. on Knowl. and Data Eng.*, **17**(6), 734–749, (2005).

[3] Alessandro Basso, Marco Milanesio, and Giancarlo Ruffo, 'Events discovery for personal video recorders', in *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pp. 171–174, New York, NY, USA, (2009). ACM.

[4] Francesca Carmagnola, Andrea Loffredo, and Giorgio Bernardi, 'Vcast on facebook: Bridging social and similarity networks', in *HT '09: Proceedings of the Twentieth ACM Conference on Hypertext and Hypermedia*, New York, NY, USA, (July 2009). ACM.

[5] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl, 'An algorithmic framework for performing collaborative filtering', in *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 230–237, New York, NY, USA, (1999). ACM.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky, 'Collaborative filtering for implicit feedback datasets', in *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 263–272, Washington, DC, USA, (2008). IEEE Computer Society.

[7] Michael Pazzani and Daniel Billsus, 'Content-based recommendation systems', *The Adaptive Web*, 325–341, (2007).

[8] Paul Resnick and Hal R. Varian, 'Recommender systems - introduction to the special section.', *Communication ACM*, **40**(3), 56–58, (1997).

[9] J. Ben Schafer, Joseph Konstan, and John Riedi, 'Recommender systems in e-commerce', in *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166, New York, NY, USA, (1999). ACM Press.