From Recordings to Recommendations: Suggesting Live Events in the DVR Context

Alessandro Basso, Marco Milanesio, André Panisson, Giancarlo Ruffo
Dipartimento di Informatica
Università degli Studi di Torino
Torino, Italy
{basso,milane,panisson,ruffo}@di.unito.it

ABSTRACT

Providing valuable recommendations in the DVR domain is quite straightforward when enough information about users and/or contents is known. In this work, we discuss the possibility of recommending future live events without knowing anything else but past user programmed recording schedules.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; J.4 [**Computer Applications**]: Social And Behavioral Sciences

General Terms

Algorithms, Experimentation, Human Factors.

Keywords

Digital Video Recorders, TV Broadcasts, Recommendation Systems, Collaborative Algorithms, Implicit Data

1. INTRODUCTION

A Digital Video Recorder (DVR) is a device aimed at recording digital streams to a storage. DVRs can be either hardware devices, such as set-top-boxes and portable media players, or software devices, such as web/PC-based Personal Video Recorders (PVRs), managing all user interactions and personalizations. By using DVRs, users are no longer bound to the broadcaster's schedule, but are free to define their personal lists of programs at any time.

In order to provide a better user experience by means of focused advices (e.g., recommendation of new contents), the arisen issues can be summarized in two main categories.
First, some logging activity must be done to find common usage patterns on which identify potential users' interests. Users are not willing to offer an *explicit profile* when using a DVR, thus we do have, possibly, only a set of observations on their activity. This is an important challenge for many known recommendation algorithms, that exploit user profiles for increasing accuracy and take into account privacy issues as well.
Second, differently from the Video on Demand domain, the usage of an Electronic Program Guide (EPG) is not always assured. This fact brings two consequences: (a) there is no knowledge on the content the user is recording and/or watching, and (b) there is no well defined one-to-one correspondence between a *recording* and a *broadcast event.* This leads to the impossibility of directly recommending recordings to users.

Taking into account these considerations brings us our research question: in such a domain, is it possible to give valuable live event recommendations to users, only considering their recording activity on the DVR? Users have to be brought to contents of interest, but, differently from other approaches, we are not using anything but collaborative filtering technique on users' activity. Thus, the main contribution of our approach is the demonstration that this can be achieved without any knowledge on what is being broadcast, neither EPGs nor content classifications.

2. RELATED WORK

The task of recommending live events, such as TV shows, has been already investigated in the past years. Proposed methods can exploit different ways to collect the required information for user profiling, as well as can make use of various recommendation algorithms. In particular, some approaches, such as [6], explicitly ask the users about their interests and build suggestions on top of the resulting user profiles. A different idea, which is adopted in several works [2, 9, 10], makes use of *implicit feedbacks*, i.e., information derived from the analysis of the user behavior while using the DVR. Other solutions, as [4, 12, 15], propose recommender systems which make use of user's view history as well as both *explicit* and *implicit feedbacks.* According to authors, such a mixed technique allows to obtain the best performance.

Another feature to tell apart existing methods for live events recommendation is the recommender algorithm used. A common approach relies on the content of the programs broadcast and it is therefore called *content-based.* Examples in this category can be found in [10, 12]. Some authors devised recommenders that make use of multiple content-based techniques, as in [3, 4].
A solution able to increase novelty of recommendations is

collaborative filtering, like the works in [2, 5]. Another interesting method is proposed in [9] and exploits the latent factor model.

In this work, we focus on implicit feedbacks only and we use a collaborative filtering approach to compute recommendations. Our aim is to minimize the information required as input of the recommender system, without sacrificing the novelty. The real challenge is to be able to recommend programs to users without actually knowing anything about what is broadcast on TV, since no EPG is used (differently from existing methods).

3. DATASET

Faucet is a PVR integrated in a podcasting service[1], which allows the recording and further downloading of Italian TV and Radio broadcasts [1]. The activity of the users is incrementally collected (hourly) into a log file containing the scheduled recordings set in the past hour as well as the occurred downloads. The resulting dataset is populated by real users expressing their preferences through the recorded programs. The dataset is publicly available at `http://secnet.di.unito.it/vcast`.
Each registered user can fix the desired settings for the recording of interest. At the end of the process, her recording is scheduled for the given time and will be further available for downloading purposes. Each recording r_i, thus, is defined as a tuple $< u_i, c_i, t_i, b_i, e_i, p_i >$ with the following notation: user u_i sets up a recording on channel c_i, starting from time b_i and ending at time e_i, with a title t_i and a periodicity p_i (e.g., once, every Tuesday, mon-fri). In Faucet, channels and periodicity values are fixed (users can choose their c_i and p_i from a combobox), while all other fields are completely up to the user.
After the end time expires, the recording is made available to the user for downloading. In case of periodic events, the recording step can occur an undefined number of times. After each recording step, the respective download is made available.

4. METHODOLOGY

In this section, we want to outline what our approach is. Given no knowledge on the broadcasts, we collect the users activity to compute what we call *discrete events*, to be used for recommendation purposes and top chart list building.

4.1 From Recordings to Events

The extraction of meaningful information from the unstructured amount of data contained in the dataset is essential to define a set of events which map the broadcast programs. Through the *event discovery* phase, we can discretize the continuous domain of timings defined by the recordings, creating the basis for the application of a recommender algorithm. The basic procedure used in the discretization was first introduced in [1] and covers a number of subsequent steps:

Clustering. Recordings are clustered together by considering the *channel*, the *periodicity* and the difference between *starting* and *ending* times. All recordings belonging to the same cluster are thus equal as channel and periodicity, whilst similar on timings. Specific values are used to define the maximum clustering distance for the start and the end times. The output of this activity is a set of clusters, each identifying a single event. The centroid of the cluster, i.e., the recording that minimizes the intra-cluster timing distances, is considered the representative of the event.

Aggregation. As the clustering occurs periodically, this operation aims to identify newly created events characterized by the same channel and periodicity of the formerly created ones, but comparable timings. Such elements refer to the same programs and are therefore merged into unique events, whose properties are updated by taking into account the values of all the similar ones.

Collapsing. A further refinement phase is required to grant the consistency of the generated events. In fact, due to the high variability of timings, especially when a new transmission appears, events which are initially considered as non referring to the same transmission tend to slowly and independently converge to more stable timeframes. This implies the need of merging them into single events.

As a result of the processing phase, given a set of recording clustered together, each one with the same channel c_i and the same periodicity p_i, we compute a *discrete event* e_i in the form of: $< \{u_i\}, \bar{t}, c_i, \bar{b}, \bar{e}, p_i >$, where $\{u_i\}$ is the set of users whose recordings were clustered together; $\bar{t}$ is the user generated title most frequent among users in $\{u_i\}$; $\bar{b}$ and $\bar{e}$ are, respectively, the starting and ending time computed as the median value of all the clustered recordings.

4.2 From Events to Recommendations

When future events are computed from scheduled recordings, we are thus able to propose them to users by means of two different charts: (1) a global chart returning those events computed starting from the largest groups of recordings, i.e., those chosen by the largest sets of users; and (2) a user-based recommendation list, returning a set of new events of possible interest to each user requesting it, computed through a similarity function over the whole population. We call them *Most Popular* and *Rec*^{2} *(Recordings times Recommendations)*, respectively. Both charts are computed by means of the memory based *collaborative filtering* approach named k-Nearest Neighbors (kNN) [14]. We apply both variants of the kNN algorithm: the user-based one [8], by identifying users interested in similar contents; and the item-based approach [7], by focusing on items shared by two or more users.

In kNN, the *weight* (i.e., a measure of interest) of an element e_i for an user u_k can be defined as:

$$\mathrm{w}(u_k, e_i) = \sum_{u_a \in N(u_k)} \mathrm{r}(u_a, e_i) \cdot \mathrm{c}(u_k, u_a), \qquad (1)$$

where $N(u_k)$ are the neighbors of user u_k and $\mathrm{r}(u_a, e_i)$ is equal to 1 if user u_a is associated to the event e_i, and 0 otherwise. The coefficient $\mathrm{c}(u_k, u_a)$ represents the neighbor's information weight for user u_k. In most of the kNN-based algorithms [8], the coefficient used is the similarity between u_k and u_a.

[1]`http://www.vcast.it/`

Most Popular. The *MostPopular* algorithm can be defined by means of eq. (1), assuming the number of neighbors unbounded, which implies $N(u_k) = U$, $\forall u_k \in U$; and $\mathrm{c}(u_a, u_b) = 1$, $\forall u_a, u_b \in U$, with U as the set of all users. Thus, the weight is modified as $\mathrm{w}(u_k, e_i) = \sum_{u_a} \mathrm{r}(u_a, e_i)$.

After calculating the weight of all elements, they are sorted in descendant order. In the *MostPopular* algorithm, as the set of neighbors is independent of the user, all users receive the same recommended elements, i.e., the most popular ones.

Rec^2. In order to provide personal suggestions, we have to define a similarity function for grouping similar users (items) from which choosing the appropriate elements to recommend. Our definition of similarity is based only on implicit feedbacks, resulting from observing the behavior of users: if she records something, then we assume that she is interested in it; otherwise, we can not infer anything about the interest of the user for that element. We are therefore considering binary feedbacks.

Given two users u and v and the associated discrete events E_u and E_v, we can choose the similarity metric, $S(u, v)$, considering several well known measures (e.g., Dice, Cosine and Matching) [11]. After choosing a metric, $\forall u$ we can compute the subset $N_u \subseteq U$ of *neighbors* of user u. A user v such that $E_v \cap E_u \neq \emptyset$ is thus defined as a neighbor of u. Starting from the neighborhood of u, the similarity with u is computed for each pair $< u, v >$ such that $v \in N_u$. Finally, if $S(u, v) > 0$, we consider u similar to v. The value $S(u, v)$ is used to weight such a relation, therefore determining a similarity order among the neighborhood of u, from which choosing new events to recommend to u.

Similarly, this approach can be adopted for the item-based similarity: two events are considered similar if the *share* at least a single user that is associated to both of them.

5. EVALUATION

In the following, we evaluate the obtained results in the event extraction process and in the recommendation of new events to users, both in Most Popular and in Rec^2.

5.1 Event Extraction

As a remainder, we are dealing with several independent, user generated recording schedules, that we cluster together and from which we compute the discrete events. In Figure 1, a view of the distribution of the recordings is given: for each detected event, the number of recordings clustered together changes according to users' activity. As it turns out, most recordings (and, thus, most users) tend to be clustered and aggregated on very few events, while there are lots of events with very few recordings. The Most Popular algorithm exploit these inner features of the resulting discrete events to compute the top chart.

5.2 Computing Recommendation

We measure how accurate is the recommendation in predicting the elements that users would program in terms of recall. These values are computed as the average of all users' recall values using the top n recommended elements [13].

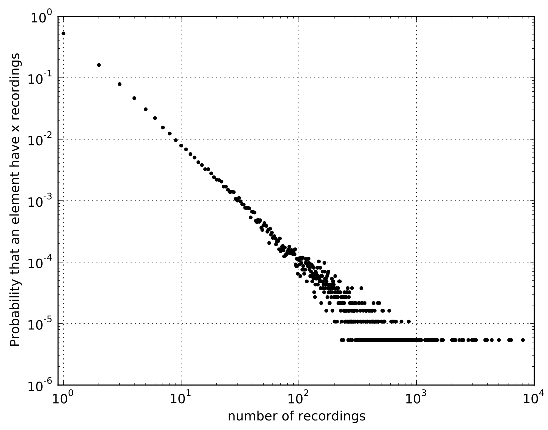


Figure 1: Number of recordings per event (Probability density function)

We are giving particular emphasis on the recall measure; in fact, since we do not have explicit feedbacks regarding the user's interest in those items which have not been considered (i.e., not programmed, nor downloaded), precision is not very meaningful [9].

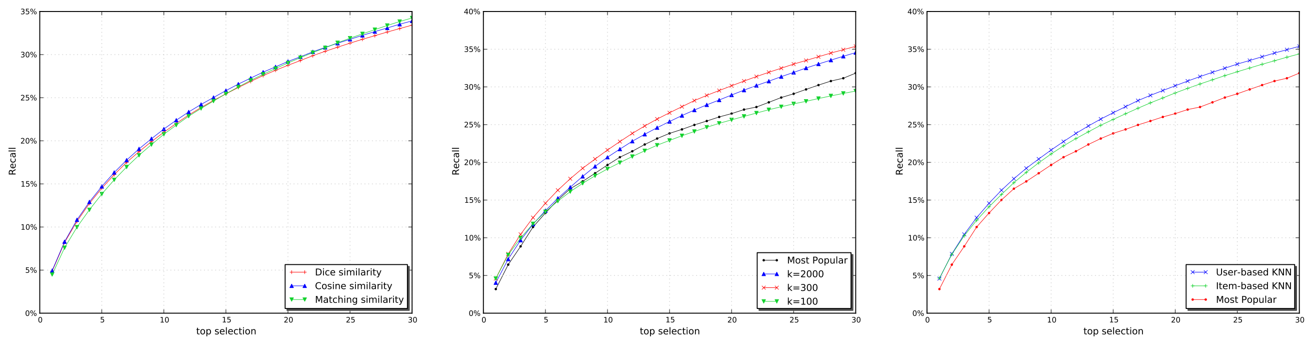
First, we choose different similarity functions to understand whether similarity influences the results of the user-based kNN algorithm. From Figure 2(a) it is clear that, in this case, all chosen similarity metrics show nearly the same performance.

The second step is to find the optimal value for k. Figure 2(b) shows the results with $k \in \{100, 300, 500, 700, 2000\}$ in user-based kNN (Dice similarity), and the *MostPopular* recommender. We omit the values of $k = \{500, 700\}$ since the results are almost equal to $k = 300$. Compared to the *MostPopular* algorithm (i.e., unbounded neighbors), a value $k = 100$ is not enough to outperform it, whilst for $k = 2000$, kNN starts to converge to it. Considering the top 10 recommended elements, we can achieve the best results for $k = 300$, whilst $k = 500$ is more suitable when taking further elements. As in most cases 10 elements are sufficient for a recommendation, $k = 300$ offers a good trade-off between valuable recommendations and resource consumption for building the neighborhood.

A comparison among user/item-based kNN and *MostPopular* is depicted in Figure 2(c). We can observe that the latter is clearly outperformed by the other two algorithms, especially when more than 7 recommended items are considered. The user-based algorithm performs slightly better than the item-based one (more noticeable with more than 15 recommended items). In general, item-based algorithms tend to perform better because usually the number of items is considerably lower than the users [14], but this property does not hold in our domain.

6. CONCLUSION

In this paper we show how to recommend live events to users without any knowledge about the broadcast content



(a) Comparison between similarity functions in user-based kNN

(b) Recall for user-based kNN

(c) Recall for kNN ($k = 300$) wrt Most-Popular

Figure 2: Comparison between recommenders and recall for kNN and Most Popular.

and user's likes. Recommendations can be given both globally and personally. It is important to underline that the most popular events are easier to predict since users tend to naturally focus on them, even without any specific suggestion. On the contrary, granting a high novelty in personal recommendations is a more challenging goal due to the reduced amount of explicit information. Nevertheless, we can obtain interesting results even exploiting a simple approach as the kNN. We are currently attempting other approaches to recommendation (e.g., latent factor model) with implicit feedbacks, with the aim of improving the prediction accuracy.

7. REFERENCES

[1] A. Basso, M. Milanesio, and G. Ruffo. Events discovery for personal video recorders. In *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 171–174, New York, NY, USA, 2009. ACM.

[2] P. Baudisch and L. Brueckner. Tv scout: Guiding users from printed guides to personalized tv program. In *In Proceedings of the 2nd Workshop on Personalization in Future TV (May 28, Malaga, Spain), Universidad de Malaga*, pages 151–160, 2002.

[3] Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, M. Ramos-Cabrer, B. Barragáns-Martínez, M. López-Nores, J. García-Duque, A. Fernández-Vilas, and R. P. Díaz-Redondo. A multi-agent open architecture for a tv recommender system: A case study using a bayesian strategy. *Multimedia Software Engineering, International Symposium on*, pages 178–185, 2004.

[4] A. L. Buczak, J. Zimmerman, and K. Kurapati. Personalization: Improving ease-of-use, trust and accuracy of a tv show recommender. In *in Proceedings of the TV'02 workshop on Personalization in TV, Malaga*, pages 3–12, 2002.

[5] P. Cremonesi and R. Turrin. Analysis of cold-start recommendations in iptv systems. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 233–236, New York, NY, USA, 2009. ACM.

[6] D. Das and H. ter Horst. Recommender systems for tv. In *In Proceedings of AAAI*, 1998.

[7] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[8] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. 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*, pages 230–237, New York, NY, USA, 1999. ACM.

[9] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[10] S. G. Kaushal, S. Gutta, K. Kurapati, K. Lee, J. Martino, J. Milanski, J. D. Schaffer, and J. Zimmerman. Tv content recommender system. In *In Proceedings of the 17th National Conference on Artificial Intelligence*, pages 1121–1122. AAAI Press / The MIT Press, 2000.

[11] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 641–650, New York, NY, USA, 2009. ACM.

[12] M. Rovira, J. Gonzàlez, A. López, J. Mas, A. Puig, J. Fabregat, and G. Fernandez. Indextv: a mpeg-7 based personalized recommendation system for digital tv. In *ICME*, pages 823–826. IEEE, 2004.

[13] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *In ACM WebKDD Workshop*, 2000.

[14] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. 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.

[15] K. K. Srinivas, S. Gutta, D. Schaffer, J. Martino, and J. Zimmerman. A multi-agent tv recommender. In *In Proceedings of the UM 2001 workshop "Personalization in Future TV"*, 2001.