Time-Evolution of IPTV Recommender Systems

Paolo Cremonesi
DEI - Politecnico di Milano
p.zza Leonardo da Vinci, 32 - 20133 Milano, Italy
Phone: +39 022399-3517
paolo.cremonesi@polimi.it

Roberto Turrin
Neptuny
via Durando, 10 - 20158 Milano, Italy
Phone: +39 022399-3482
roberto.turrin@polimi.it

ABSTRACT

In this paper we evaluate the performance of different collaborative filtering algorithms over time, where new users, new items, and new ratings are constantly added to the recommender dataset.

The analysis has been performed on the datasets collected by two IPTV providers. Both datasets have been implicitly collected by analyzing the pay-per-view movies purchased by the users over a period of several months. The first result of the paper outlines that item-based algorithms perform better with respect to SVD-based ones in the early stage of the cold-start problem. The second result shows that the accuracy of SVD-based algorithms, when using few latent factors, decreases with the time-evolution of the dataset. On the contrary, SVD-based algorithms, when used with a large-enough number of latent features, increase their accuracy with time and may outperform the item-based algorithms if the dataset does not present a long-tail behavior.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*user profiles and alert services; performance evaluation (efficiency and effectiveness)*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Recommender systems, collaborative filtering, cold-start, time evolution

1. INTRODUCTION

Recommender systems (RS) can play an important role within the Interactive TV domain (ITV), where the presence of huge catalogs of *items* (e.g., movies, TV shows) dramatically reduces the visibility of each one, potentially inhibiting users from finding interesting TV contents [1]. Recommendations are done by tracking the users' activity and by selecting a suitable on-demand TV content for the individual viewer [29].

Personalized systems that recommend TV programs according to the user's preferences have been already experimented. Netflix is an online TV provider and DVD rental service that, in 2006, began a competition with a one million dollar prize for a RS [4]. WeOnTV is a social TV application that allows users to know what others are viewing and to make recommendations [1]. LIVE (Live Staging of Media Events) is a IST-FP6 project that is developing a system for collecting feedback from IPTV users [29]. PersonalTV is a recommendation application for user-generated video content incorporated in Facebook [10].

Recommender algorithms recommend items similar to the ones the user liked in the past and can be classified into two families [16, 2, 28, 27]: *content-based filtering* (CBF) finds similarity between items on the basis of the items content (e.g., gender, director, actors, plot) while *collaborative filtering* (CF) finds similarity between items on the basis of collaborative information about users, i.e., they use the opinions (known as *ratings*) expressed by the community of ITV users. The most used RSs are based on collaborative filtering, the biggest advantage over content-based systems being that collaborative filtering relies only on explicit or implicit opinions expressed by users while explicit content description is not required [19].

To be able to provide recommendations, collaborative systems must derive users' preferences toward items, which is done through analysis of viewers' past interaction with the TV programs. The RS must thus track what the user are watching, in other words it must collect viewers' ratings. However, collaborative filtering requires a large number of ratings before they can make reasonable suggestions.

When bootstrapping a new collaborative RS, the average number of ratings per user and item is low and this can significantly degrade the quality of the algorithms. This problem is called the *cold-start* problem and refers to situations where there are only few ratings to base recommendations on. Moreover, RSs evolve over time, getting more users, more items, and collecting more ratings. At each point in time we can distinguish between *old* (i.e., existing) and *new* users, as well as between *old* and *new* items. At any given time we face with:

- *new users*: when users first register with the RS very few ratings are available to describe their profile.
- *new items*: when items are added to the catalog they have no ratings.

The issue of evaluating the quality of collaborative RSs over time has not been extensively covered in the literature. The lack of evaluations is largely due to the characteristics of publicly available

EuroITV'10, June 9–11, 2010, Tampere, Finland.

datasets, all of which have a substantial large density and fairly long user profiles with respect to many real-life applications. Moreover, few of these datasets provide the additional information about when the ratings have been collected. When this information is missing, the only approach available to simulate the cold-start problem requires to randomly sub-sample the dataset [17].

Previous works mainly attempted to combine collaborative with content-based recommendation approaches to address the new-item problem [13, 21, 25, 26]. Other works focused on the design of new collaborative filtering algorithms that, compared with state-of-the-art algorithms, are able to improve the performance on data-sparse domains without excessively worsening the performance when data is plentiful [3, 17]. Few works addressed the *new user* problem [20].

In this paper we describe two collaborative filtering algorithms for ITV systems and we analyze their time evolution comparing the quality of the recommendations against a non-personalized recommendation method.

The evaluation has been performed on the datasets collected by two IPTV providers having, respectively, 200,000 and 600,000 subscribers [9]. The two datasets (TV1 and TV2, respectively) have been implicitly collected by analyzing the pay-per-view movies purchased by the users over a period of several months and are available for download from **http://memo.elet.polimi.it**.

This paper extends previous works with the following improvements:

- RSs evolve over time, always getting more users and more items. Therefore, at each point in time the system suffers from some cold-start issues. In this paper we do not limit the discussion to the initial stage of a RSs, but we present a more general testing methodology that allows us to analyze the evolution of RSs over time.

- Many collaborative filtering algorithms adopt a computationally efficient approach by first developing a model of user ratings based on a training dataset. The model is later used to recommend items to users. Building a model can be time consuming, therefore the model is not updated in real-time with every new rating. Depending on the dataset size, the time between model updates can vary in the hours or days range. In this paper we analyze the aging of a model, i.e., we analyze if and how the quality of a RS is affected by using an "old" model to recommend items to users.

- Some ITV environments exhibit a long-tail distribution, with few items accounting for most of the ratings, while other ITV systems present a more fair distribution of ratings among items. The time evolution of recommender algorithms may differ according to the tail distribution of the ratings [7]. Such distribution change with time, according to a cyclic fashion: during promotional periods (e.g., , Christmas) the rating distribution is pushed toward popular items. In this paper we analyze the time evolution of the algorithms with respect to the item popularity.

The rest of the paper is organized as follows. Section 2 describes the two collaborative algorithms developed and tested, Section 3 formalizes the time evolution properties of a RS, Section 4 describes the testing methodology and Section 5 presents and discusses the results of the tests. Finally, Section 6 draws the conclusions and lead on some possible future work.

2. TESTED ALGORITHMS

Recommender algorithms can be classified into content-based and collaborative algorithms.

Content-based algorithms are based on the analysis of the content of items (e.g., genre, actors, directors). The algorithm proposes items with a content similar to the content of items that the user liked in the past.

On the other hand, collaborative algorithms suggest items to a particular user on the basis of the other-users' ratings (i.e., preferences). Collaborative algorithms have reached a much more interest with respect to content-based algorithms, mainly because: they are applicable in every domain [15], they promise better quality than content-based systems [14], and they have been promoted by the Netflix contest [4]. However, since collaborative algorithms are based on the user ratings, they are particularly affected by the *cold-start* problem, being not able to provide accurate recommendations at the early stages of a RS life-cycle when only few ratings have been collected.

In the following we describe the two collaborative algorithms tested in the evaluation of the cold-start problem: an item-based and a dimensionality-reduction algorithm. The algorithms' input is a $n \times m$ user-rating matrix (URM), that we refer to as $\mathbf{R}$, where n and m are, respectively, the number of users and the number of of items. The element r_{pi} represents the rating of user p on item i. Since we deal with implicit, binary datasets collected by IPTV operators, r_{pi} can be either 1 or 0, according to the case user p has watched or not item i, respectively.

2.1 Item-based algorithm

Item-based collaborative algorithms capture the fundamental relationships among items [23]. Two items are related if the community agrees about their ratings. Such relationship can be represented in a $m \times m$ matrix, referred to as $\mathbf{D}$, where the element d_{ij} expresses the similarity between item i and item j. Note that, potentially, $\mathbf{D}$ could be non-symmetric, i.e., $d_{ij} \neq d_{ji}$.

Matrix $\mathbf{D}$ represents the model of the RS and its calculation, being computational intensive, is generally delegated to a batch process.

When using implicit datasets, similarity metric is usually computed using a frequency-based approach, as the one discussed by Deshpande and Karypis in [11].

The element d_{ij} of matrix $\mathbf{D}$ can be computed with the classical *cosine similarity* among binary vectors:

$$d_{ij} = \frac{\#(i,j)}{\sqrt{\#(i)} \cdot \sqrt{\#(j)}} \tag{1}$$

where $\#(i,j)$ is the number of users that have watched both item i and item j, and $\#(i)$ is the number of users that have watched item i.

The model can been further enhanced by means of a kNN (k-nearest-neighborhood) approach. For each item (i.e., column of $\mathbf{D}$), we consider only the k most similar items (referred to as the item's neighborhood). The kNN approach discards the noise of the items poorly correlated to the target item, improving the quality of recommendations.

At real-time, given the ratings of the target user p to recommend, we can predict the unknown rating $\hat{r}_{pi}$ by summing up the similarities between item i and the items watched by user p (i.e., any item j such that $r_{pj} = 1$)

$$\hat{r}_{pi} = \sum_{j \in r_{pj}=1} d_{ji} \cdot r_{pj} \tag{2}$$

2.2 Dimensionality-reduction algorithm

Collaborative algorithms based on dimensionality–reduction techniques (that we also refer to as SVD-based algorithms) describe users and items by means of a limited set of hidden *features*.

Let us assume that items and users can be described by means of l features in a l-dimensional feature space. The correlation between user p and item i can be computed as:

$$\hat{r}_{pi} = \sum_{e=1}^{l} a_{pe} \cdot b_{ie} \tag{3}$$

where, a_{pe} and b_{ie} are the e-th (unknown) features for user p and item i, respectively.

There exist several techniques for computing the hidden features that minimize a given prediction error [22]. We have based our analysis on the singular value decomposition (SVD) (e.g., [8, 12, 18, 24]). By means of SVD, the URM can be factorized as

$$\hat{\mathbf{R}} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^{\mathrm{T}} \tag{4}$$

where, $\mathbf{U}$ is a $n \times l$ orthonormal matrix, $\mathbf{V}$ is a $m \times l$ orthonormal matrix, and $\mathbf{S}$ is a $l \times l$ diagonal matrix containing the first l singular values, sorted in decreasing order.

Assuming that $\mathbf{u}_p$ represents the p-row of $\mathbf{U}$ and $\mathbf{v}_i$ the i-row of $\mathbf{V}$, the prediction $\hat{r}_{pi}$ can be computed as

$$\hat{r}_{pi} = \mathbf{u}_p \cdot \mathbf{S} \cdot \mathbf{v}_i^T \tag{5}$$

Since $\mathbf{U}$ and $\mathbf{V}$ have orthonormal columns, we can derive that $\mathbf{u}_p \cdot \mathbf{S} = \mathbf{r}_p \cdot \mathbf{V}$, where $\mathbf{r}_p$ is the p-th row of $\mathbf{R}$ (i.e., the profile vector of user p). Consequently, (5) can be reformulated as

$$\hat{r}_{pi} = \mathbf{r}_p \cdot \mathbf{V} \cdot \mathbf{v}_i{}^{\mathrm{T}} \tag{6}$$

By means of (6) we are able to predict at real–time the ratings of any items not watched by user p.

Note that, similarly to the item-based algorithm, (6) represents a model-based approach, where the model is $\mathbf{V}$. This is a great advantage if compared, for instance, with regularized SVD where user features need to be pre–computed (e.g., [22]). In fact, with regularized SVD we are not able to recommend new users, since we first need to build a model for such users.

Furthermore, regularized SVD is based on RMSE (round mean square error) and designed to minimize it, while in the case of implicit, binary datasets RMSE can not be computed. In addition, when pursuing recommendation tasks such as the top-N task (e.g., [11, 15]), classical SVD can outperform regularized SVD (e.g., [5]).

3. DATASET TIME-EVOLUTION

The dataset evolution is driven by three inputs:

old users: existing users watch existing movies in the catalog;

new users: new users join the system (i.e., new or existing users watch their first movie since the RS is running);

new items: new items are added to the catalog (i.e., at least one user watches for the first time an existing movie or a new movie that is added to the catalog).

We refer to the *dataset density*, the *average user profile length*, and *the average number of views per item* as, respectively, the percentage of ratings with respect to the dataset size, the average number of movies viewed by a user, and the average number of users that have watched an item.

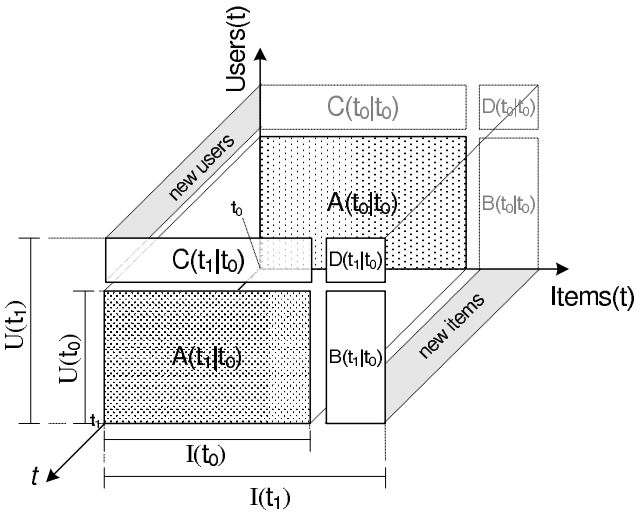


Figure 1: Decomposition of the URM representing the time-evolution of the dataset in the interval $[t_0, t_1]$.

Thus, the first input has the effect of increasing: the dataset density, the average user profile length, and the average number of views per item. The second input has the effect of decreasing both the dataset density and the average user profile length, as the new users that join the system have watched one movie. Similarly, the third input has the effect of decreasing both the dataset density and the average number of views per item.

The overall evolution of the dataset parameters (density, user profile length, and number of views per item) depends on the relative rates of the three input factors. Given a generic time instant t, let us define $\mathbf{I}(t)$ as the set of active items, i.e., items that have been rated by at least one user. Similarly, let us define $\mathbf{U}(t)$ as the set of active users, i.e., users that have rated at least one item. If we consider two time instants t_0 and t_1, with $t_0 < t_1$, the URM at time t_1 can be partitioned into the following 4 sub-matrices (Figure 1) that describe the URM time-evolution in the interval $[t_0, t_1]$:

- $\mathbf{A}(t_1|t_0)$ containing the ratings (i.e., samples) collected up to time t_1 and concerning users and items active at time t_0 (i.e., old users and old items). Formally, we refer to these users and items as $\mathbf{U}(t_0)$ and $\mathbf{I}(t_0)$, respectively.

- $\mathbf{B}(t_1|t_0)$ containing the ratings collected in the interval $[t_0, t_1]$ by users active at time t_0 (i.e., old users) about items not active at time t_0 (i.e., new items). Formally, we refer to these users and items as $\mathbf{U}(t_0)$ and $\mathbf{I}(t_1) \backslash \mathbf{I}(t_0)$, respectively[1].

- $\mathbf{C}(t_1|t_0)$ containing the ratings collected in the interval $[t_0, t_1]$ by users not active at time t_0 (i.e., new users) about items active at time t_0 (i.e., old items). Formally, we refer to these users and items as $\mathbf{U}(t_1) \backslash \mathbf{U}(t_0)$ and $\mathbf{I}(t_0)$, respectively.

- $\mathbf{D}(t_1|t_0)$ containing the ratings collected in the interval $[t_0, t_1]$ and concerning users and items not active at time t_0 (i.e., new users and new items). Formally, we refer to these users and items as $\mathbf{U}(t_1) \backslash \mathbf{U}(t_0)$ and $\mathbf{I}(t_1) \backslash \mathbf{I}(t_0)$, respectively.

Note that $\mathbf{A}(t_0|t_0)$ is the URM at time t_0, and that $\mathbf{B}(t_0|t_0) = \mathbf{C}(t_0|t_0) = \mathbf{D}(t_0|t_0) = \emptyset$.

[1] $\mathbf{X} \backslash \mathbf{Y}$ refers to the difference between set $\mathbf{X}$ and set $\mathbf{Y}$.

3.1 Cold-start vs random sampling

Many works in the literature (see, e.g., [17]) simulate the cold-start phase of a recommender system by randomly sub–sampling the set of ratings. However, we have observed that this approach introduces a number of anomalies in the dataset:

- User profiles are split in an unnatural way. For instance, if a user has watched all the movies of a film series, typically these movies have been watched in the correct sequence. Random sampling of the URM to simulate the cold-start problem may create non-realistic ordering of the movies in the film series.
- The rating frequency of an item is not stationary. Movies have a peak of views (i.e., implicit ratings) during the first days since they have been inserted in the catalog. After the initial burst, the viewing frequency decreases with time.

As an example, in order to simulate the time evolution of a dataset, we have sub–sampled the dataset of the first interactive TV providers, TV1, by *randomly timestamping* the implicit ratings (views). Figure 2 shows the statistical properties of the sub-sampled dataset over time: the number of views and the dataset density (a), the average number of views per item and per user (b), the number of active users and of active items (c).

If we compare such random evolution with the real dataset time–evolution shown in Figure 3, the most noticeable differences are in the evolution of density and number of items. This happens because the TV1 dataset has a fairly large number of ratings per item. Thus, the random sub–sampling of ratings is not able to reduce the number of items till the dataset density has been greatly reduced. The final effect shown in Figure 2(c) is that the random selection mechanism simulates a cold-start problem in which the number of items is almost constant. However, within a real cold-start problem, the rate at which new items are added to the catalog is more regular with time, as illustrated in Figure 3(c).

4. TESTING METHODOLOGY

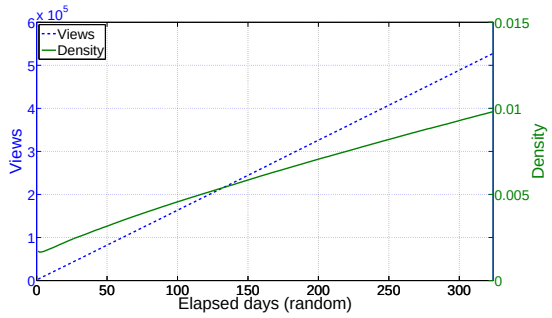
In this paper we are going to investigate:

- how the algorithms learning rate evolves with time (i.e., with increasing user profile length, increasing views per items, increasing number of users and increasing number of items);
- if and how the algorithm parameters should be tuned when the dataset evolves with time.

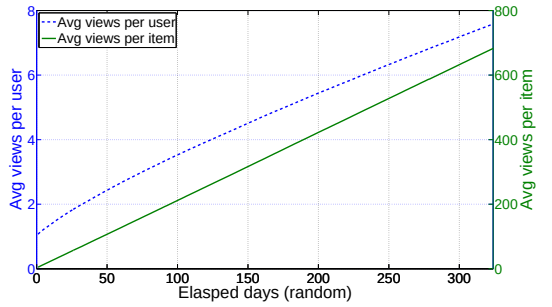
Typically the quality of RSs is evaluated by means of either error metrics (e.g., RMSE and MAE) or classification accuracy metrics (e.g., recall, precision, and fall-out) [15, 8]. Since in our case the available ratings are binary, error metrics can not be computed. Therefore we present the results in terms of an accuracy metric: the *recall*. Recall expresses the percentage of relevant movies recommended to users.

Let us define a test by means of three sets, each one composed by pairs of user–item (referred to as *samples*) that identify a subset of URM's ratings:

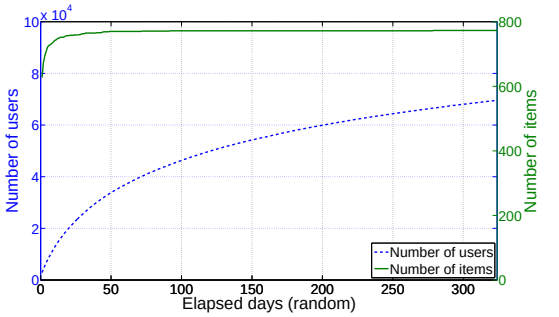
- The train set $\mathbf{M}$ contains the samples used to train the algorithm (i.e., to build the model).
- The profiles test set $\mathbf{P}$ contains the samples representing the profiles of users taken into consideration during the tests.
- The test set $\mathbf{T}$ contains the samples used to compute the accuracy metrics.



(a) URM: collected ratings and density



(b) Avg views per item and per user



(c) Number of users and items

Figure 2: TV1 dataset with *random timestamps*: number of ratings and density of URM (a), average views per item and per user (b), number of users and items (c).

Note that items not represented by the model (i.e., with no samples in $\mathbf{M}$) and users not represented by the profiles test set (i.e., with no samples in $\mathbf{P}$) can not be tested (i.e., they can not be in $\mathbf{T}$).

According to the way $\mathbf{M}$, $\mathbf{P}$ and $\mathbf{T}$ are defined, we can analyze different aspects of the time evolution of a recommender algorithm, as described in Table 1 and Figure 4.

- **Static** tests evaluate the quality of a recommender algorithm at time t using all active items. It does not exploit the dataset time-evolution.
- **Future** tests estimate the percentage of items that have been watched by users during a Δt time windows after they were recommended.
- **Incremental** tests evaluate the quality of a recommender algorithm on new ratings collected during a Δt time windows after the algorithm was trained. This test captures the capability of an 'old' model to provide good recommendations. We further distinguish this capability between:

Test	$\mathbf{M}$	$\mathbf{P}$	$\mathbf{T}$
Static	$\mathbf{A}(t\|t)$	$\mathbf{A}(t\|t)$	$\mathbf{A}(t\|t)$
Future	$\mathbf{A}(t\|t)$	$\mathbf{A}(t\|t)$	$\mathbf{A}(t+\Delta t\|t) \backslash \mathbf{A}(t\|t)$
Old incremental	$\mathbf{A}(t\|t)$	$\mathbf{A}(t+\Delta t\|t) \bigcup \mathbf{B}(t+\Delta t\|t)$	$\mathbf{A}(t+\Delta t\|t) \backslash \mathbf{A}(t\|t)$
New incremental	$\mathbf{A}(t\|t)$	$\mathbf{C}(t+\Delta t\|t) \bigcup \mathbf{D}(t+\Delta t\|t)$	$\mathbf{C}(t+\Delta t\|t)$

Table 1: Test: formalization of train set $\mathbf{M}$, profiles test set $\mathbf{P}$, and test set $\mathbf{T}$.

- *old incremental tests* evaluate the capability of an 'old' model to provide good recommendations to old viewers (i.e., viewers whose old profiles have been used to build the recommender model and who now have enriched profiles);
- *new incremental tests* evaluate the capability of an 'old' model to provide good recommendations to new users (i.e., users whose profiles have not been used to build the model).

According to the k-fold methodology described in [5] and [9] the URM is divided into $k = 10$ folds, where in turn one fold is used to form sets $\mathbf{P}$ and $\mathbf{T}$, and the other folds to form set $\mathbf{M}$:

1. We first train the algorithm by means of $\mathbf{M}$.
2. For each sample t_{ij} (i.e., rating) in $\mathbf{T}$, we consider the related user profile in $\mathbf{P}$ (i.e., the i-th row). Thereafter we create a modified user profile by hiding the rating p_{ij} from such user profile. Recommendations are then generated on the basis of such a modified user.
3. If the hidden item (i.e., item j) is recommended to the user within the first N positions we have a *hit*. In fact, it means that the algorithm recommended an item that the user effectively watched. In our tests we used $N = 5$.

The recall is computed as the percentage of hits with respect to the number of tests.

4.1 Tail distribution

Increasing the novelty of recommendations is an important key factor for IPTV providers, who are interested in augmenting the visibility and the sales of long-tail movies (i.e., the strategy of selling a large number of movies in relatively small quantities) since they represent a costly investment (copyrights, hardware infrastructure, storage, etc.).

In this section we describe a way to further refine the definition of the test set $\mathbf{T}$ in such a way that we are able to evaluate the ability of RSs to recommend novel movies.

First of all, we need to analyze movies popularity. In Figure 5 movies are ranked along the vertical axis - the most popular at the bottom - while the horizontal axis represents the cumulative percentage of ratings accounted by the most popular items.

According to the approach described in [6], we have partitioned the movies into three sets: *short-head* items are the most-popular movies that account for 33% of the views, *long-tail* items are the less-popular movies that account for 33% of the views and *mid* items are the remaining movies.

The analysis on the long-tail distribution of the TV1 dataset (solid line) shows that the short-head is accounted by 8% of the most popular movies. Since the total number of items in the TV1 dataset increases with time from 300 up to almost 800 movies, the TV1 short-head contains between 25 and 60 popular movies. As for the TV2 dataset (dashed line), the short-head is accounted by 2.3% of the most popular movies. Since the total number of items in the TV2 dataset increases from 1300 up to 3400 movies, the TV2 short-head contains between 30 and 80 popular movies.

By comparing the TV1 and TV2 tail distributions we observe that TV2 users tend to prefer popular movies. We will see in the next section that this behavior pushes item-based algorithms to provide better recommendations with respect to dimensionality-reduction algorithms.

When selecting samples (ratings) for the test set $\mathbf{T}$, each of the definitions described in the previous section (static, future and incremental) can be further refined by removing the short-tail movies (i.e., the most popular) from the evaluation. This approach, labeled as "long-tail", is used to test the quality of the algorithms only on those items that are less popular (i.e., the mid and the long-tail item sets), trying to evaluate the capability of the system in recommending non-banal items, a concept known as *serendipity* [15].

5. RESULTS

In this section we present and discuss the quality of the recommender algorithms presented in Section 2 on two datasets that we will refer to as TV1 and TV2, respectively. Both datasets are composed by implicit, binary ratings representing which items have been watched by each user.

For each dataset, we first report the statistical properties, then we show the quality of the algorithms over time according to the testing methodology explained in Section 4. The reported results are the average recall among the 10 folds.

5.1 First dataset: TV1

In the following we evaluate the recommender algorithms on the dataset collected in about one year of activity of TV1.

Figure 3 reports some statistical properties of the TV1 dataset as a function of time: the number of views (i.e., ratings) and the dataset density (a), the average number of views per item and per user (b), the number of active users and active items (c). We can observe that the number of ratings grow almost linearly with time, while the URM density tends to flatten. A slightly different behavior happens around day 250, where the IPTV provider had a promotional campaign, with a more than linear increase of collected ratings. We will discuss later the effects of such campaign on the algorithm quality.

Figures 6-8(a) shows the recall of the two classes of collaborative algorithms presented in Section 2 as a function of time: kNN item-based (referred to as *cos knn*), and dimensionality-reduction-based (referred to as *svd*). For the kNN algorithm, we have tested different values of the number k of nearest-neighborhoods items. For the SVD algorithm, we have tested different values of the latent-size parameter l. For both the algorithms, we report only the tests with the most significant results. In addition, we have also plotted the recall of a trivial, basic algorithm, referred to as *toprated*, that represents a reference value. Such algorithm suggests, for any user profile, the list of the 5 most-popular items (discarding items already rated/viewed by the user).

Figure 6(a) and 6(b) refer to full static and long-tail static tests, respectively.

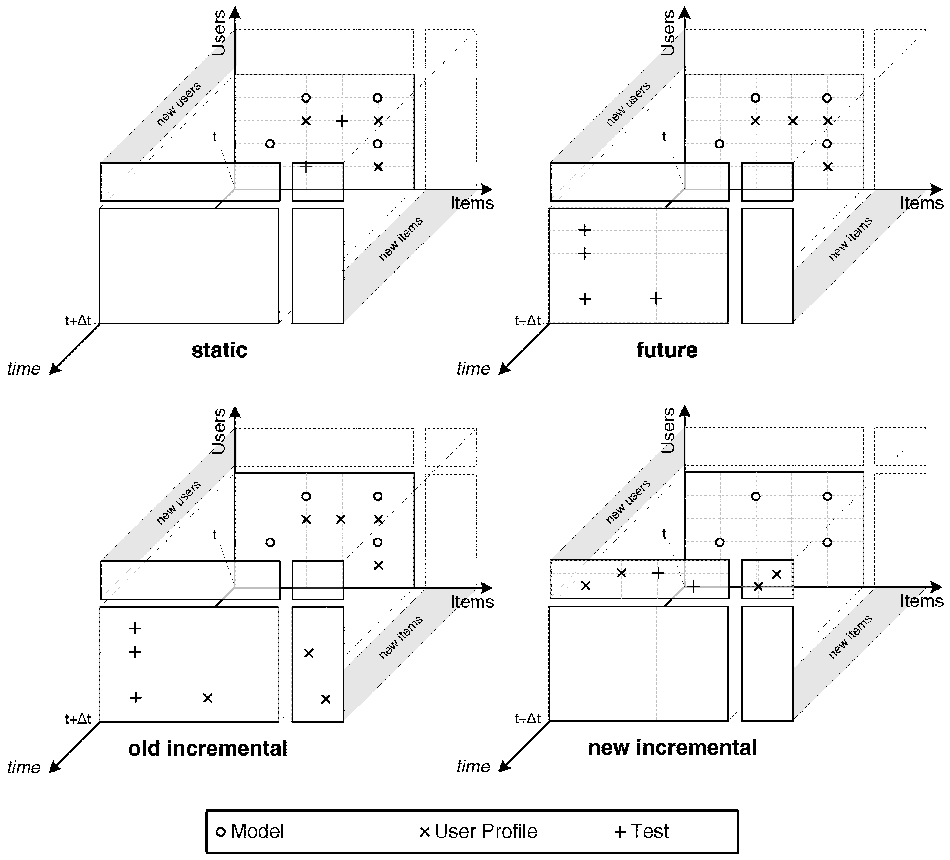


Figure 4: Formalization of the four classes of test: static, future, old incremental and new incremental.

Surprisingly, we first observe that the recall for some of the algorithms decreases with time before reaching a steady-state value. This result apparently is in contrast with other results stating that the learning rate during the cold-start phase increases with the density of the dataset [15]. However, we have to consider that, although the dataset density increases over time, the number of active items increases as well (as opposed to a simulated evolution). Indeed, the larger the number of items, the harder it is for an algorithm to select items that users have effectively watched.

Figure 6 shows also that the kNN algorithm always outperforms the SVD algorithm in the early stage of the system, when there are few ratings and items in the dataset.

Long-tail static test shows that the more we focus on long-tail items, the lower the quality of the kNN algorithm. On the other hand, SVD seems to be more robust in recommending non-popular items. Indeed, after about 4 months there is a swap between kNN and SVD.

We can further note that the optimal parameters of the two algorithms depend on the popularity of the items we are recommending. For instance, if we want to push up quality on long-tail items we have to use a high number of features (e.g., 100) for the SVD algorithm. In fact, the first features of the SVD capture the characteristics of the most popular items (since they are the strongest information in the URM), so we need more features to represent long-tail items, i.e., long tail items are a sort of 'noise' in the URM.

Figures 7(a) and 7(b) shows the capability of a model built at time t of generating correct recommendations; in fact, the test is performed on the new ratings collected in a 30-day time window after t.

The figures show that both the algorithms seem to have a better quality with new (i.e., new) users than with old users. The reason behind this behavior is that new users tend to watch more popular items than existing users, so popularity-biased algorithms recommend such users particularly well. For instance, Figure 9(a) shows the long-tail distribution on TV1 at time $t = 280$, distinguishing between old and new users. Indeed, new users have a much more 'blockbuster' behavior than old users.

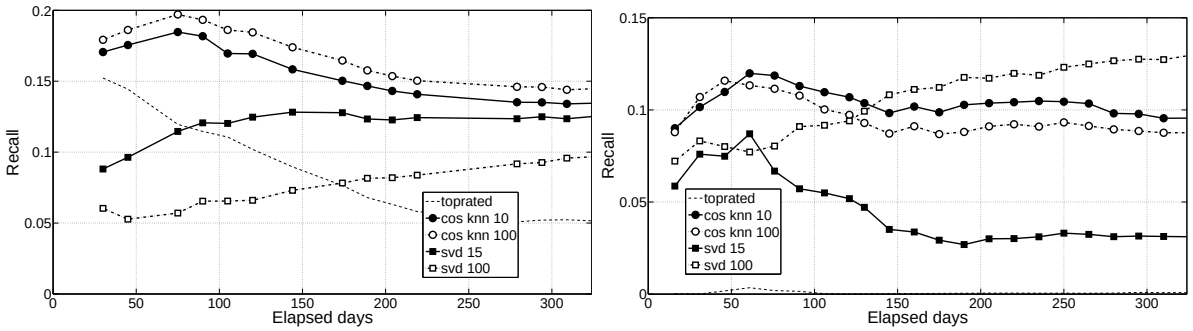
Finally, Figure 8(a) reports an interesting result: up to about 10% of watched items would have been recommended in the previous 24 hours.

Note that in correspondence of the promotional campaign (around day 250), Figures 7(a) and 8(a) show peaks of recall for any algorithm, toprated included. We can explain such behavior by analyzing the dotted line in Figure 9(b) which shows that, with respect to the average trend (solid line), users tend to watch more popular items during promotional campaigns. As a consequence, popularity-biased algorithms (e.g., cos knn and toprated) are privileged.

5.2 Second dataset: TV2

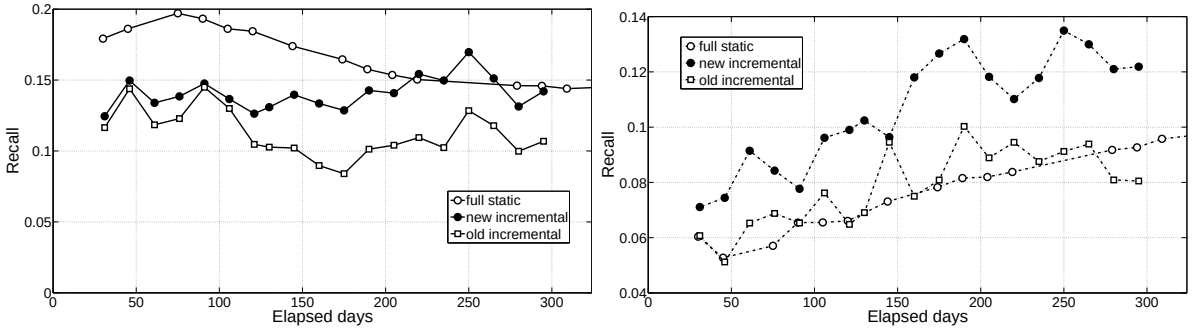
In this section we analyze the quality of the recommender algorithms on the dataset collected in 6 months of activity by the IPTV provider TV2.

Figure 10 reports some statistical properties of the dataset as a function of time. Compared to TV1, the number of active users, active items and ratings is about one order of magnitude larger. However, TV2 dataset is one order of magnitude sparser, and such



(a) full static (b) long-tail static

Figure 6: Tests on TV1 dataset: full static (a) and long–tail static (b).



(a) Item-based (b) SVD-based

Figure 7: Incremental tests versus full static tests on TV1 dataset: item-based (a) and (b) SVD-based.

density is practically constant over time (in 6 months there is a minimal increase from 0.11% to 0.17%).

Recalls reported in Figure 11 confirm that kNN has generally a better quality than SVD, both when evaluated on all items or on a subset of long-tail items.

We can observe that the average quality of the kNN algorithms is larger if compared to the quality obtained on TV1. This is mainly due to the bias of users toward popular items. Accordingly, the recall of kNN algorithms drops when recommending long-tail items. Again, the quality of the SVD algorithms is, on average, less sensible to the item popularity.

6. CONCLUSIONS

The tests conduced on the two datasets show the evolution over time of two classes of collaborative recommender algorithms during the cold-start phase of a RS, considering that new ratings are collected, new users join the system, and new items are added to the catalog.

The first result of the paper outlines that the item-based algorithm performs better with respect to the SVD-based algorithm in the early stage of the cold-start problem. The second result shows that the accuracy of the SVD-based algorithm when using few latent factors decreases with the time-evolution of the dataset. On the contrary, the SVD-based algorithm, when used with a large-enough number of latent features, increases its accuracy with the evolution of the dataset and may outperform the item-based algorithm if the dataset does not present a long-tail behavior.

Further, ongoing tests are meant to evaluate other aspects of the cold-start problem, such as: differentiating the evaluation between the existing users and the new users (with respect to a certain reference time), or comparing the quality of collaborative algorithms with the quality of content-based algorithms.

References

[1] J. F. Abreu, P. Almeida, R. Pinto, and V. Nobre. Implementation of social features over regular iptv stb. In *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 29–32, New York, NY, USA, 2009. ACM.

[2] 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.

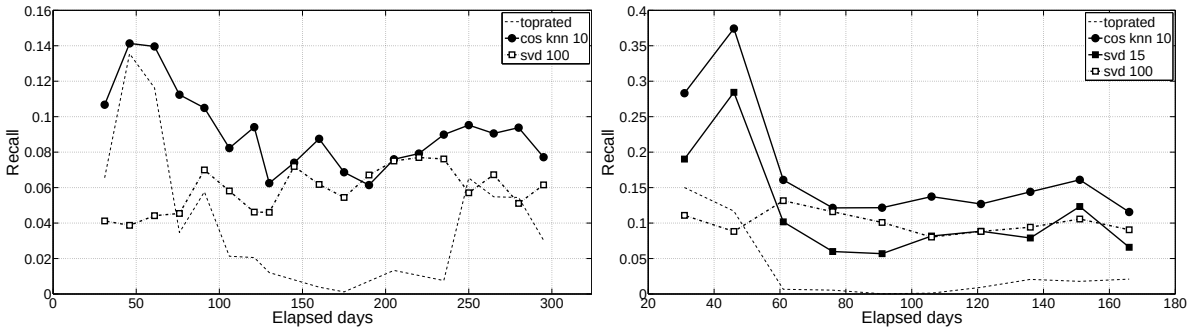
[3] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Inf. Sci.*, 178(1):37–51, 2008.

[4] J. Bennett and S. Lanning. The Netflix Prize. *Proceedings of KDD Cup and Workshop*, pages 3–6, 2007.

[5] E. Campochiaro, R. Casatta, P. Cremonesi, and R. Turrin. Do metrics make recommender algorithms? In *IEEE Advanced Information Networking and Applications (AINA)*, 2009.

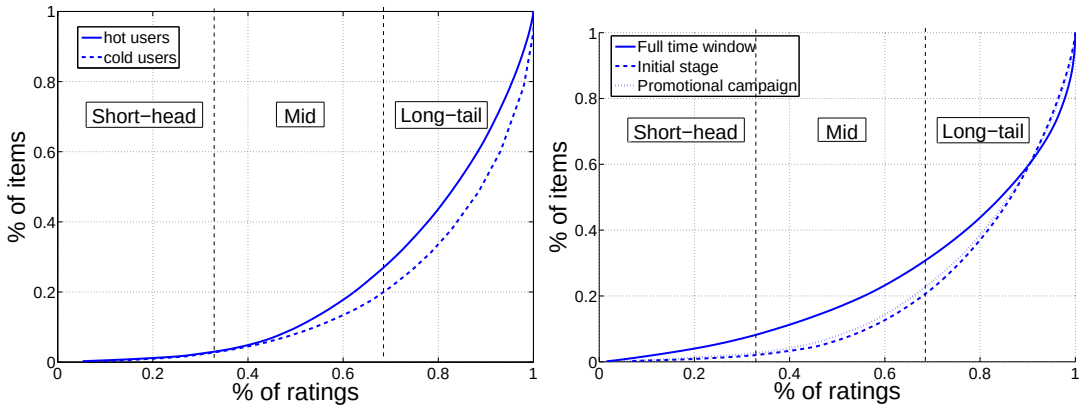
[6] O. Celma and P. Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. Las Vegas, USA, August 2008.

[7] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *RecSys '08: Proceedings of the 2008*



(a) TV1 (b) TV2

Figure 8: Future test: % of recommended items watched in the next 24 hours.



(a) Old vs new users (b) Time evolution

Figure 9: Long-tail distribution of TV1. Figure (a) compares old users (solid line) - i.e., users existing at time $t = 280$ - versus new users (dashed line). Figure (b) shows the long-tail on the complete set of ratings (solid line), in the initial stage (dashed line), and during the promotional campaign (dotted line).

ACM conference on Recommender systems, pages 179–186, New York, NY, USA, 2008. ACM.

[8] P. Cremonesi, E. Lentini, M. Matteucci, and R. Turrin. An evaluation methodology for recommender systems. *4th Int. Conf. on Automated Solutions for Cross Media Content and Multi-channel Distribution*, pages 224–231, Nov 2008.

[9] P. Cremonesi and R. Turrin. Analysis of cold-start recommendations in iptv systems. In *RecSys '09: Proceedings of the 2009 ACM conference on RecommenderSystems*, pages 1–4. ACM, 2009.

[10] T. De Pessemier, T. Deryckere, and L. Martens. Context aware recommendations for user-generated content on a social network site. In *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 133–136, New York, NY, USA, 2009. ACM.

[11] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[12] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. pages 465–480, New York, NY, USA, 1988. ACM Press.

[13] A. Gunawardana and C. Meek. Tied boltzmann machines for cold start recommendations. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 19–26, New York, NY, USA, 2008. ACM.

[14] J. Herlocker, J. Konstan, and J. Riedl. An algorithmic framework for performing collaborative filtering. *22nd ACM SIGIR Conf. on R&D in Information Retrieval*, pages 230–237, 1999.

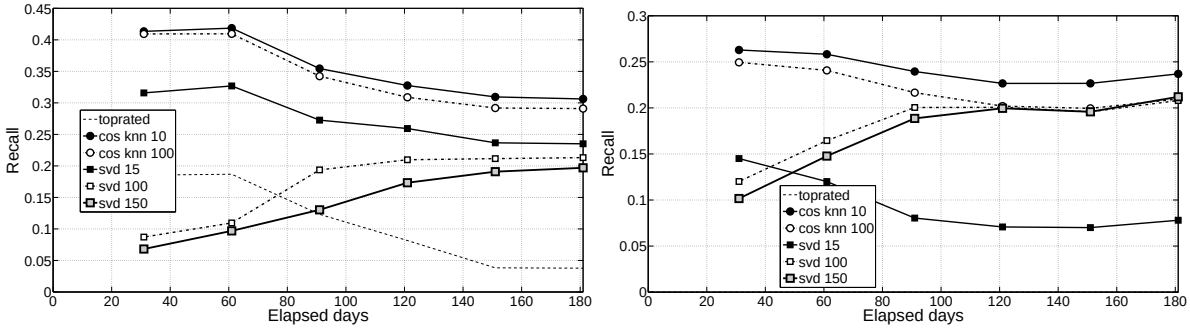
[15] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[16] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. pages 241–250, New York, NY, USA, 2000. ACM.

[17] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):116–142, 2004.

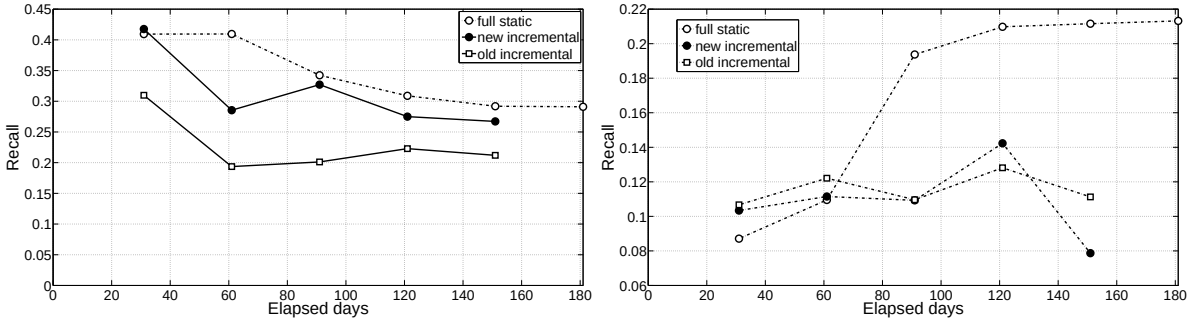
[18] P. Husbands, H. Simon, and C. Ding. On the use of singular value decomposition for text retrieval. Oct. 2000.

[19] P. Lops, M. Degemmis, and G. Semeraro. Improving social filtering techniques through wordnet-based user profiles. In



(a) full static (b) long-tail static

Figure 11: Tests on TV2 dataset: full static (a) and long–tail static (b).



(a) old incremental (b) new incremental

Figure 12: Tests on TV2 dataset: old (a) and new incremental (b).

UM '07: Proceedings of the 11th international conference on User Modeling, pages 268–277, Berlin, Heidelberg, 2007. Springer-Verlag.

[20] A.-T. Nguyen, N. Denos, and C. Berrut. Improving new user recommendations with rule-based induction on cold user data. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 121–128, New York, NY, USA, 2007. ACM.

[21] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste. Naïve filterbots for robust cold-start recommendations. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 699–705, New York, NY, USA, 2006. ACM.

[22] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. *10th Int. Conf. on World Wide Web*, pages 285–295, 2001.

[24] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. *5th Int. Conf. on Computer and Information Technology (ICCIT 2002)*, pages 399–404, 2002.

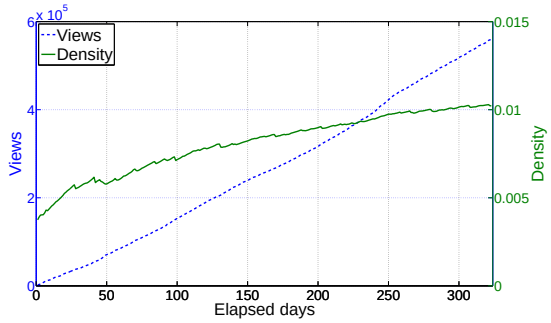
[25] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Generate models for cold-start recommendations. In *ACMSIGIR Workshop on RecommenderSystems.*, 2001.

[26] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 253–260, 2002.

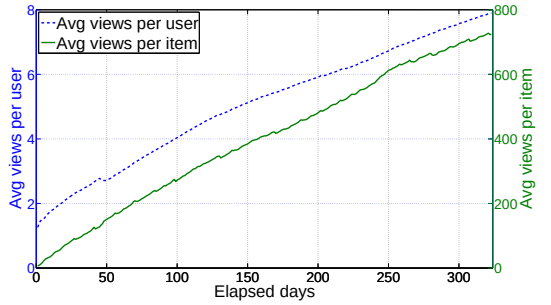
[27] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, 2009.

[28] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. *Proc. of the 6th Hellenic European Conf. on Computer Mathematics and its Applications*, 2003.

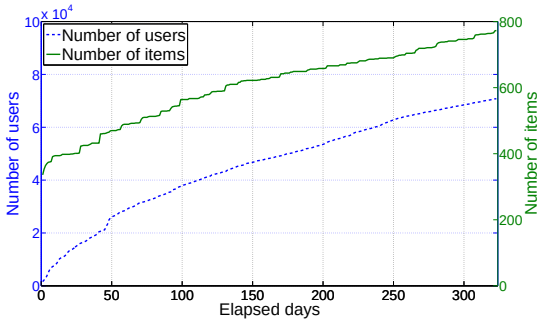
[29] J. Zaletelj, M. Savić, and M. Meža. Real-time viewer feedback in the itv production. In *EuroITV '09: Proceedings of the seventh european conference on European interactive television conference*, pages 149–152, New York, NY, USA, 2009. ACM.



(a) URM: collected ratings and density



(b) Avg views per item and per user



(c) Number of users and items

Figure 3: TV1 dataset: number of ratings and density of URM (a), average views per item and per user (b), number of users and items (c).

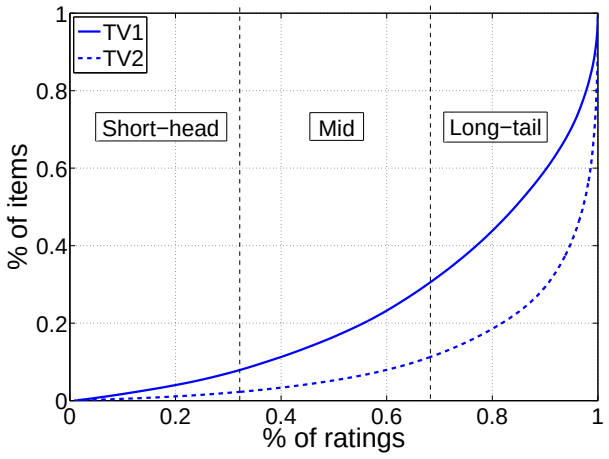
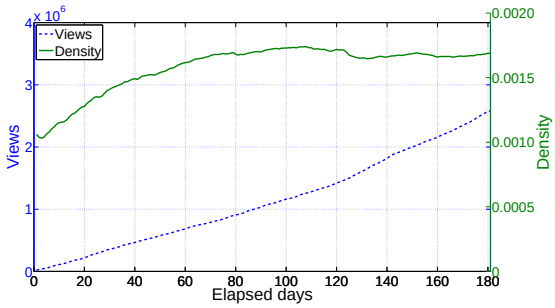
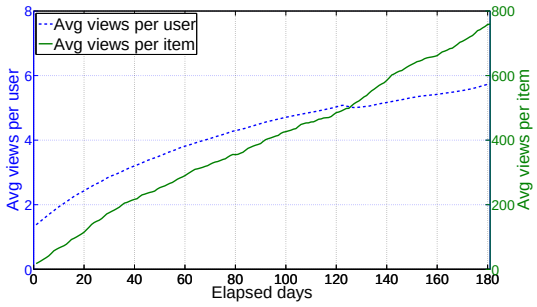


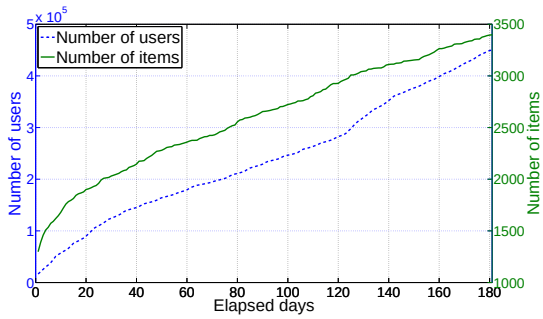
Figure 5: Long-tail distribution of TV1 (solid line) and of TV2 (dashed line).



(a) URM: collected ratings and density



(b) Avg views per item and per user



(c) Number of users and items

Figure 10: TV2 dataset: number of ratings and density of URM (a), average views per item and per user (b), number of users and items (c).