

# Analysis of Cold-Start Recommendations in IPTV Systems

Paolo Cremonesi and Roberto Turrin

WHITEPAPER *March 2010*

# ANALYSIS OF COLD-START RECOMMENDATIONS IN IPTV SYSTEMS

**Abstract** *In this paper we evaluate the performance of different collaborative algorithms in cold-start situations, where the initial lack of ratings may affect the quality of the algorithms. Instead of trying to design an algorithm that provides, on average, a reasonable good accuracy in both the early and steady stages of a recommender system, in this paper we suggest adopting different algorithms in the different stages of the recommender system life-cycle.*

*The evaluation has been performed on the datasets collected by two digital-television providers in Europe. Both the 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.*

## General Terms

Cold start, collaborative algorithms, implicit dataset.

## 1 INTRODUCTION

Cold-start problems (also referred to as start-up problems) refer to situations where there are only a few ratings on which to base recommendations.

According to [21] the cold-start problem can occur under three scenarios:

**new user:** when a new user first registers with the recommender system, there are very few ratings available to describe that user profile.

**new item:** when a new item is added to the catalog, it has no ratings.

**new system:** when bootstrapping a new recommender system, the average number of ratings per user and item is low and this can significantly degrade the performance of collaborative algorithms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

This paper is a draft preprint version of Paolo Cremonesi and Roberto Turrin, "Analysis of cold-start recommendations in IPTV systems", RecSys, page 233-236. ACM, (2009)

Please, refer to the published version for citations. The copyright is owned by the publisher.

As recommender systems are spreading into data-sparse domains, algorithm learning rate during the cold-start phase has become a significant evaluation factor. However, the issue of evaluating the performance of collaborative recommender systems during their initial stage of use has not been extensively covered in the literature. The lack of evaluations is largely due to the characteristics of the public available 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 sub-sample the dataset [13]. Previous works mainly attempted to combine collaborative filtering with content-based recommendation approaches to address the new-item problem [10, 17, 22, 23]. Other works focused on the design of new collaborative 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 [1, 13]. Few works addressed the user problem [16].

In this paper we benchmark two collaborative algorithms against a non-personalized recommendation method on the cold-start new system problem, where the initial lack of ratings may affect the quality of the algorithms. The evaluation has been performed on the datasets collected by two IPTelevision providers in Europe having, respectively, 200000 and 600000 television subscribers. Both providers have started projects to implement a recommender system within their television infrastructure. The two datasets have been implicitly collected by analyzing the pay-per-view movies purchased by the users over a period of several months. The time evolution of the datasets has allowed the coldstart evaluation of two different recommender algorithms: an item-based and a SVD-based.

This paper differs from previous works because, instead of designing a new algorithm suited to the cold-start problem, we try to provide guidelines to help in the selection and tuning of different state-of-the-art collaborative algorithms in the different stages of the recommender systems life-cycle.

Moreover, we combine the experience achieved on two real-life cold-start applications to obtain a deeper understanding of the performance characteristics of recommender systems.

The main results suggest adopting item-based algorithms in the early stage of the cold-start period and, eventually to switch to SVD-based algorithms.

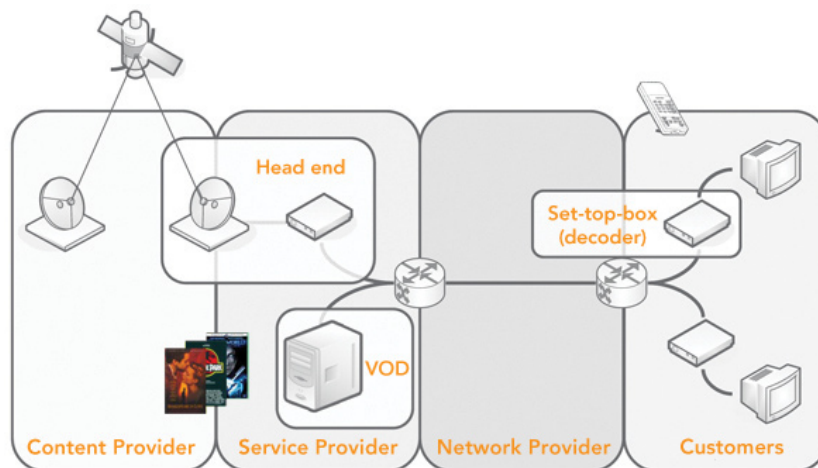


Figure 1: IPTV architecture. Head-end devices receive live tv-channels and broadcast them on the network. VOD servers store on-demand movies.

## 2. IPTV SYSTEMS

IPTV is a video service supplied by a telecommunications service provider that controls content distribution over a broadband network for reliable delivery to the consumer's set-top-box (STB). The STB is an electronic appliance which communicates with the IPTV service provider, shows the multimedia content and manages all the user interactions. IPTV system architectures, as the one shown in Figure 1, support applications such as live broadcast TV and video on demand (VOD).

Within IPTV systems, browsing and searching for interesting multimedia content is challenging [9], especially due to the limited capabilities of the input device and to the slow channel switching time. The remote control does not allow to point an arbitrary object on the screen or to easily enter text and, compared to traditional television, IPTV is not particularly responsive to user actions, mainly due to technological limitations [15].

The use of a recommender system into an IPTV infrastructure improves the user experience and alleviates the questions concerning the search for interesting programs and movies.

However, such integration faces with several problems, mainly the real-time requirements and the poor data quality and availability. Recommender systems for IPTV must generate recommendations at real-time with very strict time constraints, in order to not slow down the user navigation, already conditioned by the channel switching time. IPTV systems usually dispose of quite poor data, in terms of quality of the ratings. Differently from other domains (e.g., webbased domains), that operate over explicit rating schemes, wherein users are required to express a preference for items that they have reviewed or purchased, the ratings in IPTV systems are implicitly collected by monitoring which movies have been watched by the users. However, this collection mechanism does not necessarily match the user taste.

## 3. TESTED ALGORITHMS

Collaborative algorithms provide, on average, a better performance with respect to content-based algorithms [11]. However, collaborative algorithms are affected by the *cold-start* problem. As a consequence, at the early stages of a recommender system life-cycle, the algorithms cannot provide accurate recommendations.

In the following we describe two state-of-the-art collaborative algorithms tested for the cold-start problem: an item-based and a dimensionality-reduction algorithm. Both algorithms use a dataset formed by 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 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.

### 3.1 Item-based algorithm

Item-based collaborative algorithms capture the fundamental relationships among items [19]. 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 recommender system 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 [7].

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

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

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 be further enhanced by means of a  $k$ NN ( $k$ -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  $k$ NN 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  $r_{pj} = 1$ )

$$\hat{r}_{pi} = \sum_{j \in r_{pi}=1} d_{ij} \cdot r_{pj} \quad (2)$$

### 3.2 Dimensionality-reduction algorithm

Collaborative algorithms based on dimensionality-reduction techniques 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:

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

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 [18]. We have based our analysis on the singular value decomposition (SVD) (e.g., [6, 8, 14, 20]). By means of SVD, the URM can be factorized as

$$\hat{\mathbf{R}} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T \quad (4)$$

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

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

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

$$\hat{r}_{pi} = \mathbf{r}_p \cdot \mathbf{V} \cdot \mathbf{v}_i^T \quad (6)$$

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 represents a great advantage if compared, for instance, with other dimensionality-reduction techniques where the features of users need to be pre-computed (e.g., [18]).

## 4. TESTING METHODOLOGY

In the next sections we are going to investigate:

- how the two algorithms perform when they are applied on sparse datasets;
- how their learning rate evolves with time (i.e., with increasing user profile length, increasing views per items, and increasing number of users and of items in the catalog);
- if and how the algorithm parameters (e.g., latent size  $l$ , neighborhood size  $k$ ) should be tuned when the dataset evolves with time.

The dataset evolution is driven by three inputs:

1. existing users watch (i.e., they implicitly rate) new movies in the catalog;
2. new users join the system (i.e., new or exiting users watch their first movie since the recommender system is running);
3. 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).

The first input source has the effect of increasing the dataset density, the average user profile length, and the average number of views per item. The second input factor 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 factor 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. Consequently, both the dataset parameters and the quality metrics should be expressed as a function of time  $t$ . However, to simplify the notation, we will omit the time parameter.

Typical approaches for recommender system evaluation are based either on error metrics (e.g., RMSE and MAE) or classification accuracy metrics (e.g., recall, precision, and fall-out) [12, 6]. Since we only dispose of implicit, binary ratings (i.e., views of users, that are assumed to express positive preferences), we are constrained in evaluating the quality of the system by means of an accuracy metric: the *recall*. Recall tries to estimate how many movies relevant to the users are recommended by the algorithm.

In order to define a methodology for evaluating the recall of an algorithm at time  $t$ , we refer to the following three sets:

- The train set  $\mathbf{M}$  contains the users (and their implicit ratings) used to train the algorithm.
- The user test set  $\mathbf{P}$  contain the user profiles adopted to test the algorithm.
- The item test set  $\mathbf{T}$  contains the items used to compute the recall. The set  $\mathbf{T}$  will be better defined in the following.

Note that items not active at time  $t$  (i.e., with no ratings in  $\mathbf{M}$ ) and users not active at time  $t$  (i.e., with no ratings in  $\mathbf{P}$ ) can not be tested. Such items and users can be tested at later time if they become active (e.g., if they have at least one rating).

In order to stress also the new-user problem, we have adopted a  $k$ -fold methodology [6, 3]:

1. The URM is divided into  $k$  folds by randomly splitting the rows of the matrix (i.e., the users) into  $k$  sets. In our tests  $k = 10$ .
2. Part of the users ( $k - 1$  folds) are used to build the model. These users form the train set  $\mathbf{M}$ .
3. The remaining users (one fold) are used to evaluate the quality of the recommender algorithm on the

dataset, as detailed in the following. This fold represents the user test set  $\mathbf{P}$ .

The process is repeated  $k$  times, selecting a different  $\mathbf{P}$  fold at each iteration. Within step 3, each user is tested by using a leave-one-out approach:

- i. For each user in  $\mathbf{P}$ , we select all the items in  $\mathbf{T}$  which have been rated by the user. These items are the *test items*.
- ii. For each test item, the rating is removed from the current user profile, and recommendations are generated on the basis of such a modified user.
- iii. If the test item is recommended to the user within the first  $N$  positions we have a *hit*. According to typical IPTV user interfaces [2], we assume  $N = 5$ .

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

The item test set  $\mathbf{T}$  has been chosen on the basis of two different approaches [5]: (a) we consider all the active items, or (b) we consider the subset of items that represent the long-tail.

The first approach evaluates the overall quality of the algorithms on the whole set of items. The second approach tests the quality of the algorithms only on those items that are less popular (the long-tail), trying to evaluate the capability of the system in recommending non-banal items, a concept known as *serendipity* [12]. 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, since they represent a costly investment (copyrights, hardware infrastructure, storage, etc.).

#### 4.1 Cold-start vs random sub-sampling

Many works in the literature (see, e.g., [13]) 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 nonrealistic 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, we have sub-sampled the dataset of the first IPTV operator, 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 actual dataset evolution shown in Figure 3, the most noticeable differences are in the time-evolution of the density and of the 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).

## 5. RESULTS

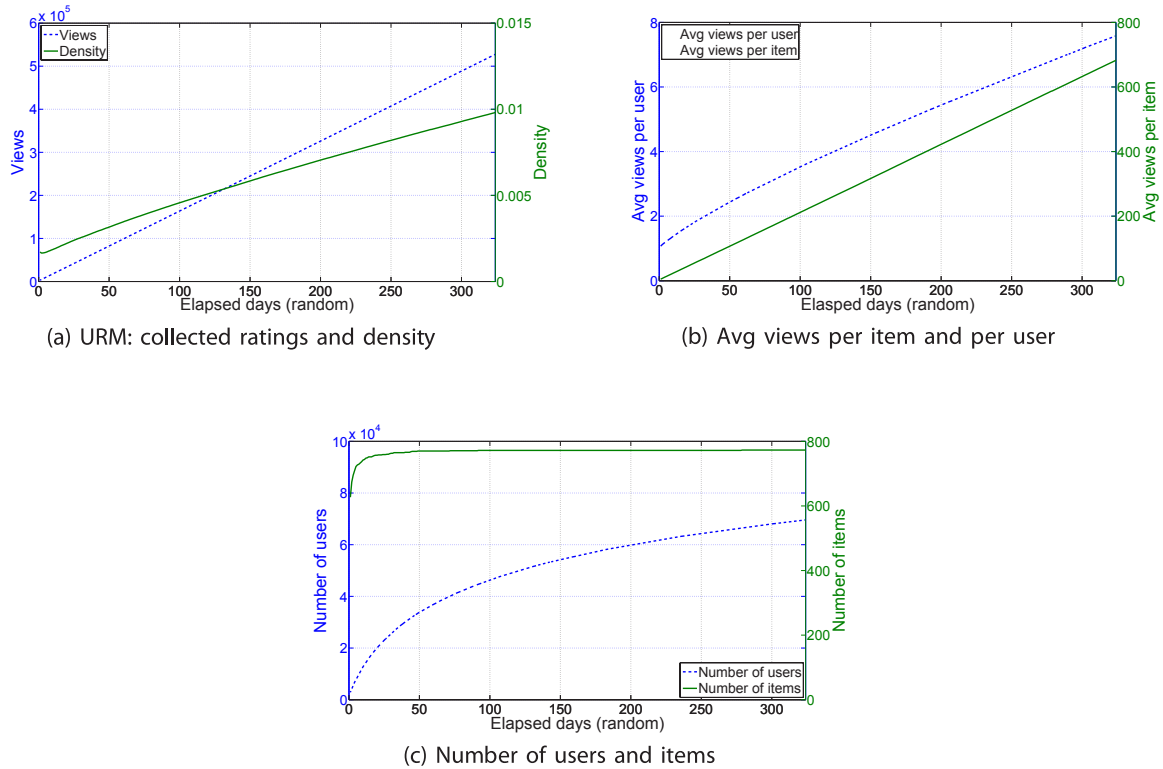


Figure 2: TV1 dataset with random timestamps: number of ratings and density of URM (a), average views per item and per user (b), number of users and items (c).

In this section we present and discuss the quality of the recommender algorithms presented in Section 3 on two datasets provided by two different European IPTV providers, 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.

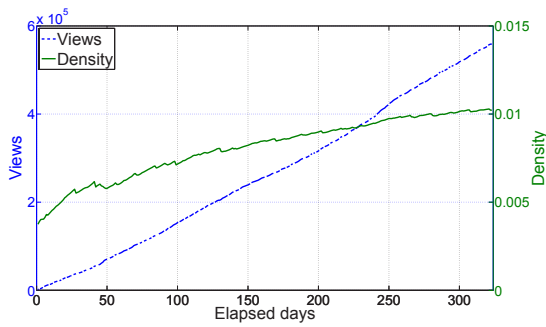
For each dataset, we first report the statistical properties, then we show the quality of algorithms over time according to the testing methodology explained in Section 4. The reported results are the average recall among the 10 folds.

### 5.1 First dataset: TV1

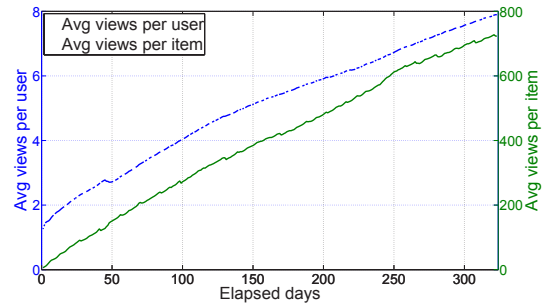
In the following we evaluate the recommender algorithms on the dataset collected in about one year of activity of TV1.

Figure 3 reports some statistical properties of such 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.

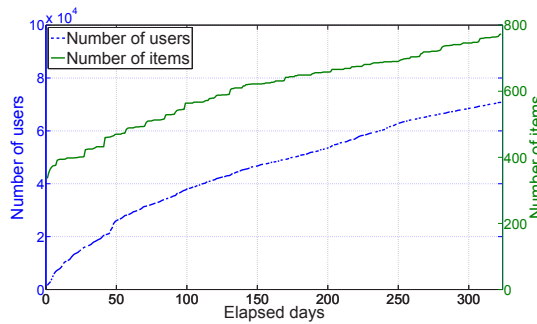
Figure 4 represents the percentage of ratings accounted by the most popular items. Plotted data refer to the most recent time period.



(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).

The analysis on the long-tail distribution of TV1 (solid line) shows that about 80% of ratings is concentrated with 40% of the most popular items. Compared to the 80/20 rule-of-thumb for long-tail distributions, the unbalance of ratings toward popular items is not particularly strong.

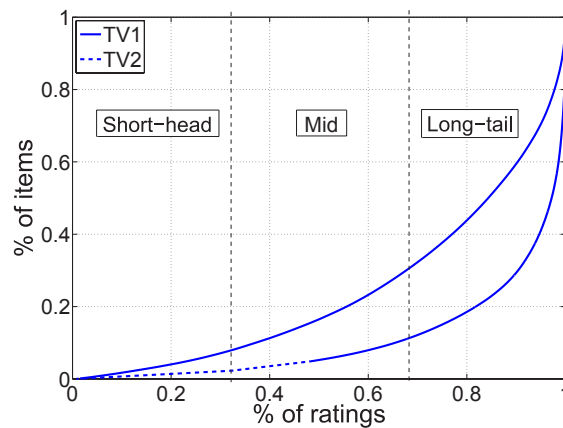


Figure 4: Long-tail distribution of TV1 (solid line) and TV2 (dashed line).

Figure 5 shows the recall of the two classes of collaborative algorithms presented in Section 3 as a function of time: 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  $l$ . 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).

The three graphs in Figure 5 refer to three different item test sets  $T$ , respectively:

- The whole set of items.
- The set of items, having removed the 10 most-popular items, i.e., we discard items that users can typically access from other menus of the IPTV interface.
- The set of items, having removed the most-popular items that account for the 33% of ratings. Such removed items represent the *short-head* of the rating distribution [4]. The 33% of ratings is accounted by about the 5% of items: since the number of active items varies over time, we discard from 20 to 40 popular items.

Surprisingly, we first observe that the recall for some of the algorithms decreases with time before reaching a steadystate 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 [12]. However, we have to consider that, although the dataset density increases over time, the number of active items in a real cold-start problem (as opposed to a simulated cold-start) increases as well. Indeed, the larger the number of items, the harder it is for an algorithm to select items that users have effectively watched.

Figure 5 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.

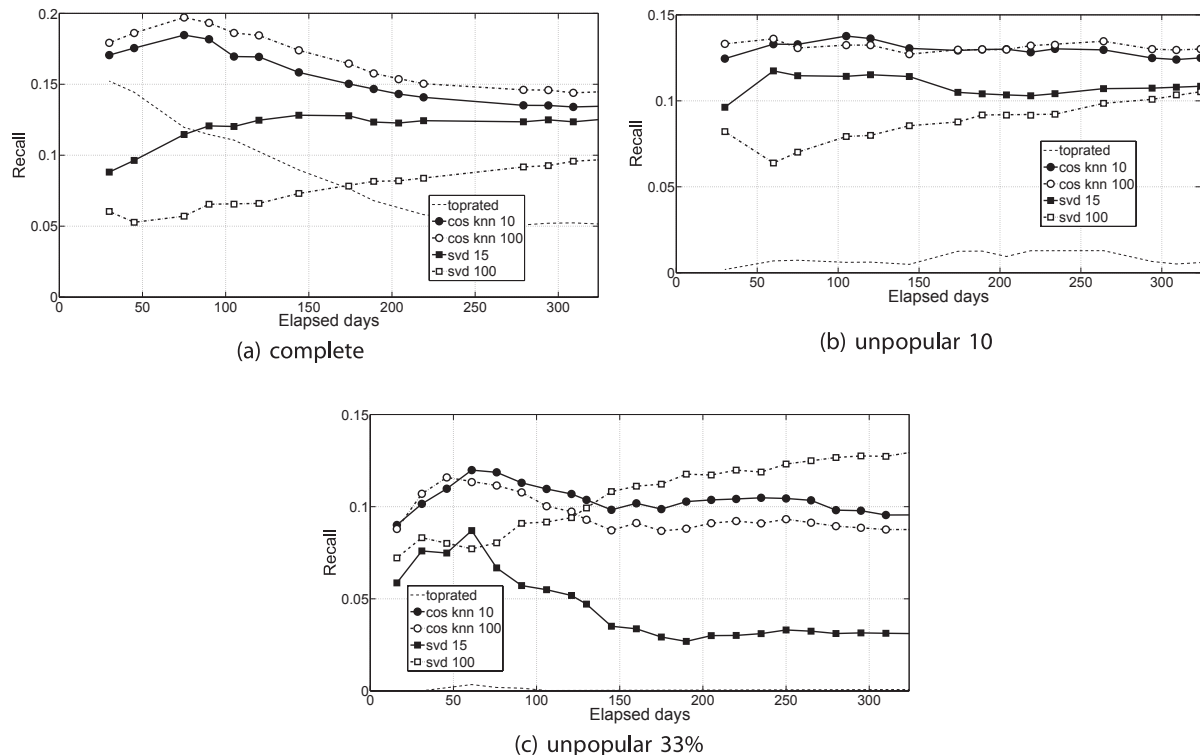


Figure 5: TV1 dataset. Evolution of algorithms over time: on the complete set of items (a) and on long-tail items (b and c).

Test set (b) proves that the *toprated*, which is a nonpersonalized algorithm, is unable to recommend non-popular items, as expected. Moreover, we observe that also the kNN algorithms slightly decrease their accuracy because of the difficulties in recommending non-popular items.

Finally, test set (c) shows that the more we focus on longtail 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 the kNN and the 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, so we need more features to represent long-tail items.

## 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 6 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 higher. However, TV2 dataset is one order of magnitude sparser, and such density is practically constant over time (in 6 months there is a minimal increase from 0.11% to 0.17%).

Observing the dashed line in Figure 4, the long-tail distribution of TV2 respects exactly the 80/20 rule, since about 20% of active items account for about 80% of ratings. As a consequence, since users of this IPTV provider tend to prefer popular items, item-based algorithms are expected to outperform dimensionality-reduction algorithms, as confirmed in the following analysis.

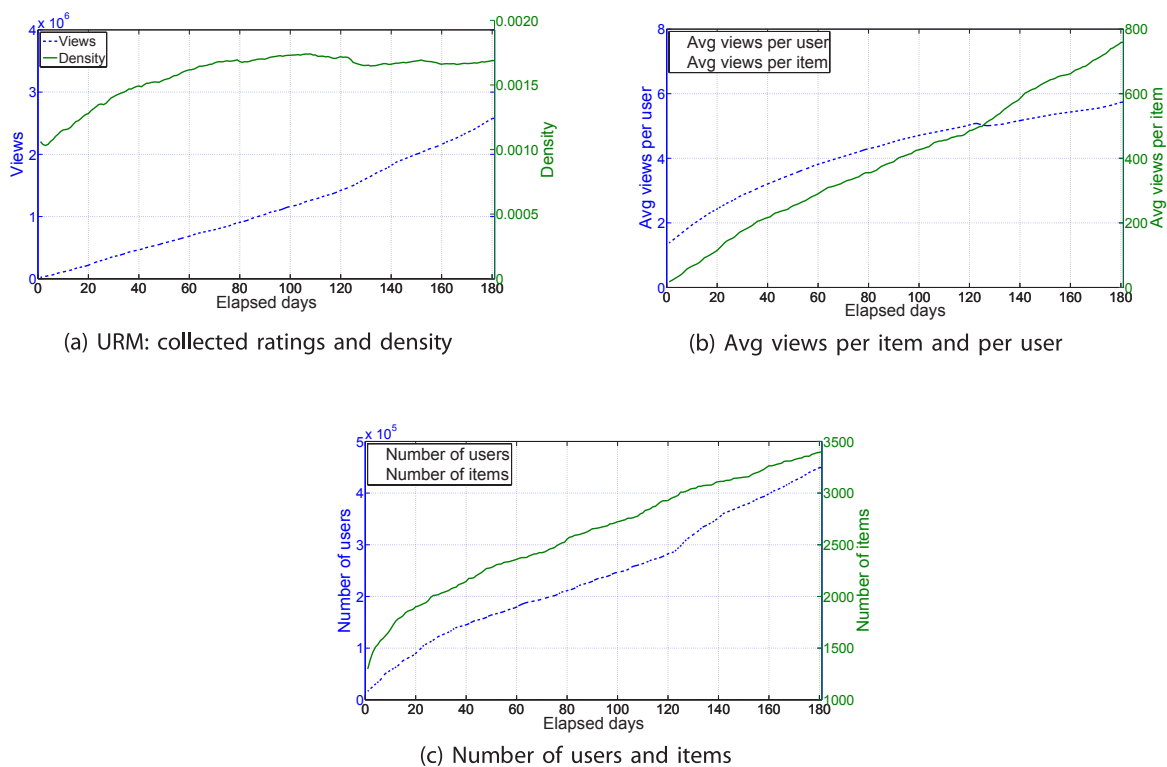


Figure 6: TV2 dataset: number of ratings and density of URM (a), average views per item and per user (b), number of users and items (c).

Recalls reported in Figure 7 confirm that kNN has generally a better quality than SVD, both when evaluated on all items or on a subset of long-tail items. As for Figure 7(c), the 33% of ratings is accounted by 2.3% of items, i.e., we discard from 40 to 80 popular items.

We can observe that the average quality of the kNN algorithms is higher 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.

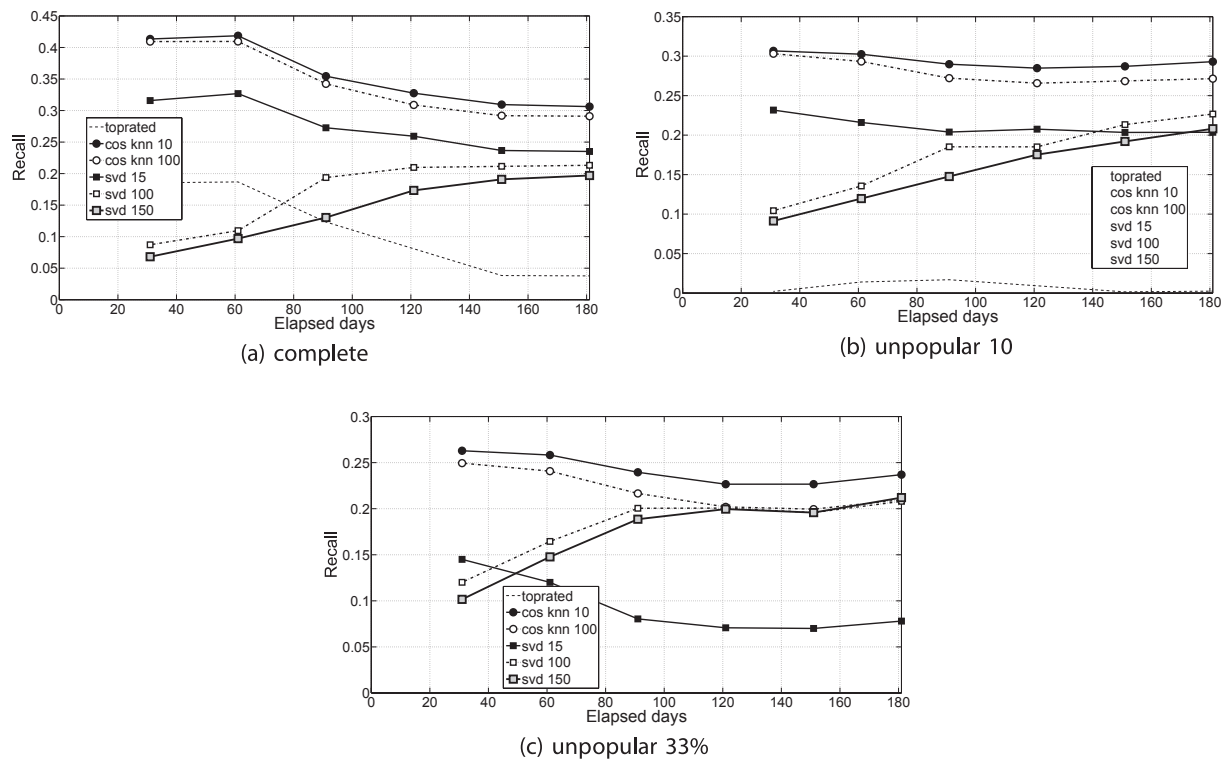


Figure 7: TV2 dataset. Evolution of algorithms over time: on the complete set of items (a) and on long-tail items (b and c).

## 6. CONCLUSIONS

The tests conducted on the two datasets show the evolution over time of two classes of collaborative recommender algorithms during the cold-start phase of a recommender system, 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.

## 7. REFERENCES

- [1] 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.
- [2] R. Bambini, P. Cremonesi, and R. Turrin. *Recommender Systems Handbook*, chapter A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. under revision, Springer, 2009.
- [3] E. Campochiaro, R. Casatta, P. Cremonesi, and R. Turrin. Do metrics make recommender algorithms? In *IEEE Advanced Information Networking and Applications (AINA)*, 2009.
- [4] O. Celma and P. Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. Las Vegas, USA, August 2008.
- [5] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 179–186, New York, NY, USA, 2008. ACM.
- [6] 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.
- [7] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.
- [8] 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.
- [9] U. D. Geneve and S. Marchand-maillet. Vision content-based video retrieval: An overview.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] P. Husbands, H. Simon, and C. Ding. On the use of singular value decomposition for text retrieval. Oct. 2000.
- [15] Y. Lee, J. Lee, I. Kim, and H. Shin. Reducing iptv channel switching time using h.264 scalable video coding. *Consumer Electronics, IEEE Transactions on*, 54(2):912–919, May 2008.

- [16] 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.
- [17] 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.
- [18] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.
- [19] 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.
- [20] 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.
- [21] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. pages 291–324. 2007.
- [22] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Generate models for cold-start recommendations. In *ACMSIGIR Workshop on RecommenderSystems.*, 2001.
- [23] 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.