A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment

Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin

Abstract In this chapter we describe the integration of a recommender system into the production environment of Fastweb, one of the largest European IP Television (IPTV) providers. The recommender system implements both collaborative and content-based algorithms, suitable tailored to the specific requirements of an IPTV architecture, such as the limited screen definition, the reduced navigation capabilities, and the strict time constraints. The algorithms are extensively analyzed by means of off-line and on-line tests, showing an effective success of the recommender systems: up to 30% of the recommendations are followed by a purchase, with an estimated lift factor (increase in sales) of 15%.

1 Introduction

IP Television (IPTV) broadcasts multimedia content (e.g., movies, news programs, documentaries) in digital format via broadband Internet networks [18, 15]. IPTV services include scheduled television programs and video on demand (VoD) contents [23]. In the rest of the chapter we will generically refer to both scheduled television programs and video-on-demand contents as *items*.

In this chapter we present the integration of the Neptuny's ContentWise recommender system in Fastweb, the first company in the world to have launched fully IP-based broadband TV services, in 2001. Fastweb serves hundreds of thousands of IPTV customers, with a catalog of thousands of multimedia contents. Since 2007

Riccardo Bambini
Fastweb, via Francesco Caracciolo 51, Milano, Italy

Paolo Cremonesi
Politecnico di Milano, p.zza Leonardo da Vinci 32, Milano, Italy

Roberto Turrin
Neptuny, via Durando 10, Milano, Italy

Fastweb is part of the Swisscom group. ContentWise recommender algorithms have been developed with the cooperation of the Computer Science Department at the Politecnico di Milano.

Differently from conventional television, IPTV allows an interactive navigation of the available content [12] and, in particular, IPTV allows to collect implicit usage profiles and explicit user preferences for providing a personalized user navigation. The user interacts with the IPTV system by means of a special electronic appliance, referred to as *set-top-box* (STB). There are substantial peculiarities of the STB that limit the user interaction: (i) users control the STB by means of a remote control, rather limited in the set of actions it allows to perform, (ii) the user interface is shown on a TV screen typically designed to be looked at from a distance, and (iii) the system deals with multimedia content, whose navigation is not particularly fast with a STB, mainly because of the channel switching time.

Differently from traditional e-commerce domains (e.g., Amazon, Netflix, iTunes, IMDB, Last.fm), IPTV recommender systems need to satisfy particular requirements:

- the list of proposed items has to be small because of the limited screen definition and the reduced navigation capabilities;
- the generation of the recommended items must respect very strict time constraints (few milliseconds) because TV's customers are used to a very responsive system;
- the system needs to scale up in a successful manner with both the number of customers and items in the catalog;
- part of the catalog is highly dynamic because of the presence of live broadcast channels.

The recommender system deployed in Fastweb generates recommendations by means of two collaborative algorithms (based on item-item similarities and dimensionality reduction techniques) and one content-based algorithm (based on latent semantic analysis). The recommender system selects the proper algorithm depending on the context. For instance, if the user is reading a movie synopsis, looking for movies with his preferred actors, the algorithm used is the content-based. In order to respect the strict real-time requirements, the recommender system and the underlying algorithms follow a model-based approach and have been logically divided into two asynchronous stages, the batch stage and the real-time stage.

The input data of the whole architecture is composed by: (i) the item-content matrix and (ii) the user-rating matrix. The item-content matrix (ICM) describes the main attributes (metadata) of each item, such as the title of a movie, the set of actors and its genre(s). The user-rating matrix (URM) collects the ratings (i.e., preferences) of users about items. Ratings are mainly implicit, e.g., we know if a user watched a program, without knowing explicitly the user's opinion about that program.

Before deploying the recommender system in production, extensive accuracy analysis has been performed by means of k-fold cross validation. The results suggests a 2.5% recall for the content-based algorithm, while the collaborative algorithms are able to reach a recall of more than 20%.

The recommender system has been released to production environment in October 2008 and is now available for one of the Fastweb VOD catalogs. The system is actually providing, on average, 30’000 recommendations per day, with peaks of almost 120 recommendations per minute during peak hours. On-line analysis shows that almost 20% of the reccomendations are followed by a purchase from the users. The estimated lift factor (i.e., increase in VOD sales) is 15%.

The rest of the chapter is organized as follows. Section 2 shows the typical architecture of an IPTV provider. Section 3 presents the architecture of the recommender system. Section 4 describes the implemented recommender algorithms. Section 5 explains the recommender services implemented into the Fastweb IPTV architecture. Section 6 evaluates the quality of recommendations. Finally, Section 7 draws the conclusions.

2 IPTV Architecture

IPTV, also called Internet Protocol Television, is a video service that deliver high quality traditional TV channels and on-demand video and audio contents over a private IP-based broadband network. From the end users perspective, IPTV looks and operates just like a standard TV service. The providers involved in deploying IPTV services range from cable and satellite TV carriers to large telephone companies and private network operators. IPTV has a number of unique features [12]:

Support for interactive TV: differently from conventional TV, where the communication is unidirectional, the two-way capabilities of IPTV systems allow the user to interact with the system.

Time shifting: IPTV permits the temporal navigation of programs (e.g., fast forward, pause and rewind) thanks to the Personal Video Recording (PVR), a mechanism for recording and storing IPTV content for later viewing.

Personalization: IPTV allows end users to personalize their TV viewing habits by letting them decide what they want to watch and when they want to watch it.

Figure 1 shows a generic IPTV system architecture that supports live broadcast TV channels (also called *linear channels*) and video on-demand (VOD). Broadcast TV service consists in the simultaneous reception by the users of traditional TV channels either free-to-air or pay-per-view. Video on-demand service consists in viewing multimedia content made available by the service provider, upon request.

The IPTV data center (also known as the *headend*) receives linear channels from a variety of sources including terrestrial, satellite and cable carriers. Once received, a number of different hardware components, ranging from encoders and video servers, are used to prepare the video content for delivery over an IP based network. On-demand contents are stored in fast storage boxes (e.g., using solid-state disks).

The set-top-box (STB) is an electronic appliance that connects to both the network and the home television: it is responsible for processing the incoming packet stream and displaying the resulting video on the television. The user interacts with

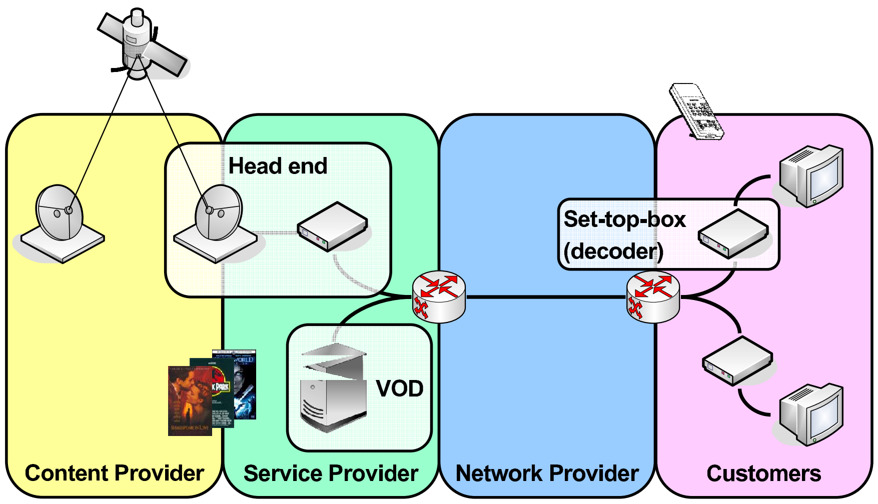


Fig. 1 Architecture of an IPTV system.

the STB by means of a hand-held remote control. The remote control gives the user access to additional features of the STB, such as the Electronic Program Guide (EPG), a listing of available channels and program for an extended time period (typically 36 hours or more).

2.1 *IPTV search problems*

To benefit from the rich set of IPTV channels and contents, users need to be able to rapidly and easily find what they are actually interested in, and do so effortlessly while relaxing on the couch in their living room, a location where they typically do not have easy access to the keyboard, mouse, and close-up screen display typical of desktop web browsing. However, searching for a live channel or a VOD content is a challenging problem for IPTV users [10].

When watching live television, users browse through a set of available channels until they find something interesting. Channel selection (zapping) involves two steps: (a) sampling the content to decide whether to continue or stop watching the channel, and (b) switching across multiple channels for repeated sampling, until a desired channel is found. The problem of quickly finding the right channel becomes harder as the number of channel offerings grows in modern IPTV systems. Moreover, IPTV channel switching time is not particularly responsive, compared to traditional TV, because of technological limitations [16].

When searching for VOD content, IPTV users generally have to either navigate a complex, pre-defined, and often deeply embedded menu structure or type in titles or other key phrases using an on-screen keyboard or triple tap input on the remote control keypad. These interfaces are cumbersome and do not scale well as the range of content available increases. Moreover, the television screens usually offer a limited resolution with respect to traditional personal computer screens, making traditional graphical user interfaces difficult to use.

This differs from traditional web-based domains (e.g., e-commerce web sites), where the content is textual, suited for information categorization and keyword-based seek and retrieval, and the input devices (keyboard and mouse) allow to point an arbitrary object on the screen and to easily enter text.

The integration of a recommender system into the IPTV infrastructure improves the user experience by providing a new and more effective way of browsing for interesting programs and movies. However, such integration faces with some difficulties:

User identification. The STB is used indistinctly by all the components of a family, and the IPTV recommender system can not identify who is actually watching a certain program.

Real-time requirements. The IPTV recommender systems must generate recommendations within very strict real-time constraints (few milliseconds) in order to avoid a slow down of the user navigation, already affected by the long channel switching time.

Quality of content metadata. Differently from web-based domains, content-based IPTV recommender algorithms makes use of low-quality metadata. This aspect is particularly evident with live channels, where new content is added every day at a very high rate, and the only available metadata that can be used to describe programs can is found in the EPG.

3 Recommender system architecture

The architecture of the Fastweb recommender system is shown in Figure 2. The several components are discussed in the following section. Section 3.1 describes the information available to the recommender system. Section 3.2 describes the two-stage architecture of the recommender algorithms, separating between batch and real-time stage. Section 4 details the three algorithms implemented in ContentWise. Finally, Section 5 shows the integration of the recommender system into the existing Fastweb architecture.

3.1 Data collection

The logical component in charge of pre-processing the data and generating the input of the recommender algorithm is referred to as *data collector*. The data collector gathers data from different sources, such as the EPG for information about the live

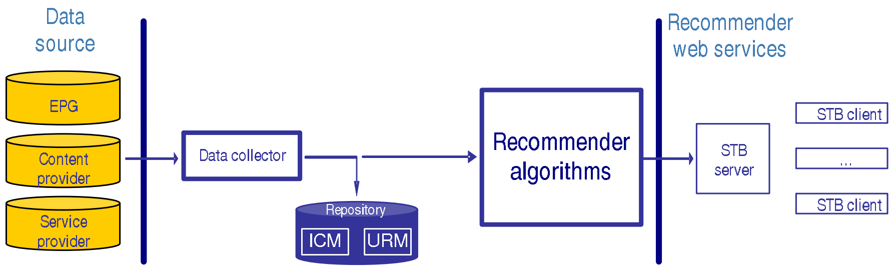


Fig. 2 Architecture of the recommender system ContentWise.

programs, the content provider for information about the VOD catalog and the service provider for information about the users.

The Fastweb recommender system does not rely on personal information from the users (e.g., age, gender, occupation). Recommendations are based on the past users' behavior (what they watched) and on any explicit preference they have expressed (e.g., preferred genres). If the users did not specify any explicit preferences, the system is able to infer them by analyzing the users' past activities.

An important question has been raised in Section 2: users interact with the IPTV system by means of the STB, but typically we can not identify who is actually in front of the TV. Consequently, the STB collects the behavior and the preferences of a set of users (e.g., the component of a family). This represents a considerable problem since we are limited to generate per-STB recommendations. In order to simplify the notation, in the rest of the paper we will refer to user and STB to identify the same entity. The user-disambiguation problem has been partially solved by separating the collected information accordingly to the time slot they refer to. For instance, we can roughly assume the following pattern: housewives use to watch TV during the morning, children during the afternoon, the whole family at evening, while only adults watch TV during the night. By means of this simple time slot distinction we are able to distinguish among different potential users of the same STB.

Formally, the available information has been structured into two main matrices, practically stored into a relational database: the item-content matrix (ICM) and the user-rating matrix (URM).

The former describes the principal characteristics (metadata) of each item. In the following we will refer to the item-content matrix as $\mathbf{W}$, whose elements w_{ci} represent the relevance of characteristic (metadata) c for item i. The ICM is generated from the analysis of the set of information given by the content provider (i.e., the EPG). Such information concerns, for instance, the title of a movie, the actors, the director(s), the genre(s) and the plot. Note that in a real environment we can face with inaccurate information because of the rate new content is added every day. The information provided by the ICM is used to generate a content-based recommendation, after being filtered by means of techniques for PoS (Part-of-Speech) tagging, stop words removal, and latent semantic analysis. Moreover, the ICM can be used to perform some kind of processing on the items (e.g., parental control).

The URM represents the ratings (i.e., preferences) of users about items. In the following we will refer to such matrix as **R**, whose elements r_{pi} represent the rating of user p about item i. Such preferences constitute the basic information for any collaborative algorithm. The user rating can be either explicit or implicit, according to the fact that the ratings are explicitly expressed by users or are implicitly collected by the system, respectively.

Explicit ratings confidently represent the user opinion, even though they can be affected by biases [3] due to: user subjectivity, item popularity or global rating tendency. The first bias depends on arbitrary interpretations of the rating scale. For instance, in a rating scale between 1 and 5, some user could use the value 3 to indicate an interesting item, someone else could use 3 for a not much interesting item. Similarly, popular items tend to be overrated, while unpopular items are usually underrated. Finally, explicit ratings can be affected by global attitudes.

On the other hand, implicit ratings are inferred by the system on the basis of the user-system interaction, which might not match the user opinion. For instance, the system is able to monitor whether a user has watched a live program on a certain channel or whether the user has uninterruptedly watched a movie. Despite explicit ratings are more confidential than implicit ratings in representing the actual user interest towards an item, their collection can be annoying from the user's perspective.

The current deployment of the Fastweb recommender system collects only implicit ratings, but the system is thought to work when implicit and explicit ratings coexist. The rating scale is between 1 and 5, where values less than 3 express negative ratings, values greater or equal to 3 express positive ratings. In absence of explicit information, the rating implicitly inferred by monitoring the user behavior is assumed to be positive (i.e., greater than 3). In fact, whether a user starts watching a certain program, there must be some characteristic of this program appealing for the user (e.g., actors or genre). The fact that in well-know, explicit datasets, such as Netflix and Movielens, the average rating is higher than 3, motivates this assumption. The system treats differently live IPTV programs and VOD content:

IPTV programs. The rating is proportional to the percentage user play time, i.e., the percentage of program the user has watched. Let us assume L is the program play time and t is the user play time. Play times less than 5 minutes are discarded. If a user watched the entire program the rating is 5, if the user watched 5 minutes the rating is 3, otherwise the rating is a value between 3 and 5 given by:

$$\hat{r} = 3 + 2\frac{t-5}{L-5}, \quad 5 \leq t \leq L \tag{1}$$

where t and L are expressed in minutes.

VOD movies. When watching a VOD movie, users explicitly request to buy and to pay for that movie. For that reason, independently from the user play time, when a user requests a VOD movie, the system assign an implicit ratings equals to 4.

As aforementioned, should Fastweb start collecting explicit ratings too, they will naturally coexist with implicit ratings in the URM.

The ratings stored in the URM, before being used by the recommender algorithms, are normalized by subtracting the constant value 2.5. This allows the algorithms to distinguish between positive and negative ratings, because values greater or equals to 2.5 (i.e., 3, 4, and 5) remain positive, while values less than 2.5 (i.e., 1 and 2) become negative. In the rest of the chapter we will assume that the recommender algorithms receive as input a normalized URM.

Finally, users can express *explicit preferences* about the content they would like to watch. For instance, by means of the graphical interface, a user can set his preferred actors. The content-based algorithm explained in Section 4.2 takes into consideration such information and biases the recommended movies toward the expressed preferences.

3.2 Batch and real-time stages

In order to respect the strict real-time requirements, the recommender system and the underlying algorithms follow a model-based approach and have been logically divided into two asynchronous stages, the *batch* stage and the *real-time* stage (see Figure 3):

- The batch stage creates a low dimensional representation (i.e., a model) of the input data. It usually executed during the service off-peak hours, with a frequency which depends on the rate new items/users are added into the system (e.g., once a day).
- The real-time part uses the model in order to serve calls coming from the web services interface and satisfying the real-time constraints. Because of marketing requirements, the output of the real-time component can be post-processed by means of business rules (e.g., pushing up some movies, or filtering some channels).

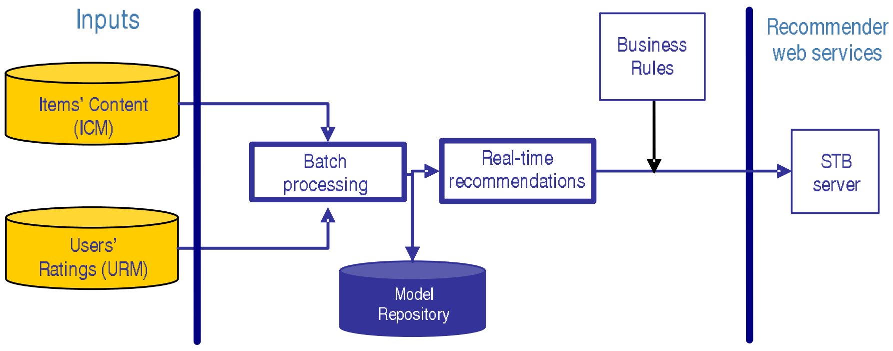


Fig. 3 Recommender system: batch and real-time stage.

Despite such logical division of the recommending process, the model construction in real domains can still be challenging because of input data size and the related time and memory requirements. For this reason, we have implemented high-performing, parallel versions of the most demanding matrix operations, optimized for sparse and big matrices, such as: matrix-matrix and matrix-vector multiplication, matrix transposition, column/row normalization, and singular value decomposition (svd). In particular, svd has been used with two of the three recommender algorithms (one content-based and one collaborative), allowing to greatly reduce the space dimensionality, with benefits both in terms of memory and time complexity. As we will show in the following sections, by its own, svd defines a model of the data, cleaning up the data noise and strengthening the correlation among similar information.

Realistic datasets with millions of users and items can have very pretentious memory requirements. Fortunately, matrices such as URM and ICM are typically very sparse. In fact, most of the users interact (e.g., rate or watch) with very few items with respect to the size of the catalog (e.g., the average users have watched few dozens of movies in a catalog of thousands). Sparse matrices can be treated using very efficient representations. Note that, even though such matrices are sparse, we could have difficulties in maintaining the data in memory. For such reasons, we opted for a solution based on a sort of *memory virtualization*, similar to the swap capability of any operating systems. Differently from the operating system virtual memory, our virtualization policy is tailored ad-hoc for each matrix operation, in order to limit the data exchange between memory and storage.

4 Recommender algorithms

The recommender system implements one content-based algorithm (CB) and two collaborative algorithms (CF):

- a latent-semantic content-based algorithm, referred to as *LSA-CB*
- an item-based collaborative algorithm, referred to as *item-based-CF*
- a dimensionality-reduction-based collaborative algorithm, referred to as *SVD-CF*

In the following section we present a brief overview of recommender algorithms. Section 4.2, 4.3, and 4.4 present the details of the three algorithms, i.e., respectively, the LSA, the item-based, and the dimensionality-reduction algorithms.

4.1 Overview of recommender algorithms

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

The content-based approach to recommendation has its roots in the information retrieval field, which provides a set of tools for searching for textual information, such as in documents, web sites, usenet news and mail messages.

A content-based system is based on the analysis of the content of the items. The model of an item is so composed by a set of features representing its content. The assumption underlying content-based techniques is that the meaning and the relevance of items can be captured by such features:

- each feature has assigned a weight which indicates how representative it is for an item
- similar items contain similar features
- the more items contain a feature, the less representative the feature is (i.e., it is less important in order to distinguish an item from an other)

The feature extraction is probably the most critical phase of such systems and it can be a particularly challenging in IPTV, where resources are non-textual, such as audio/video streams. For instance, the textual features of a movie can be the genre (e.g., commedy), the list of actors, etc. While more interesting information could be obtained by analyzing the audio/video tracks, this technology [10] is fairly recent and it is necessary to examine whether it can really bring some improvement in this specific domain.

The classical way of representing items in content-based recommender is by means of the *bag-of-words* (BOW) approach [5], where we consider textual features and we only retain frequencies of words, discarding any grammar/semantic connection. Usually the words are pre-processed by means of tokenisation, stop-words removal and stemming. The former simply splits text into tokens (e.g., words). Tokens not useful for representing an item in a certain domain are discarded (stop-words). Finally, stemming is used to normalize some kind of grammar variability by converting tokens to their morphological root. For example, the words ’play’, ’player’, ’playing’, and ’played’ would all be converted to their root form, ’play’. After the pre-processing, each token has assigned a weight which is proportional to its frequency normalized using various schemes, the most known being the TF-IDF scheme [20]. The BOW representation can be summarized in the matrix $\mathbf{W}$, where column i represents item i and the element w_{ci} represents the weight (relevance) of token c for item i. Analogously, also users are represented as vectors in the space of tokens. In fact, the profile of a user is derived by composing the vectors corresponding to the items he has rated, suitable weighted with the user rating. Recommendations are then obtained by comparing the similarity between the vector representing the user profile and the vectors representing the items. The most similar items are then proposed to the user. Similarity between two vectors can be expressed by several metrics, such as the euclidean distance and the cosine distance [20].

Content-based systems recommend items similar to those that a user liked in the past, by considering their features. For example, the system analyzes the movies that a user liked in the past and it constructs a model of the user, whose features are the actors, the producers, the genres, the directors, etc., that such user prefers. Then, those movies that have a high degree of similarity to the user's preferences would be recommended. For instance, whether a user is used to watch many action movies, he will be recommended other action movies. This characteristic of content-based recommender systems has two direct effects: it assures that the recommended items are coherent with the user's interests, but, at the same time, the set of recommended items could be obvious and too homogeneous. This issue is usually referred to as *over-specialization* problem [2].

In addition, content-based recommender systems do not usually consider the quality of the items. Thus, if two different items have the same genre, director and actors, they result equivalent, even though one could be a valuable movie and the other a bad one.

The main advantage of content-based techniques is that, since they are based on evident resource features, they can provide an understandable and immediate explanation of the recommended items. Furthermore, content-based filtering is based on a well-know and mature technology.

In opposition to content-based, collaborative systems try to suggest items to a particular user on the basis of the other-users' preferences. In fact, in everyday life, we often rely on recommendations from other people such as by word of mouth or movie reviews. Such systems use the opinions of a community of users to help individuals more effectively identify content of interest. Collaborative systems assist and augment this process. They are based on the following two assumptions:

- There are groups of users with similar tastes, which rate the items similarly.
- Correlated items are rated by a group of users similarly.

The concept of correlation is strongly different to the content-based similarity among items. For instance, here we are saying that whatever the content of a movie is, such movie is considered somehow "similar" to another one because the community expressed the same evaluation for both the movies. For instance, if a user has watched the movie "Forrest Gump", from a collaborative point of view the system could suggest him to watch the movie "The Sixth Sense". The relation among these movies has apparently no sense because, looking at the content, they are not similar movies, but they are actually strongly-correlated because most of the people who have watched "Forrest Gump", also watched "The Sixth Sense".

Starting from the previous two assumption, we can define two classes of collaborative recommenders, respectively, the user-based and the item-based [24]. Both of them are based on social interactions. In practice, user-based recommenders are seldom used because of their poor quality and their memory and time requirements.

Note that collaborative recommendation does not need to extract any feature from the items. Thus, such systems do not have the same shortcomings that content-based systems have. In particular, since collaborative systems use other-users' preferences, they can deal with any kind of content. Furthermore they can recommend any items, even the ones with a content which does not correspond to any item previously liked.

However, also collaborative systems have their own limitations, as described in the following.

The main drawback is that collaborative recommenders are affected by the *new item* (or first-rater) problem. Since such systems recommend the items most preferred by the user's neighborhood, a new item can not be recommended because nobody has rated it so far (the system can not define a model for such item). Therefore, until the new item is rated by a substantial number of users, the recommender system will not be able to recommend it. For such reasons, collaborative algorithms are not practicable in live TV domains, where new programs enter the system at a very high rate and appear and receive ratings for a very limited time window (e.g., few hours). Note that content-based recommenders do not suffer for such a problem because when new items enter the collection their model is given by their own features.

A second issue is called the *sparsity* problem. In fact, the effectiveness of collaborative systems depend on the availability of sets of users with similar preferences. Unfortunately, in any recommender system, the number of ratings already obtained is usually very small compared to the number of ratings to estimate. As a consequence, it might not be possible to recommend someone with unique tastes, because there will not be anyone enough similar to him.

As a consequence of the above two points, at the beginning of its activity, a brand new system will not be able to provide any accurate recommendation; it is called the *cold start* problem. The problem is common to all kinds of recommender systems, but it is particularly evident in collaborative systems since their model is based only on user ratings.

In addition, since popular items are the most rated, collaborative recommenders are likely to be biased toward the most popular items. For instance, if a movies has been rated by only few people, this movie would be recommended very rarely, even if those few users gave high ratings.

4.2 LSA Content-based algorithm

The content-based algorithm implemented in Fastweb is based on the BOW approach, enhanced by means of latent semantic analysis.

Referring to Figure 2, the retrieving of features available from each item in the catalog is performed by the data collector. The features are filtered and weighted, forming the ICM. The features of an item are classified into several groups, referred to as *metadata*. Different kinds of items have different sets of metadata:

- VOD content: actors, directors, producers, title, series title, episode name, studio name, country, year, runtime, synopsis, available languages
- Live IPTV program: actors, directors, producers, channel and time scheduling, country, runtime, year, synopsis

The main difference between VOD and live IPTV content is that the former can be accessed by users at any time upon request, while the latter can only be accessed by users at the time it is broadcasted on a certain channel. This must be taken into consideration by the recommender algorithms.

For any item, each metadata is represented by either a string (e.g., the title) or a vector of strings (e.g., the list of actors). According to the kind of metadata, each string is differently pre-processed and weighted. For instance, whether the metadata contains proper names (e.g., actors) we do not apply any processing, but we simply keep the string as it is. On the other hand, metadata containing sentences (e.g., the synopsis) are tokenized, filtered (stop-word removal) and stemmed. Furthermore, some metadata is more important than others, and so the assigned weights.

In addition to this data pre-processing, the content-based algorithm is powered by LSA (latent semantic analysis), a method well-known in the settings of information retrieval for automatic indexing and searching of documents [14, 7]. The approach takes advantage of the implicit structure (*latent semantic*) in the association of terms with documents. The technique consists in decomposing **W** into a set of orthogonal factors whose linear combination approximates the original matrix. The decomposition is performed by means of singular value decomposition (SVD)

Supposing the ICM is a $c \times n$ matrix (c metadata and n items), it can be factorized into three matrices, **U** ($c \times l$), **S** ($l \times l$), and **V** ($n \times l$) so that:

$$\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{2}$$

where l is the number of latent semantic characteristics of items. Generally l is unknown and it must be computed with cross-validation techniques. **S** contains the first l singular value of **W** that, roughly speaking, are related to the importance of each latent characteristic. The columns of **U** and **V** are orthonormal and represent, respectively, the left and right singular vectors. The product $\mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$ is the best rank-l linear approximation of **W** in terms of the Frobenius norm [19] Note that SVD is unique except for some linear combinations of rows and columns of the three resulting matrices and, conventionally, the diagonal elements of **S** are constructed so to be positive and sorted by decreasing magnitude.

The SVD defines a new vector space, whose dimensions are not the c metadata, but the $l << c$ latent semantic features. We can represent item i into the latent space by projecting (folding-in) the related column of **W**; being $\mathbf{d}_i$ such column vector, its projection $\tilde{\mathbf{d}}_i$ is given by:

$$\tilde{\mathbf{d}} = \mathbf{U}^{\mathrm{T}}\mathbf{d} \tag{3}$$

Similarly, metadata c can be represented into the latent space as the projection of the related row of **W**, referred to as $\mathbf{w}_c$, into the latent space:

$$\tilde{\mathbf{w}} = \mathbf{w}_c\mathbf{V}\mathbf{S} \tag{4}$$

Figure 4 describes the folding-in. Let's observe that we can project back the vectors into the original space, obtaining an approximate representation of the original vector. Although LSA is an approximation of the original BOW space, it has two main advantages.

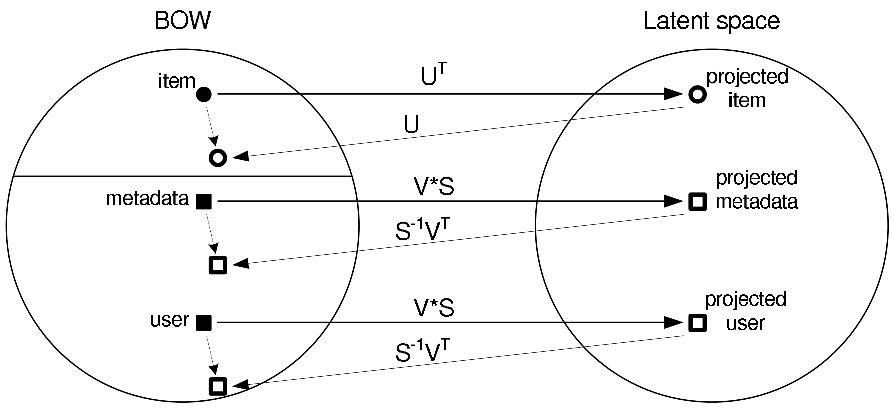


Fig. 4 LSA: folding-in of users, items and metadata into the common latent semantic space.

- It constitutes a great improvement in terms of memory and computation requirements. In fact, once the SVD has been computed by the batch stage, the system works at real-time on the low-dimensional space defined by the l latent semantic dimensions, much smaller than the BOW space.
- By keeping only the l most important characteristics, we filter out the data noise and we strengthen the relationships between items and metadata. For instance, if two metadata co-appear in many items, this means they are somehow correlated and they will be represented similarly in the latent space. The correlation might also be indirect, discovering hidden dependences.

The major issue with SVD is its computational complexity: in fact, in the general case, the decomposition of a $m \times n$ matrix is $O(mn^2)$. Anyway, in the case of sparse matrices, there exist very efficient and scalable solutions. For instance, the SVD implementation by Lanczos [4] is optimized for sparse, large matrices: referring to z as the number of non-zero elements in the URM, the memory requirement is $O(z)$, and the computational complexity is $O(zl)$, i.e., directly proportional to z and to the number of singular values to be computed [25, 22]. In the Fastweb recommender system we have adopted the Lanczos implementation for the SVD, porting it to run on multi-processor architectures.

Recommending items to a user requires to estimate their relevance (rating). Thus, as well as we represented items in the latent space, we represent users in the same space, so that we can compute user-item correlations. A user is represented as a set of ratings and, as well as a row of the ICM (i.e., a metadata), the user ratings can be projected into the latent space by means of (4), where $\mathbf{w}_c$ must be replaced with the user profile, i.e., a row vector of ratings. Once items and users have been represented in the same vector space, we can compute the relevance of item i for user p, referred to as $\hat{r}_{pi}$, by means of any correlation metric among vectors. The metric used in Fastweb is a shrank version of the *cosine*. Assuming that the l-dimensional vectors $\tilde{\mathbf{r}}_p$ and $\tilde{\mathbf{d}}_i$ represent, respectively, the projected user and the projected item, the estimated rating of user p about item i is given by:

$$\hat{r}_{pi} = \frac{\sum_{e=1}^{l} \tilde{r}_{pe} \cdot \tilde{d}_{ie}}{\sqrt{\sum_{e=1}^{l} [\tilde{r}_{pe}]^2} \cdot \sqrt{\sum_{e=1}^{l} [\tilde{d}_{ie}]^2} + c} \tag{5}$$

where, for instance, $\tilde{r}_{pe}$ indicates the e-th element of vector $\tilde{\mathbf{r}}_p$. The constant c is the shrinking factor which corrects the metric in the case of scarce information, i.e., when user or item vectors are meaningless because very close to the origin (e.g., an item with few metadata).

Observe that this representation allows to integrate *explicit* user preferences, e.g., the actors a user has explicitly declared to like. In fact, a vector of explicit user preferences can be treat similarly to an item, i.e., a vector of metadata. Once the explicit preferences have been folded into the latent space, the projected user and the projected explicit preferences can be combined to form a new user profile biased toward the explicit preferences.

4.3 Item-based collaborative algorithm

Item-based collaborative algorithms capture the fundamental relationships among items. As explained in Section 4.1, two items are similar (from a 'collaborative' point of view) if the community agrees about their ratings. Such similarity can be represented in a $m \times m$ matrix, referred to as **D**, where the element d_{ij} expresses the similarity between item i and item j. Note that, potentially, **D** could be non-symmetric, i.e., $d_{ij} \neq d_{ji}$. That means that, for instance, item i could be very similar to item j (thus if a user likes item i he would like item j), even if item j is not similar to item i.

Item-based algorithms represent items in the user-rating space, i.e., an item is a vector whose dimensions are the ratings given by the n users. The coordinate of each dimension is the user rating. As a consequence, item i corresponds to the i-th column of **R**, and the relationships among items are expressed by means of the similarities among the related vectors. In the following sections we describe several techniques to calculate the similarities among these vectors.

According to the system architecture shown in Figure 3, matrix **D** represents the model of the recommender system and its calculation, being computational intensive, is delegated to the batch part of the recommender system. The real-time part generates a recommendation list by using such model. Given the profile of the target user p to recommend (represented by a vector of ratings), we can predict the rating $\hat{r}_{pi}$ by computing the weighted sum of the ratings given by user p on the items similar to i. Such ratings are weighted by the similarity with item i. Referring to Q_i as the set of items similar to i, the prediction of $\hat{r}_{pi}$ can be formulated as:

$$\hat{r}_{pi} = \frac{\sum_{j \in \mathrm{Q}_i} d_{ji} \cdot r_{pj}}{F} \tag{6}$$

where F is a normalization factor. Such factor could be simply set to 1 or, as discussed in [21], it can be computed as:

$$F = \sum_{j \in \mathrm{Q}_i} \left| d_{ji} \right| \tag{7}$$

thus assuring that $\hat{r}_{pi}$ is within the predefined rating scale. Note that, being a model-based approach, user p can be recommended even tough it is not taken into account during the model construction. This allows, for example, (i) to build the model with a subsample of users (e.g., in order to respect time and memory constraints) and (ii) to recommend a user even if his profile is new or update with respect to the moment the model was calculated.

Once computed the predicted ratings for all the items in the dataset that have not been rated by the target user, such ratings are sorted and the N highest-rated items compose the top-N recommendation list.

The set Q_i can be reduced by considering, for instance, only the items with a similarity greater than a certain threshold, or the k most similar items. This latter approach is the classical kNN (k-nearest-neighbors) approach. Section 6 shows that, by varying k, the quality of recommendations varies accordingly.

Alternative ways to compute the real-time prediction are based, for instance, on approximating the ratings by means of regression models [21].

When using implicit datasets, similarity metric is usually computed using a frequency-based approach, as the one discussed by Deshpande and Karypis in [8]. For instance, when we only dispose of binary values, a high similarity between item i and item j means that when someone buys item i, it is very likely that he will buy also item j.

We can treat implicit datasets by considering each item as a vector in the user-rating space, where now the coordinates are binary values. Again, the similarity between two items can be computed as the similarity between the correspondent vectors, for example by means of the cosine metric.

With regard to implicit ratings, the cosine similarity is a special case of a more general approach that we refer in the following as direct relations (DR). In its basis formulation, the item-item matrix **D** used with the DR is given by:

$$\mathbf{D} = \mathbf{R}^{\mathrm{T}} \cdot \mathbf{R} \tag{8}$$

The elements d_{ii} on the principal diagonal is the total number of ratings for item i, while the other elements d_{ij} represent the number of users that have seen both item i and item j. The model (i.e., **D**) can be post-processed by means of a post-normalization, whose general expression is:

$$d_{ij} \leftarrow \frac{d_{ij}}{{d_{ii}}^{\beta} {d_{jj}}^{\gamma} + c} \tag{9}$$

where γ, β, and c are constant parameters whose optimal values depend on the dataset. The constant c is a shrinking factor, correcting the item-item similarity measure where poor information is available.

The model has been further enhanced by considering a kNN (k-nearest-neighborhood) approach. For each item, we consider only the k most similar items (referred to as the item's neighborhood), where k is to be chosen, for instance, by means of cross-validation techniques. By keeping only these items, we discard the noise of the items poorly correlated to the target item, improving the quality of recommendations.

Note that the aforementioned approaches are based on counting the co-rated items and they can be efficiently performed by any DBMS (Database Management System) using simple SQL statements without the need of external programs.

4.4 Dimensionality-reduction-based collaborative algorithm

Collaborative algorithms based on dimensionality reduction techniques describe the dataset (i.e., users and items) by means of a limited set of *features*. These features are different in their meaning from the features typically extracted in the case of content-based algorithms. In fact, the latter are characteristics concerning the content of items (e.g., the genre of a movie, the singer of a song,...), while the features used by collaborative algorithms are not based on the content, but on the implicit way the user community interacts with the items.

Let us assume that an item can be described by means of l features, i.e., it is represented as a vector in the l-dimensional feature space. Similarly, a user is represented by a vector in the same space. As a consequence, the correlation between user p and item i (i.e., how much the item matches the user interests) can be computed as the similarity between the correspondent vectors, for instance by means of their inner product:

$$\hat{r}_{pi} = \sum_{e=1}^{l} a_{pe} \cdot b_{ie} \tag{10}$$

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

The point is to compute the l features which minimize the prediction error between the estimated $\hat{r}_{pi}$ and the actual value r_{pi}.

For instance, Paterek in [17] applies an optimization method, referred to as regularized singular value decomposition, already used in the domain of natural language processing [11]. The l features of users and items are estimated by minimizing the metric RMSE (Root Mean Square Error), one feature at a time, using an optimization technique based on gradient descent. The metric RMSE is defined as:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{p,i} (\hat{r}_{pi} - r_{pi})^2} \tag{11}$$

In this implementation we have used again SVD, that has applied directly to the URM, similarly to the LSA. In fact, the URM can be factorized as:

$$\hat{\mathbf{R}} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^{\mathrm{T}} \tag{12}$$

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

The rating of user p about item i can be predicted as:

$$\hat{r}_{pi} = \sum_{e=1}^{l} u_{pe} \cdot s_{ee} \cdot v_{ie} \tag{13}$$

where u_{pe} is the element in the p-th row and e-th column of **U**, v_{ie} is the element in the i-th row and e-th column of **V**, and s_{ee} is the singular value in the e-th row and e-th column of **S**.

Assuming that $\mathbf{u}_p$ represents the p-row of **U** and $\mathbf{v}_i$ the i-row of **V**, (13) can be rewritten as:

$$\hat{r}_{pi} = \mathbf{u}_p \cdot \mathbf{S} \cdot \mathbf{v}_i^T \tag{14}$$

Reminding that **U** and **V** have orthonormal columns, by multiplying both terms of (12) by **V**, we can state that:

$$\mathbf{u}_p \cdot \mathbf{S} = \mathbf{r}_p \cdot \mathbf{V} \tag{15}$$

where $\mathbf{r}_p$ is the p-th row of **R** (i.e., the profile vector of user p). Consequently, (14) can be reformulated as:

$$\hat{r}_{pi} = \mathbf{r}_p \cdot \mathbf{V} \cdot \mathbf{v}_i{}^{\mathrm{T}} \tag{16}$$

By means of (16) we are able to recommend any user, even if his profile $\mathbf{r}_p$ is new or it has been updated since the model was created (i.e., since the SVD was performed). This represents a great advantage when compared, for instance, with other dimensionality-reduction techniques (e.g., the regularized SVD), where the features for a certain user are pre–computed and fixed during the model construction.

In order to predict all the ratings for user p, (16) can be straightforwardly extended as:

$$\hat{\mathbf{r}}_p = \mathbf{r}_p \cdot \mathbf{V} \cdot \mathbf{V}^{\mathrm{T}} \tag{17}$$

Note that the product between **V** and $\mathbf{V}^{\mathrm{T}}$ results into a $m \times m$ item-item matrix, whose meaning is very similar to the item-item matrix **D** discussed in Section 4.3 about item-based algorithms.

Similarly to LSA, there are several advantages in using such SVD-based approach:

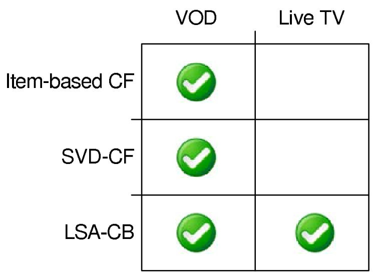
- SVD represents items and users in a low-dimensional space. Once **R** has been factorized, which can result particularly challenging, the system operates with vectors having only l dimensions, much less than the original space of n users and m items.

- SVD reduces the noise in the data. In fact, by neglecting the singular values with low magnitude we are discarding the least-informative data, which is typically noisy [9, 7].
- SVD strengthens the relationships among the data. Thus, if two vectors (either users or items) are similar (because somehow related), they are represented closer in the l-dimensional feature space than in the original space. Observe that the relationship might also be indirect, i.e., by means of the SVD we could discover hidden dependences among users or items.

With regard to the algorithm architecture described in Section 3.2, the matrix factorization (12) is delegated to the batch part, while the prediction of ratings (16) can be performed at real-time. The real-time process estimates the ratings for all the unrated items of the target user, then such ratings are sorted and the N highest-rated items are selected to form the top-N recommendation list. In our tests, the time spent for computing the top-N recommendation list of a certain user by means of (16) is few milliseconds, fitting any real-time requirements.

5 Recommender services

This section presents the implemented recommender services and how they impact into the user interfaces and the IPTV architecture. The recommender system can generate both content-based and collaborative-based recommendations. As summarized in Figure 5, content-based algorithms are applied both to VOD and live TV domains, while collaborative algorithms are applied only to VOD. In fact, we have already observed in Section 4.1 that collaborative algorithms are not practicable in this domain since new programs continuously enter the system, and collaborative algorithms are not able to recommend new items till they are viewed/rated by a substantial number of people.



	VOD	Live TV
Item-based CF	✓	
SVD-CF	✓	
LSA-CB	✓	✓

Fig. 5 Application of recommender algorithms to VOD and live TV.

At the current stage of integration, Fastweb is exposing the full recommender services to a selected set of beta test users before the effective release. The other customers have access to a reduced set of the recommender functionalities. An example of the user interface available by means of the STB is shown in Figure 6.

The services released to the full customer base concern one of the catalog of VOD domain. Recommendations are provided by the LSA-CB algorithm presented in Section 4.2. The choice of fully releasing only the content-based feature of the recommender system is mainly motivated by the cold-start problem and by some legal issues. The content-based algorithm has been preferred for the first few months of activity because collaborative algorithms suffer from the cold-start problem, as explained in Section 4.1. Moreover, collaborative recommenders need to record the behavior of users. This faces Fastweb with delicate legal questions that require, for instance, to obtain authorizations from customers for storing and managing their data, and to implement solutions to grant confidentiality and anonymity of such information.

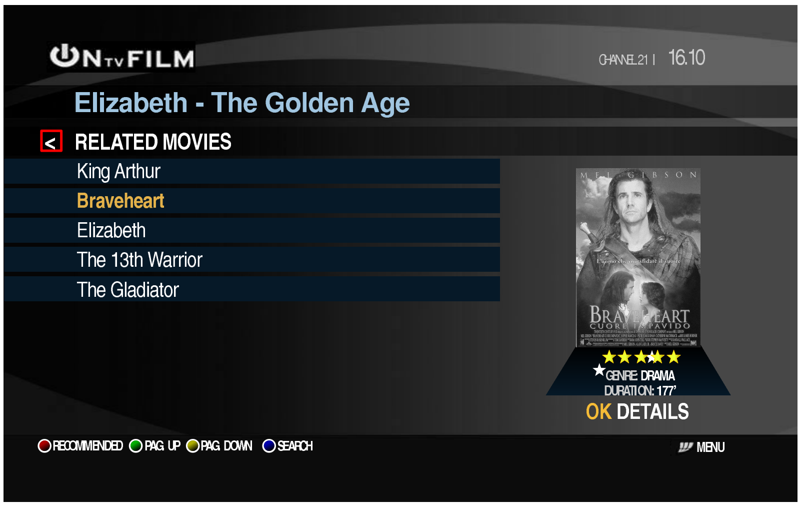


Fig. 6 Recommender system user interface

6 System evaluation

In this section we first discuss the quality of the recommender system by means of accuracy metrics computed adopting an off-line testing. We later present some feedbacks from the on-line analysis of the recommender system.

The off-line tests are based on the views collected during 7 months of users' activity from one of the VOD catalogs. Figure 7 shows the evolution of the number of views. The average number of views per days is about 1600, with peak up to 3300 views during week-end days. Figure 8, 9, and 10 complete the analysis by showing the time evolution of, respectively, the number of active users, the number of active items, and the dataset density. Active users are users that have rated at least one item. Similarly, active items are items that have been rated by at least one user. The dataset density is the ratio between the number of ratings and the product of the number of active users and the number of active items. We can notice from Figure 10 that the trend is not monotone. In fact, when a new user watches her/his first item, we have one new active user, and the dataset decrease its density.

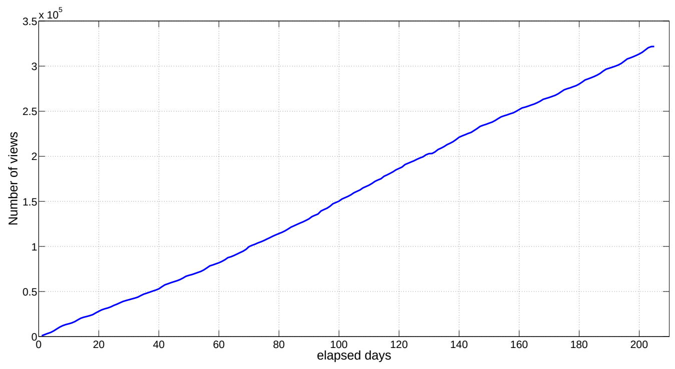


Fig. 7 Number of views collected during 7 months of users' activity from one of the VOD catalogs.

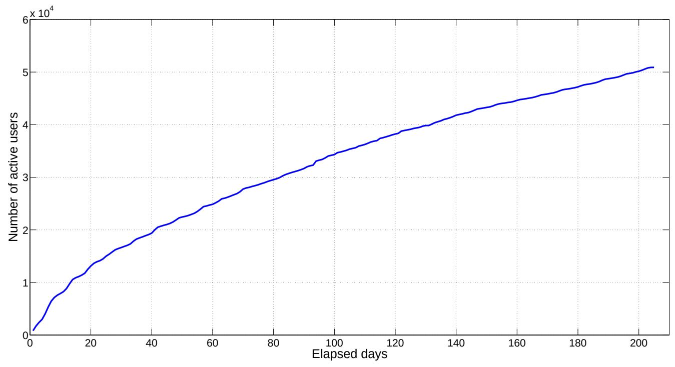


Fig. 8 Number of active users from the same VOD catalog.

6.1 Off-line analysis

Typical approaches for recommender system evaluation are based either on error metrics (e.g., RMSE and MAE) or classification accuracy metrics (e.g., recall, pre-

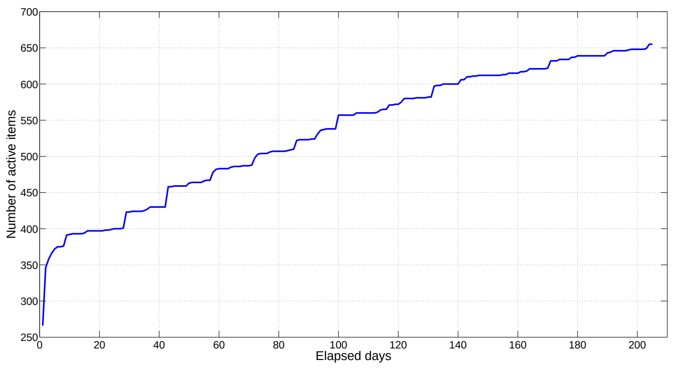


Fig. 9 Number of active items from the same VOD catalog.

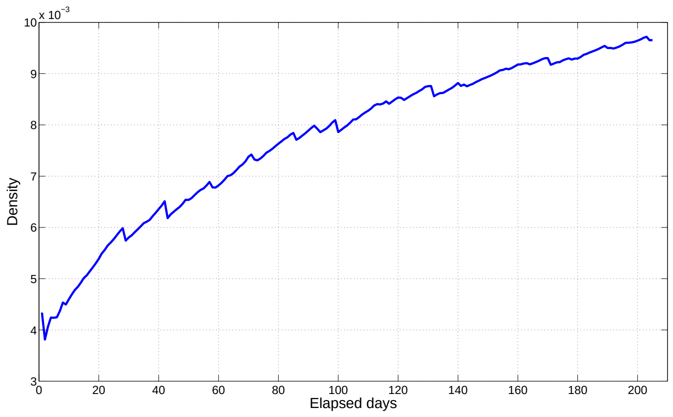


Fig. 10 Evolution of rating density in the same VOD catalog. Density is computed by considering the number of views (i.e., ratings) with respect to the number of active users and active items.

cision, and fall-out) [13, 6]. Since we only dispose of implicit ratings, expressing positive user interests, we are practically constrained in evaluating the quality of the system by means of accuracy metrics. To this end, Tables 1 and 2 present the recall of the three algorithms described in Sections 4.2, 4.3, and 4.4, respectively: the LSA-CB, the item-based-CF and the SVD-CF algorithms.

Recall is often used in information retrieval, where it specifies the percentage of relevant items that have been retrieved by, for instance, a search engine. In our domain, recall indicates how many movies that users have effectively watched are recommended by the recommender algorithm. To this purpose, we follow a leave-one-out approach:

- for each user in the test set, we select one rated item
- the selected item is removed from the user profile, and we generate a recommendation for this modified user profile; items already rated by the user are filtered out from the recommendation list.

- if the removed item is recommended within the first 5 positions we have a *hit*, i.e., a movie that has been watched by a user has been recommended by the algorithm (accordingly to the Fastweb user interface, the recommended list is limited to 5 items);
- the process is repeated for each item and for each user.

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

The test set is selected differently according to the kind of algorithm. In fact, content-based algorithms build their model by using the ICM, and the test set can be the whole URM. On the other hand, the model of collaborative algorithms is based on the URM itself, so they have been evaluated with a 10-fold cross validation approach, i.e., we have randomly split the users into 10 folds and, in turn, one fold has been used as test set for computing the recall, while the remaining nine folds have been used to generate the model. Each test fold is analyzed by means of the leave-one-out approach. The reported results are the average recall among the 10 folds.

The tables report the recall of the recommender algorithms both after 3 months of activity and after 6 months of activity, showing the time evolution of the system. Furthermore, the quality of recommendation of the three algorithms described in Section 4 are compared with a trivial algorithm, used only for comparison purposes: the *top-rated*. The top-rated algorithm is a basic collaborative algorithm that recommends to any user a fix list of items, ordered from the most popular to the less popular (discarding items already rated by the user).

Algorithm	Parameter	Recall	
		3 months	6 months
Item-based-CF	$k = 10$	16.8%	14.9%
	$k = 50$	18.7%	16.4%
	$k = 100$	**19.0**%	**16.6**%
	$k = 150$	18.8%	16.5%
SVD-CF	$l = 5$	15.1%	12.7%
	$l = 15$	12.6%	13.3%
	$l = 25$	10.9%	11.5%
	$l = 50$	9.3%	9.9%
	$l = 100$	6.3%	8.0%
LSA-CB	$l = 50$	1.9%	1.7%
	$l = 100$	2.3%	2.3%
	$l = 150$	2.4%	2.4%
	$l = 200$	2.5%	2.5%
Top-rated		12.2%	7.7%

Table 1 Recommendation quality concerning the considered VOD catalog.

For instance, Table 1 shows that during these 6 months of activity the best algorithm is the item-based collaborative algorithm, and the best configuration is with a neighborhood size k equals to 100. From Table 1 we can observe some particular aspects:

1. in some cases the quality of recommendations after 6 months is lower than after 3 months
2. the quality of the top-rated algorithm is discrete
3. the quality of the content-based is poor, even less than the top-rated algorithm

As for the first observation, we expect that as the system collects ratings, it acquires more precise user profiles and the quality of recommendations shoul improves. However, this is not always true, as, for instance, [6] shows about a family of item-based collaborative algorithms based on naive Bayesian networks (NBN). Furthermore, the analysis we are conducing is not taking into consideration that time evolution concerns not only the ratings, but the items too. Indeed, after 3 months of activity there are 510 active items, while after 6 months we have 621 active items. In terms of probability, after 3 months an algorithm has to pick up 1 items among 510 candidates, while after 6 months the number of candidates is 621, as shown in Figure 9. As a partial counter-effect, while active items are increasing, users rate more items, and algorithms discard these items. Anyway, this minimally compensates the item-increase effect. In fact, while active items increase from 510 to 621, the average profile length increases from about 3 items per user to about 6 items per user, as shown in Figure 11.

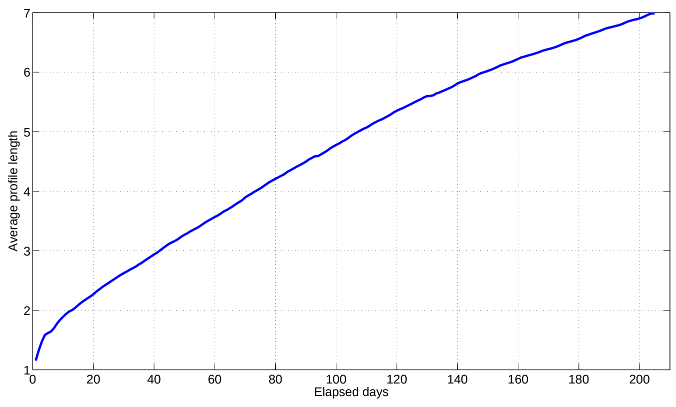


Fig. 11 Time evolution of the average user profile length. Profile lengths are computed on users active in one of the VOD catalogs.

As for the second and the third observation, they find a common explanation. The scarce quality of the content-based algorithm and the high quality of the top-rated algorithm partially depend on the testing methodology based on the leave-one-out approach. Indeed, the recall resulting from leave-one-out is biased toward the recall of the algorithm on popular items, since they are the most numerous, so the most tested. Content-based is extremely disadvantaged because it recommends items regardless of their popularity. On the other hand, top-rated is particularly advantaged because, when the user profiles are short (e.g., during the cold start), most of the users have probably watched the most popular items, as shown in [6]. Furthermore, often users expect novel possibilities from a recommender system and recommending popular items does not address this concept known as *serendipity* [13].

For the above reasons, we present in the following a further evaluation of the quality of the recommender algorithms, where the most popular items have been excluded from the tests and the recall is computed only on the non-popular items, addressing the well-know concept referred to as long-tail [1]. Figure 12 illustrates the distribution of ratings between popular and unpopular items. For instance, we can observe that about the 50% of ratings is concentrated in the 10% of the most-popular items (short-head), while the remaining 90% of items (long-tail) refers only to the 50% of ratings: one of the primary reason for integrating a recommender system is to push up the sells of long-tail items, since they represent potential incoming for a service prodiver. However, recommending long-tail items is harder than recommending short-head items.

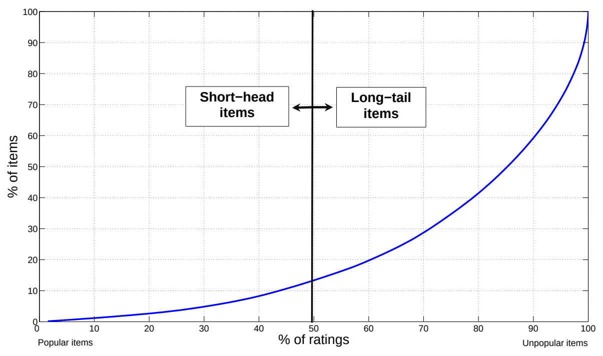


Fig. 12 Long-tail effect: 50% of ratings concerns 10-12% of popular items (short head).

Table 2 reports the recall when the 10 most-popular items have been discarded from testing (referred to as non-top-10), and the recall when the short-head (most-popular) items, representing the 50% of the total number of ratings, have been discarded from testing (referred to as non-top-50%).

From Table 2 we can note that:

1. the quality of the content-based algorithm is constant
2. collaborative algorithms decrease their quality when recommending unpopular items, and top-rated fails.
3. unpopular items are better recommended by the dimensionality-reduction-based collaborative algorithm than by the item-based collaborative algorithm

As for the first observation, the content-based algorithm is confirmed not to be affected by item popularity.

On the contrary, the recall of collaborative algorithms decreases. Among them, the top-rated algorithm quality drastically falls down and, in fact, top-rated is not able to recommend long-tail items.

Algorithm	Parameter	Recall non-top-10		Recall non-top-50%	
		3 months	6 months	3 months	6 months
Item-based-CF	$k = 10$	14.0%	13.2%	7.7%	9.6%
	$k = 50$	**14.0**%	**13.8**%	6.8%	9.0%
	$k = 100$	13.8%	13.5%	6.2%	8.3%
	$k = 150$	13.5%	13.2%	6.1%	7.9%
SVD-CF	$l = 5$	6.6%	6.8%	0.7%	1.4%
	$l = 15$	11.5%	10.2%	1.2%	3.5%
	$l = 25$	12.6%	12.0%	2.2%	4.9%
	$l = 50$	11.4%	11.2%	4.8%	7.8%
	$l = 100$	7.6%	9.3%	**9.8**%	**11.8**%
LSA-CB	$l = 50$	2.1%	1.8%	1.8%	1.7%
	$l = 100$	2.3%	2.3%	2.0%	2.5%
	$l = 150$	2.5%	2.5%	2.1%	2.5%
	$l = 200$	2.6%	2.6%	2.2%	2.6%
Top-rated		0.4%	1.0%	0%	0%

Table 2 Recommendation quality in one of the VOD catalogs for long-tail items, i.e., items not in the top-10 and not in the top-50%, respectively.

Moreover, we can observe that for recommending non-top-10 items the best algorithm is again the item-based collaborative algorithm. However, when we focus on the long-tail (non-top-50%), the dimensionality-reduction-based collaborative algorithm overtakes the item-based. Again, we can observe that the dimensionality-reduction-based collaborative algorithm follows a positive trend as the system collects more ratings, increasing its capability in recommending unpopular items.

6.2 On-line analysis

In this section we integrate the previous results, obtained from an off-line analysis of the recommender algorithms, with an on-line analysis, i.e., we directly study the feedback on the running recommender system. As explained in Section 5, the reported data refer to the content-based algorithm applied on one of the VOD catalogs.

In order to empirically evaluate the recall, we assume that whether a user watches a movie after it has been recommended by the system, such movie is relevant for the user and this represents a *success* for the recommender system.

Let us define the *recommendation success*, which measure the number of movies that have been viewed within a certain time period after being recommended. Indicating with $b(t)$ the recommendation success and with $w(t)$ the number of movies watched by the same users within a time period t from a recommendation, we can compute an *empirical recall* as the percentage ratio between the recommendation success and the number of views:

$$\text{empirical recall}(t) = \frac{b(t)}{w(t)} \tag{18}$$

The empirical recall represents the percentage of views that have been triggered by the recommender algorithm. The specified indexes depend on the time period *t* that is taken into consideration after the recommendation has been provided to the user. Please note that a too long time period *t* could loose the dependency between the recommendation and the view. Table 3 shows the average quality of the system computed by monitoring the views within 2 hours, within 24 hours, and within 7 days from the recommendation. The reported results distinguish between popular and unpopular items.

From the table we can observe that the empiric recall is larger for unpopular movies with respect to popular movies. In fact, popular movies are already known by users, even without being suggested by the recommender system. For instance, either the user has already watched a popular movie (e.g., at cinema) or it is not interested in it.

As a further analysis, about 64% of the recommendation successes refers to unpopular movies (i.e., non-top 50%), while only 36% refers to popular movies (i.e., top 50%), i.e., the recommender system is stimulating users to watch unpopoular movies, with a positive effect on the long-tail.

	2 hours	24 hours	7 days
All	17.0%	19.8%	24.7%
Top 10	5.1%	7.0%	10.6%
Non-top 10	24.2%	27.6%	32.1%
Top 50%	9.4%	11.5%	16.2%
Non-top 50%	28.4%	32.2%	36.1%

Table 3 Average empiric recall on the considered VOD catalog. Results refer to three time periods after the recommendation (2 hours, 24 hours, and 7 days) and are separated between popular and unpopular movies.

Moreover, we highlight the benefits of the recommender system by measuring the *lift factor* that it introduces in the number of views, i.e., the increase of views due to the recommender system. Generally speaking, the number of views in IPTV systems depends on the size of the customer base. Furthermore, we have to take into consideration that new users tend to view more movies than existing users. In addition to a constant incoming of new users, we have bursts of new users in correspondence to marketing campaigns. For instance, Figure 13 shows the trend of the whole Fastweb customer base during more than two-year activity. The peaks of new users are related to promotional campaigns. For privacy reasons, the real number of users is hidden and replaced with a proportional value.

In order to describe the correlation between users and views, we define an autoregressive moving average (ARMAX) model, whose inputs are the current size of the customer base and the number of new users. The parameters of the ARMAX model are estimated and validated by considering 50 weeks of users' activity before the integration of ContentWise. Figure 14 compares the actual number of views with the number of views estimated by the model. In order to smooth daily variability,

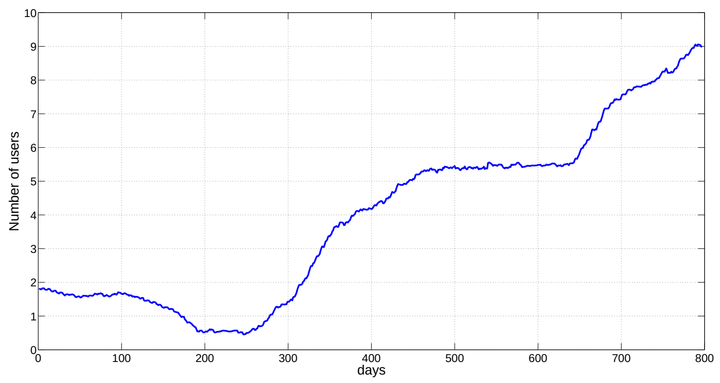


Fig. 13 Number of Fastweb users. The real number of users is proportional to the reported value.

views are aggregated by week. Splitting the data into training and validation sets, the RMSE on the validation set results below 2%.

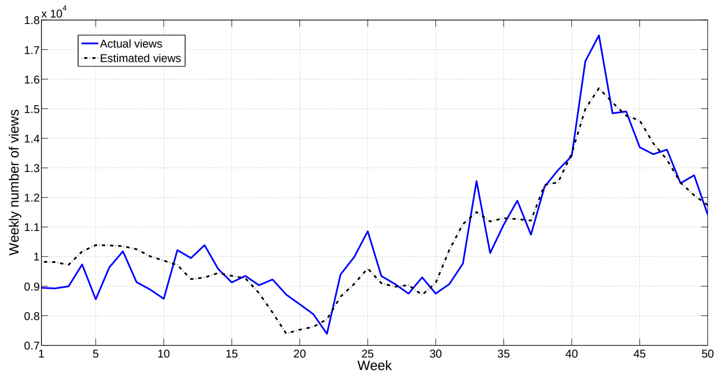


Fig. 14 Weekly number of views *before* the introduction of ContentWise.

The model is then used to estimate the number of views in the first 20 weeks after the integration of the recommender system. As shown in Figure 15, we have an increase of views with respect to the number of views estimated by the model, and this increase can be attributed to the impact of the recommender system, since the other potential factors (e.g., marketing campaigns) are included into the ARMAX model. On average, the lift factor within this period is equals to 15.5%.

Finally, we analyze how users look for interesting content in the considered VOD catalog. Figure 16 shows the daily number of search requests by means of the recommender system, the keyword-based search engine, and the alphabetic browsing, respectively. The gap between the requests to the recommender system and the requests to the other searching tools indicates that users effectively utilize the recommender algorithm to search for movies.

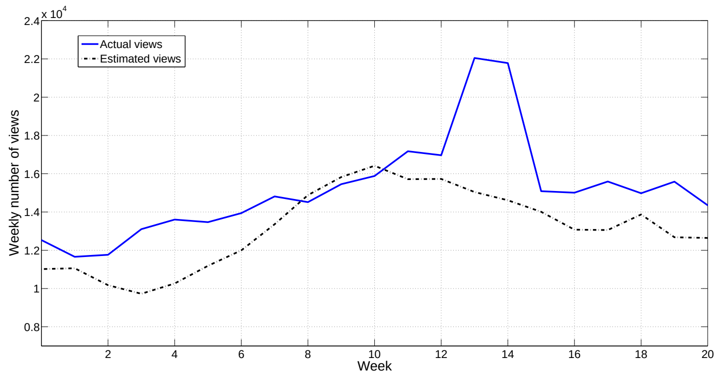


Fig. 15 Weekly number of views *after* the introduction of ContentWise.

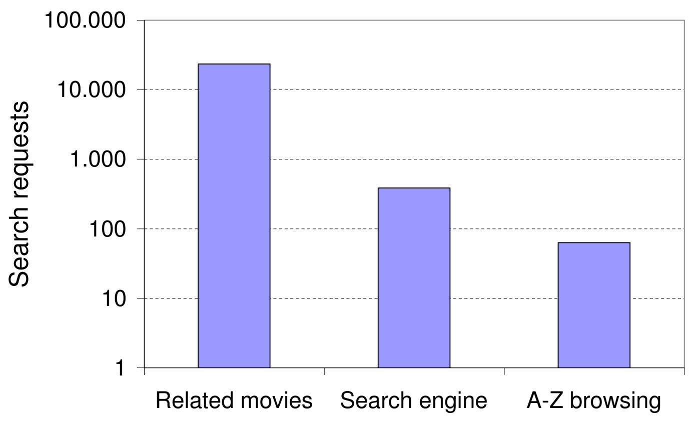


Fig. 16 Comparison among different ways of searching for interesting content: the recommender system (related movies), the keyword-based search engine, and the alphabetic browsing. Values are reported in a logarithmic scale.

7 Conclusions

The integration of the ContentWise recommender systems into the Fastweb architecture positively impacts both the customers and the service provider. Three major considerations derive from the on-line analysis, confirming the positive effects of the recommender system: (i) users prefers to browse the VOD catalog by means of the recommender interface, (ii) users tend to watch recommended movies within few hours, and (iii) users increase the number of watched movies.

Further experiments are currently running on the other catalogs of Fastweb, testing and tuning the quality of all the implemented recommender algorithms and monitoring the cold-start phase of the system in order to complete the release of recommender services.

References

1. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion (2006). URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike04-20{\&}path=ASIN/1401302378`
2. Balabanović, M., Shoham, Y.: Fab: content-based, collaborative recommendation. Commun. ACM **40**(3), 66–72 (1997). DOI http://doi.acm.org/10.1145/245108.245124
3. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. 7th IEEE Int. Conf. on Data Mining pp. 43–52 (2007)
4. Berry, M.W.: Large-scale sparse singular value computations. The International Journal of Supercomputer Applications **6**(1), 13–49 (1992). URL `citeseer.ist.psu.edu/berry92large.html`
5. Chai, K.M.A., Chieu, H.L., Ng, H.T.: Bayesian online classifiers for text classification and filtering pp. 97–104 (2002). DOI http://doi.acm.org/10.1145/564376.564395
6. Cremonesi, P., Lentini, E., Matteucci, M., Turrin, R.: An evaluation methodology for recommender systems. 4th Int. Conf. on Automated Solutions for Cross Media Content and Multi-channel Distribution pp. 224–231 (2008)
7. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science **41**(6), 391–407 (1990). URL `http://citeseer.ist.psu.edu/deerwester90indexing.html`
8. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS) **22**(1), 143–177 (2004). DOI http://doi.acm.org/10.1145/963770.963776
9. Furnas, G.W., Deerwester, S., Dumais, S.T., Landauer, T.K., Harshman, R.A., Streeter, L.A., Lochbaum, K.E.: Information retrieval using a singular value decomposition model of latent semantic structure. pp. 465–480. ACM Press, New York, NY, USA (1988). DOI http://doi.acm.org/10.1145/62437.62487
10. Geneve, U.D., Marchand-maillet, S.: Vision content-based video retrieval: An overview
11. Gorrell, G.: Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing. 11th Conference of the European Chapter of the Association for Compuational Linguistics (2006)
12. Hand, S., Varan, D.: Interactive narratives: Exploring the links between empathy, interactivity and structure pp. 11–19 (2008)
13. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) **22**(1), 5–53 (2004)
14. Husbands, P., Simon, H., Ding, C.: On the use of singular value decomposition for text retrieval (2000). URL `citeseer.ist.psu.edu/article/husbands00use.html`
15. Jensen, J.F.: Interactive television - a brief media history **5066**, 1–10 (2008)
16. Lee, Y., Lee, J., Kim, I., Shin, H.: Reducing iptv channel switching time using h.264 scalable video coding. Consumer Electronics, IEEE Transactions on **54**(2), 912–919 (2008). DOI 10.1109/TCE.2008.4560178
17. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. Proceedings of KDD Cup and Workshop (2007)
18. Rafey, R.A., Gibbs, S., Hoch, M., Gong, H.L.V., Wang, S.: Enabling custom enhancements in digital sports broadcasts pp. 101–107 (2001). DOI http://doi.acm.org/10.1145/363361.363384
19. Saad, Y.: Numerical methods for large eigenvalue problems. Halsted Press New York (1992)
20. Salton, G. (ed.): Automatic text processing. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1988)

21. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. 10th Int. Conf. on World Wide Web pp. 285–295 (2001)
22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of Dimensionality Reduction in Recommender System-A Case Study. Defense Technical Information Center (2000)
23. Sun, J., Gao, S.: Iptv based on ip network and streaming media service station. MIPPR 2007: Remote Sensing and GIS Data Processing and Applications; and Innovative Multispectral Technology and Applications **6790**(1), 67904Q (2007). DOI 10.1117/12.749611. URL `http://link.aip.org/link/?PSI/6790/67904Q/1`
24. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying user-based and item-based collaborative filtering approaches by similarity fusion. pp. 501–508. ACM Press, New York, NY, USA (2006). DOI http://doi.acm.org/10.1145/1148170.1148257
25. Zhang, X., Berry, M.W., Raghavan, P.: Level search schemes for information filtering and retrieval. Information Processing and Management **37**(2), 313–334 (2001). DOI http://dx.doi.org/10.1016/S0306-4573(00)00032-7