Building an IPTV VoD Recommender System: An Experience Report

Krešimir Pripužić[*], Ivana Podnar Žarko[*], Vedran Podobnik[*], Ignac Lovrek[*],
Marko Čavka[‡], Ivan Petković[‡], Petra Stulić[‡] and Mario Gojčeta[‡]
[*] University of Zagreb,
Faculty of Electrical Engineering and Computing, Zagreb, Croatia
Email: kresimir.pripuzic@fer.hr, ivana.podnar@fer.hr, vedran.podobnik@fer.hr, ignac.lovrek@fer.hr
[‡] T-Hrvatski Telekom d.d., Zagreb, Croatia
Email: marko.cavka@t.ht.hr, ivan.petkovic@t.ht.hr, petra.stulic@t.ht.hr, mario.gojceta@t.ht.hr

***Abstract*— Internet Protocol Television (IPTV) is an increasingly popular multimedia service which is used to deliver television, video, audio and other interactive content over proprietary IP-based networks. Video on Demand (VoD) is one of the most popular IPTV services, and is very important for IPTV providers since it represents the second most important revenue stream after monthly subscriptions. In addition to high-quality VoD content, profitable VoD service provisioning requires an enhanced content accessibility to greatly improve end-user experience. Moreover, it is imperative to offer innovative features to attract new customers and retain existing ones. To achieve this goal, IPTV systems typically employ VoD recommendation engines to offer personalized lists of VoD items that are potentially interesting to a user from a large amount of available titles. In practice, a good recommendation engine does not offer popular and well-known titles, but is rather able to identify interesting among less popular items which would otherwise be hard to find. In this paper we report our experience in building a VoD recommendation system. The presented evaluation shows that our recommendation system is able to recommend less popular items while operating under a high load of end-user requests.**

Keywords—recommender system; video on demand; internet protocol television.

I. Introduction

Internet Protocol Television (IPTV) is a system used for television services delivery over Internet protocols such as IP (Internet Protocol), UDP (User Datagram Protocol), RTP (Real-time Transport Protocol), etc. Typical television services comprise live TV, video on demand (VoD) and time-shifted TV. Data transferred via IPTV is prioritized in telecom operator's core network which guaranties quality of service (QoS) and quality of experience (QoE) differencing IPTV from other Internet television services. Moreover, it enables interactivity which is not offered by traditional television.

Finding an interesting VoD item from a list of typically more than a couple of thousand titles while using a typical set-top box remote controller is quite a challenging task, especially since today's users are mostly used to touchscreen devices and/or computer peripherals such as mice and keyboards. Of course, IPTV providers can always spend more money to create better user interfaces, to improve categorization of VoD content or to expand search capabilities. Nevertheless, despite these enhancements, the process of finding interesting titles will stay quite slow due to the nature of the remote controller and regardless of the quality and intuitiveness of end-user interfaces. On the other hand, recommender systems offer a much better experience in finding interesting items by offering a simple end-user interface. For each IPTV user, such an interface shows a personalized list of recommended VoD items from which the user can pick an interesting item easily, and even with a bulky remote controller, which is usually a single available input device for IPTV users. Therefore, offering recommendations to users speeds up the process of finding interesting items and makes it more intuitive compared to alternative solutions.

However, generating a list of interesting items for each user, i.e. recommendations, is a technically complex task. Although there are different recommendation engines available on the market (e.g. Apache Mahout [22], MultiLens [21] and MyMediaLite [23]), neither one is all purpose due to the fact that each one has to be fine-tuned to existing data and/or specific application requirements. From a high level view, recommendation engines either predict whether a particular user will be interested in a particular item or identify a set of *k* items that will be of interest to a certain user. To accomplish this task, recommender engines take users' interests as input and generate personalized lists of recommended items as output. Depending on the application, users' interest can be obtained from various internal data sources, e.g. purchase history and explicit user grades given to previously seen items, and also from different external sources, such as social and demographic data sources, etc.

Alongside recommendation calculation, a typical recommendation service also needs to process all incoming client requests for recommendations from users' devices known as set-top boxes. If the service is unable to process all such requests, this will completely degrade end-user experience, which is not tolerable. Usually, when the frequency of incoming request is high, pre-calculated recommendations have to be stored in main memory, and cached if possible, to keep up with the incoming traffic.

Our main contributions can be summarized as follows:

- We report the architecture and main design choices of our VoD recommender system which is tailored to ensure high-availability and responsiveness while serving user requests with personal recommendations;

- We present our item-based recommendation algorithm which combines content-based and collaborative approaches; and
- We provide an analysis of our recommender system performance for the period of six month of operation to investigate the set of VoD items which are recommended and rented by users. Our goal is to push up the sale of less-popular items, since they represent potentially increased income for a service provider.

The paper is organized as follows. Section II presents the architecture of our recommendation system. In Section III we give a brief overview of recommendation algorithms and explain specifics of the algorithm we use in our implementation. Section IV presents an evaluation of our recommendation system during its operational period of initial 6 months. In Section V we survey related work in the field. Section VI concludes the paper and gives some ideas for future work.

II. Architecture Of The IPTV Recommender System

The main architectural elements of an IPTV system are the head end and middleware. The former is responsible for receiving, digitalizing and encoding live TV signals to IP packets. The latter is the core of an IPTV system that manages user data, content metadata and all IPTV components such as DHCP (Dynamic Host Configuration Protocol), digital rights management and monitoring systems. Besides managing IPTV subsystems, the middleware is responsible for user interactivity and for answering user requests via a graphical user interface. VoD and time-shifted TV can be either part of the middleware or a separate architectural element that consists of data storage devices. Besides the head-end and middleware, there are systems responsible for day-to day-system operations (such as user provisioning and billing) collectively called the operational system support (OSS) and business system support (BSS). A communication endpoint is a set-top box, a customer premises equipment used to receive and decode content from IPTV systems and display it on user's TV.

The architecture of our IPTV recommender system is depicted in Fig. 1. It consists of the three main components:

1. data processing component,
2. data serving component and
3. recommender database.

The data processing component calculates personal recommendations for all VoD service users based on the input data read from the recommender database. These recommendations serve as input for the data serving component whose main task is to generate answers to incoming requests for recommendations from users' set-top boxes. These two components perform completely different types of processing. The data processing component periodically processes data from the database in a batch mode, while the data serving component does the real-time processing of incoming requests to assure uninterrupted performance with low response times. Since recommendation calculation is a processing-intensive task, these two components are separated (on two different servers) to be able to process all incoming requests during peak hours (i.e. highest VoD item rental frequency), similarly to the architecture proposed in [4]. With such a separation, our recommendation system is able to serve all incoming requests without additional unnecessary delays.

The data processing component is responsible for batch processing of large amount of user and VoD item data. It consists of the pre-processing and recommender components. The pre-processing component gathers input data about VoD items and user viewing history (i.e. transactions) to create VoD item profiles and user profiles, which are stored in the

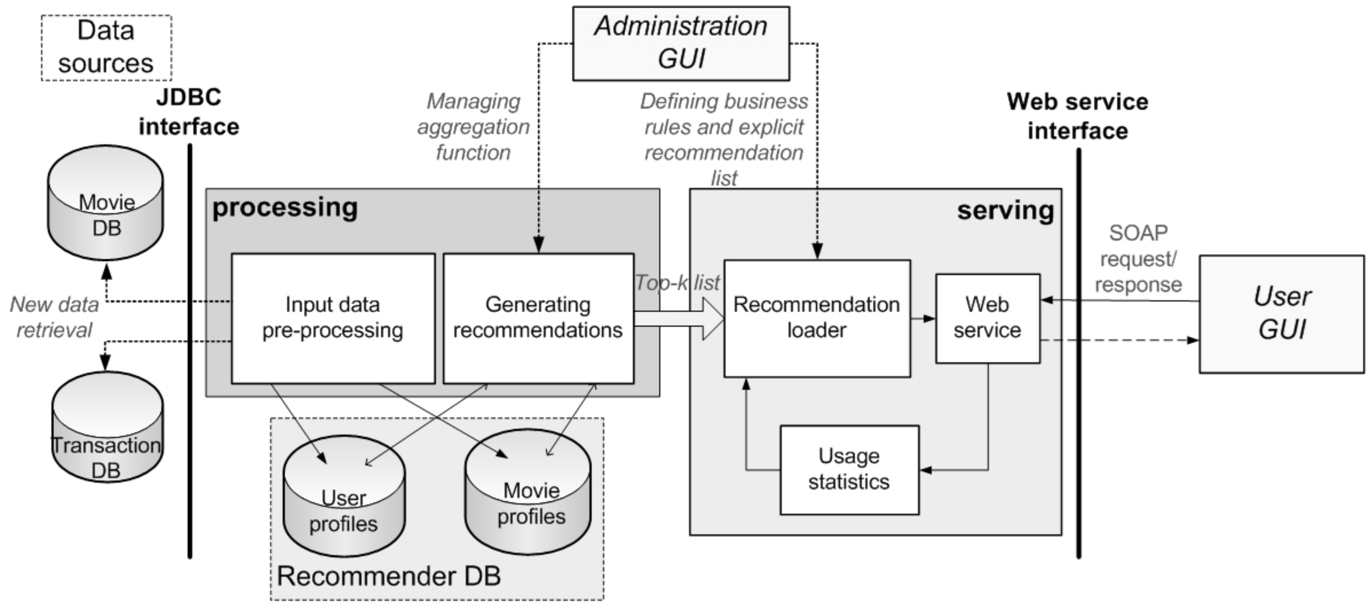

Fig. 1. IPTV Recommender System Architecture

recommender database. First, it filters forbidden VoD items (e.g. adult content) from the list of VoD items that can be recommended to users as well as from viewing history. Second, it creates VoD item profiles, containing data about VoD items which is useful for recommendation, and user profiles. The recommender component implements content and collaborative recommendation algorithms [5] that are used for calculating VoD item *utility function* which tries to predict user interest in a particular item. These algorithms are explained in more detail in Section III. The data processing component produces the following four different data structures, which are stored in the recommender database:

- *Generic recommendations*: a list that contains k_g most popular VoD items.
- *Personal recommendations*: a matrix (number of users by k) in which each row represents the k_p highest ranked (by prediction) recommendations for a user.
- *VoD similarities*: a matrix (number of VoD items by k_s) in which each row represents the k_s most similar VoD items to a VoD item.
- *Average VoD grades*: a map that contains average user grades given to each VoD item.

Data processing component periodically calculates these data structures and stores them in the recommender database. Based on user transaction analysis, off-peak period can be observed around 5 a.m., as shown in Fig. 2. Hence, this period is chosen for the regular daily update of data structures between the data processing and serving components.

In order to offer short response times and to serve all incoming user requests in real-time, the data serving component loads these data structures from the recommender database into its memory. This allows an almost instant reply to each incoming request which is a necessity for the required responsiveness of set-top box user interface and related end-user experience. The data serving component consists of three sub-components: *the recommendation loader*, *web service* and *usage statistics logger*.

The recommendation loader periodically retrieves required data structures from the recommendation database. It is also filters and re-ranks VoD items according to defined business rules. This component is also able to recognize items that were recommended too many times to a single user, but not found interesting, or to remove items that still exist in the recommendations, but were consumed by a user in the meantime.

The web service component represents the main interface that connects our recommendation system with the rest of the IPTV system and implements a web service interface which is defined in the corresponding WSDL (Web Services Description Language) file. The web service implements a number of methods which are invoked by user set-top boxes, e.g. it retrieves a list of recommend VoD items for each user, sets a grade given by a user to a VoD item, gets an average item grade, and enables a user to grant approval that his/her viewing history can be used for calculation of personalized recommendations. Note that the service offers a generic non-personalized list of recommendations to users without consent for recommendation.

The statistic logger measures the quality of given recommendations. It records user requests for recommendations and matches them with rental transaction log, and thus identifies movies that were selected as a result of a given recommendation.

Our recommender system also features a web application for the administrating personnel. This application runs on the same server as the processing component and thus does not represent an overhead for the data serving component. This application is used for:

- Fine-tuning of recommender algorithms,
- Defining various business rules items,
- Modifying generic recommendations,
- Searching the recommender database and showing its content,
- Triggering manual update actions and
- Running various tests on the recommender.

Finally, all components of our recommendation system are implemented in Java EE. To run them, we use two Glassfish 3 [30] instances on two separate machines. As a database server we use PostgreSQL [31] which also runs on a separate machine. All the necessary communication with the database is accomplished through the JDBC (Java Database Connectivity) interface.

III. Algorithms Behind IPTV Recommender System

A. Recommendation Algorithms Overview

The process of predicting whether a user will be interested in a particular item is known as recommendation calculation. In practice, different algorithms can be used for such a calculation, but the most common approaches in the literature are the following [5]: 1) *collaborative*, 2) *content-based* and 3) *hybrid* recommendation algorithms. Even though these algorithms produce the same result in general, i.e. the list of k recommendations for each user, the quality of given recommendations varies, depending on available input data and application of interest. Therefore, when designing a new recommender engine, it is very important to select a suitable recommendation algorithm which will produce satisfactory and interesting recommendations to users.

The traditional collaborative filtering represents users as vectors in the space of size m, which is equal to the number of items in the database. According to whether a user has watched, graded, or not seen an item, the value of the corresponding user vector component is either one, normalized grade or zero, multiplied by the inverse frequency of the item occurrence in previous transactions or grades (of all users). By normalizing given grades we get negative values for poorly graded items, and thus avoid them in future recommendations. The multiplication with the inverse item frequency allows retrieving of interesting items among the less popular by increasing the scores of less popular items at the expense of the scores of popular items. In the first step, this algorithm calculates user vectors and uses them as input into the next step in which the algorithm finds similar users by multiplying user vectors and getting their cosine similarity. In the last step, the algorithm for each user picks the *s* most similar other users, and calculates the scores of his/her non-seen items according to their popularity and grades given by the similar users. The bad side of this algorithm is its large time complexity, e.g. the process of finding similar users requires s^2 vector multiplications, where vector length is equal to *m*. This algorithm thus has severe performance and scaling issues [1].

The item-based collaborative filtering takes a completely different approach to collaborative filtering than the traditional algorithms. To avoid their complexity issues, item-based algorithms calculate a so-called model matrix of size *m* by *m*, whose entries are similarities among the items. These similarities are calculated by an arbitrary metric, e.g. counting the number of users who have seen the same pair of items. Contrary to traditional algorithms, item-based algorithms have to consider each user just once during the process of counting, and when this happens, they need to increase the count values for each item pair among the items that this user has seen. When the counting process is finished, the model matrix always has to be normalized to eliminate the influence of popular items. In the first step this algorithm calculates the model matrix, while in the next step it retrieves k_s similar items for each of the user's graded and seen items. In the last step, it combines those k_s similar items to get the final k_p recommendations. To conclude, although collaborative filtering algorithms have a similar logic, item-based algorithms have much better time complexity and thus scale better.

Clustering-based recommendation algorithms try to group similar users by clustering them according to some arbitrary similarity metric, such as the number of mutually seen or similarly rated items. Therefore, these algorithms treat the task of finding similar users as a typical classification task. To get recommendations for a user, these algorithms calculate the scores of unseen items according to their popularity and/or grades given by the users in the same cluster. These algorithms work similarly as collaborative recommendation algorithms, but perform and scale better than them [2].

Content-based algorithms calculate the similarity between items according to similarities of their content. Usually, the content of an item is transformed to a query which is then executed on other items to find similar ones. These algorithms are very sensitive to the quantity of available information about items. More available information will produce better results as the similarities would be calculated more precisely. In the first step, these algorithms calculate item similarities, to be used in the second step while retrieving k_s similar items for each of the user's graded and seen items. In the last step, these algorithms combine k_s similar items to get the final k_p recommendations. The problem with content-based methods is overspecialization which often results in recommending only items that are very similar to those already rated or seen by a user.

Neither of the previously mentioned algorithms is perfect due to the fact that they are not prone to the new user and/or new item problems [3]. When a new user enters the system, we cannot identify similar users since we do not have any information about this user. Additionally, this user has not seen nor graded any item and thus we cannot identify items which are similar to his/her previously seen or graded items. Content-based algorithms are able to start recommending new items very soon, especially if there are very similar to the popular ones. Other algorithms cannot recommend new items because such items are not yet seen nor graded. Obviously, to solve these problems and get better results we have to combine these approaches and use a hybrid recommendation algorithm.

B. *Our Recommendation Algorithm*

In our recommender system we use a hybrid approach which combines item-based collaborative filtering and content-based algorithms as suggested in [4]. The inputs into our algorithm are VoD item profiles and user grades and transactions, while the outputs are data structures explained in Section II: *Generic recommendations*, *Average VoD grades*, *VoD similarities* and *Personal recommendations*. The first two data structures are calculated directly from user transactions and grades. To get the latter two data structures we first need to calculate three model matrices of size *m* by *m* which represent different similarities of VoD items:

- Transaction-based model matrix – contains similarities of items according to users' transactions,
- Grade-based model matrix – contains similarities of items according to users' grades and
- Content-based model matrix – contains similarities of items according to the items' content.

To calculate the similarity of any two VoD items in the grade-based model matrix we first identify all users which have graded both of these items. For each such user we then calculate the product of normalized grades of the two items. We normalize grades to get negative values for poorly graded, and positive for the well graded items. Finally, we sum all such products to calculate similarities between pairs of items.

The transaction-based model matrix is calculated by counting all users who have seen the same pair of VoD items. This way, any two items are similar to each other if the same group of users seen both of them. During matrix calculation we ignore all item pairs which are graded because the grade-based model matrix is more important for recommendation than the transaction-based model matrix.

The similarities of VoD items in the content-based model matrix are calculated by comparing VoD item profiles which contain both structured information (e.g. category, genre, actors, directors, scenarists, artists) and unstructured text (e.g. title, short and long synopsis). To compare the profiles we use information retrieval methods [28]. In practice we have implemented this component in Java EE using Apache Lucene [29], a text search engine library written entirely in Java. We represent VoD item profiles as vectors in a multidimensional space. As a measure of similarity between two vectors in this space we use the cosine similarity between them, i.e. the cosine of the angle between these vectors.

We first normalize each of these matrices and then produce a weighted sum of them to get the combined model matrix,

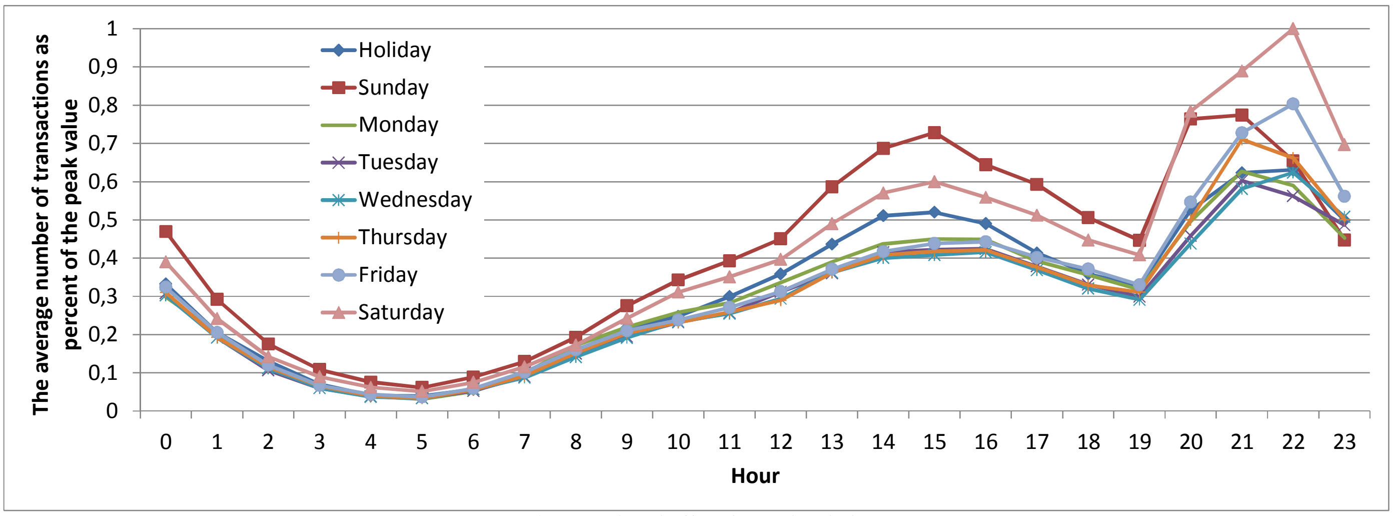

Fig. 2. Peak and off-peak rental periods

which is mentioned in Section II as *VoD similarities* data structure. The first two matrices need to be normalized to reduce the influence of popular items. We also normalize the third matrix to make it comparable to the first two matrices before summarizing them into the combined model matrix. Please note that we conducted a special evaluation of the recommendation quality for different matrix weights. The best results were obtained when matrix weights were in the following order: $w_g > w_t > w_c$, where w_g, w_t and w_c represent weights of the grade-based, transaction-based and content-based matrix, respectively.

Finally, we use the combined model matrix as the base for our item-based recommendation algorithm which calculates *Personal recommendations* as previously explained in the recommendation algorithm overview. Thus our hybrid recommendation algorithm handles new user and new item problems efficiently, while being both scalable and time-efficient.

IV. Evaluation Of The IPTV Recommender System

After the initial service rollout which was completed by the end of March 2012, our recommender system has gradually attracted an increasing number of users that are requesting personal recommendations, and has reached almost 25% of VoD users by the end of December 2012. In this section we investigate and compare the set of VoD items that were rented before the recommender system release and after its release. In particular, the following figures and analysis have been produced based on two six month periods: the first one is for a six month period before March 2012, while the second one is for the period July 2012 – December 2012 when the recommender system was fully operational.

Fig. 2 shows the average number of observed transactions, i.e. VoD item rentals, per hour on different week days as a percentage of the peak value. As we can see, there are two peaks and one off-peak during the day. The first peak is in the afternoon, while the other is in the evening. The off-peak is always around 5 a.m. and thus we selected this time to do the data exchange between components of our recommendations system. We also see that the highest peak is on Saturday evening, which is not surprising at all because most people stay at home and rest during this period. Sundays and holidays are very popular during the day, but not so much in the evening.

In Fig. 3 we can see the popularity of VoD items sorted by the relative number of rentals before enabling our recommender system. As expected, this curve follows the Zipf law [24] as we see a few very popular items followed by the long tail of less popular items. By developing our recommendations system we wanted to change the shape of this curve by recommending less popular VoD items which are interesting to users, but very hard to find without the recommender.

The following two figures investigate the set of VoD items that was rented by users as a consequence of given recommendations produced by our recommendation engine. We are interested to reveal whether the titles rented through our recommender system are indeed less popular and less-known titles or not.

Fig. 5 investigates the characteristics of VoD items that were rented due to personal recommendations. First, it depicts the popularity of VoD items that were rented following a personal recommendation measured by the total number of item rentals during a six month period (regardless of the means by which a user has found an item in the VoD catalog).

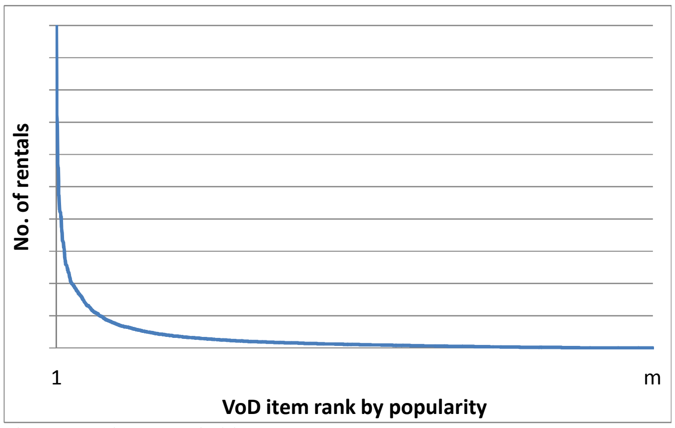

Fig. 3. VoD item popularities

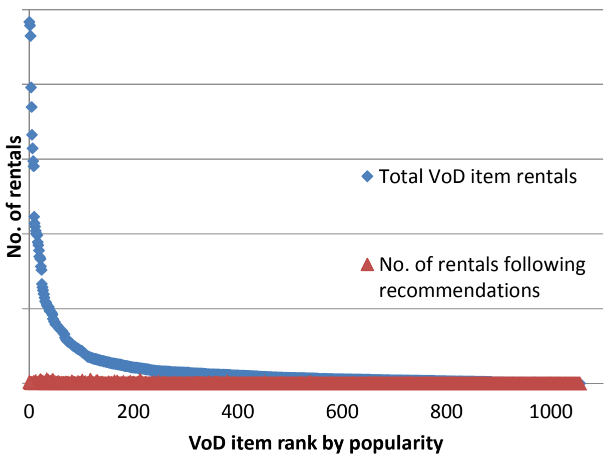

Fig. 5. VoD items rented due to personal recommendations

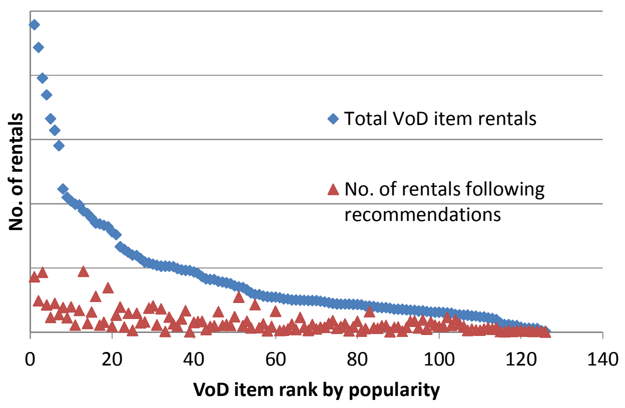

Fig. 6. VoD item rentals following generic recommendations

Second, it depicts the number of VoD item rentals which occurred after a corresponding personal recommendation, and are thus considered as its consequence. It is visible that size of the subset of VoD items recommended and rented through the use of our recommender system is over 1000 items, while their popularity follows Zipf distribution which means that most of the items from personal recommendations are less-popular and less-known items. Furthermore, 69% of the titles rented due to personal recommendations are unpopular titles, while only 10% of the titles are popular and well-known titles. Finally, one can notice that the majority of rentals for less-known items is a direct consequence of personal recommendations, and such

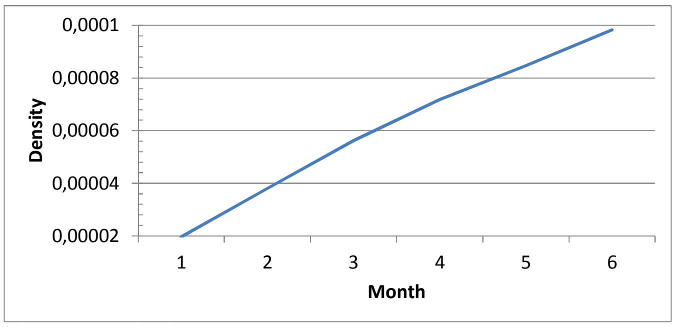

Fig. 4. Dataset density

titles would hardly ever been rented without a recommender system.

In contrast to personal recommendations, Fig. 6 depicts the popularity of VoD items that were rented following generic recommendations. One can notice that the diversity of items is as expected limited to a small number of VoD items (10 times smaller than the previous set size). The popularity of this set of items does not follow the Zipf distribution as these are mostly popular VoD items. In particular, even 40% of the rentals are the ones attributed to extremely popular VoD items. On a positive side, one can notice that a large fraction of rentals for such popular items is a consequence of our generic recommendation.

Our recommender system is indeed stimulating users to watch unpopular and less-known titles, with a positive effect on the long-tail. It is visible in Fig. 7 which shows the popularity of rented VoD items before and after the introduction of our recommender system.

Fig. 4 shows the density of ratings dataset during the six months period July 2012 and December 2012 when the recommender system was fully operational. The dataset density is defined as the ratio between the number of ratings and the product of the number of active users and the number of active items. As we can see in the figure, our recommender had a constant growth in the number of ratings during this period which is respectable.

Finally, Fig. 8 shows the average number of observed recommendations per hour on different week days. As we can see, there are also two peaks and one off-peak during the day, similarly to observed transactions, i.e. VoD item rentals, shown in Fig. 2. We conclude that users of our recommender use it

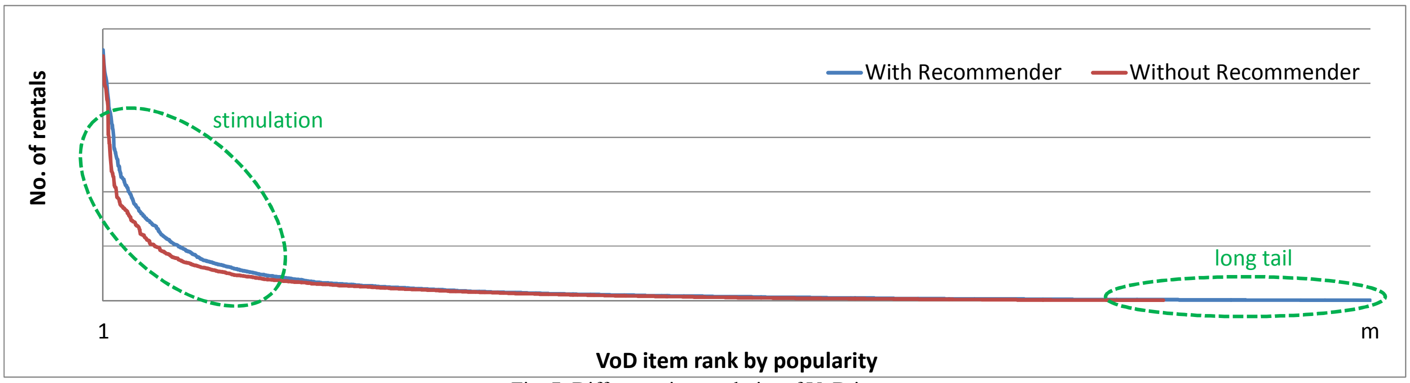

Fig. 7. Difference in popularity of VoD items

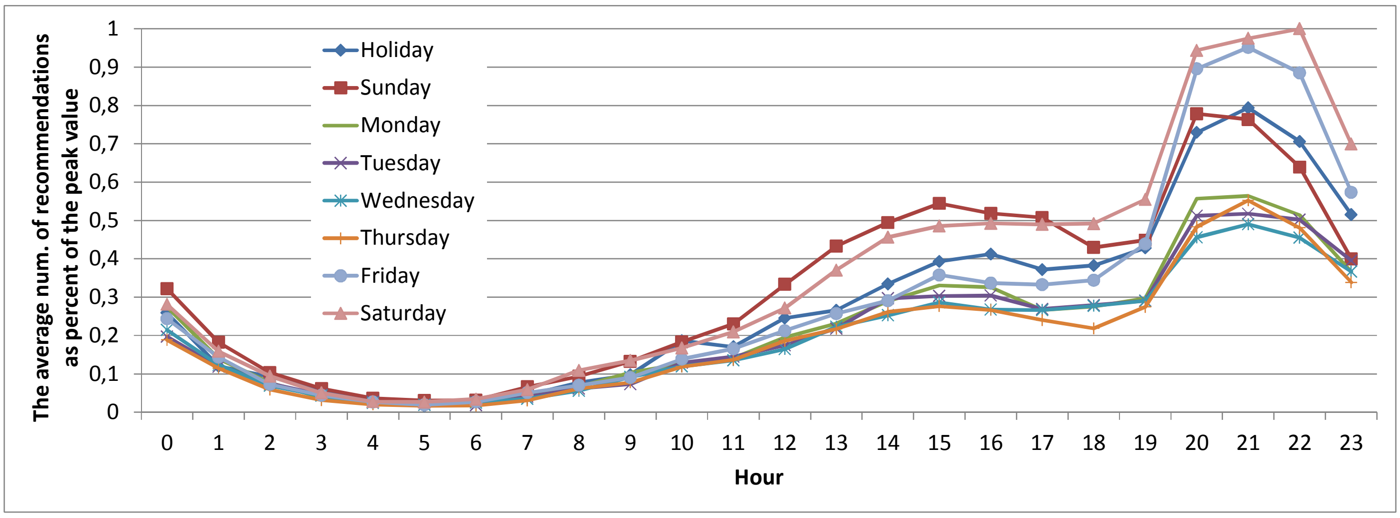

Fig. 8. Peak and off-peak recommendation periods

exactly at the same time when they mostly watch VoD content. This represents an additional confirmation of usage among the users.

V. Related Work

It is really difficult to enumerate all different fields in which recommender systems are applied. For example, they are used in e-commerce [6] [7] [25], for recommending music [8], movies [9] [10], scientific papers [11] etc.

GroupLens [12] was one of the first implementations of recommender systems. Unlike our hybrid technique, it uses the traditional collaborative filtering approach to calculate new recommendations. Later, the same research group developed MovieLens, a movie recommender system which is also based on the traditional collaborative filtering. In [18] the authors propose a clustering approach based on semi-supervised learning in order to increase recommendation accuracy. Based on this approach, they developed a movie recommendation system which combines both the content-based and collaborative information about movies. In [19] the authors present a hybrid recommender system that combines content-based and collaborative filtering through the training of neural network classifiers. In order to create a small and sufficient recommendation set, this system combines collaborative results with Boolean and fuzzy content aggregation. More interactive movie recommender system, named MovieGEN is presented in [20]. Basically, this is a hybrid recommendation system, which uses machine learning and cluster analysis to calculate recommendations. However, this system uses personal information of users to predict their movie preferences using well-trained support vector machine (SVM) models. Then, based on SVM predictions, it selects movies from the dataset, clusters them and generates questions to users. Finally, it uses the information collected through the answers to refine user recommendation lists.

ContentWise IPTV recommender system [5] is a commercial recommender system that generates social recommendations based on the activity of similar users, analysis of social network profiles and friend user activities. Additionally, it analyses live program information from electronic program guide (EPG) and performs a time division of recommendations in order to distinguish different users of the same set-top box. Jinni [27] is a semantic search engine and recommender for movies and TV content. While other engines only analyze standard movie information (such as a title, list of actors, genre, director, scenarist, synopsis) Jinni tries to understand the meaning from the data by analyzing additional movie information including mood, plot, time/period, place, etc. This approach allows semantic content search, and enables quick adaption to user interests.

In [14] the authors introduce contextual pre-filtering technique based on implicit user feedback which splits user profiles into several sub-profiles, called micro-profiles, each representing user in particular context. Two different context-aware techniques to movie recommendation are presented in [17]: 1) increasing the performance of traditional CF approaches by using contextual time-related factors, and 2) using machine learning techniques to identify a household user that submitted a given rating. Several other authors also proposed exploiting of temporal information in order to improve recommendation quality. For example, [13] proposes a technique for modeling temporal dynamics of customer preferences by separating transient factors from the lasting ones. An interesting location-based recommender service which also takes temporal context into account is presented in [15], while [16] presents a novel approach to collaborative recommender systems based on implicit user feedback and temporal information about user purchase time and item launch time to achieve better recommendation accuracy.

VI. Conclusions

Today's continued trend for service personalization is demonstrated by a growing popularity of social networks [26] and interactive applications. Thus, the core IPTV services (LiveTV and VoD) need to be enriched with additional applications which increase user's Quality of Experience and IPTV service portfolio. Additionally, mature IPTV VoD libraries have a substantial number of items, which makes it difficult for a user to choose a satisfactory title at a given time. A personalized recommender system has the potential to solve both mentioned problems while providing a simple user interface. In addition to demands for improved and innovative applications, telecom operators put an emphasis on user data security and privacy, as well as solution robustness and scalability. Moreover, any solution being developed for a telecom operator needs to be customized in order to adapt to operator's legacy systems and business logic. The listed requirements were taken into account when designing and implementing our recommender system for a VoD catalog to serve personalized lists of recommended titles to service users.

This paper reports our experience in building a VoD recommendation system. It presents the developed architecture and our main design choices which led to a highly-responsive and robust implementation. Furthermore, we outline our item-based hybrid recommender algorithm and present an evaluation report which shows that our recommendation system is indeed able to recommend less popular VoD titles such that it offers a diversity of titles to end-users.

Future work will be directed to introduce context into our recommendations such that, e.g., the time of the day or preferred VoD category influence the list of recommended titles. All these efforts have a goal to improve the usability of our service while having a positive effect on operator's revenue.

ACKNOWLEDGMENT

This work was carried out within the research project "Content Delivery and Mobility of Users and Services in New Generation Networks", supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

REFERENCES

[1] G. Linden, B. Smith, and J. York. Amazon.com recommendations: itemto-item collaborative filtering. IEEE Internet Computing, 7(1):76–80,

[2] 2003. J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1998, pp. 43-52.

[3] G. Adomavicius, and A. Tuzhilin Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. In IEEE Transactions on Knowledge And Data Engineering, Vol 17, No. 6, June 2005

[4] M. Deshpande and G. Karypis, "Item-Based Top-N Recommendation Algorithms," ACM Trans. Information Systems, vol. 22, no. 1, pp. 143-177, 2004.

[5] Riccardo Bambini, Paolo Cremonesi, Roberto Turrin: A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. Recommender Systems Handbook 2011: 299-331

[6] W. Hsiao-Fan and W. Cheng-Ting: A strategy-oriented operation module for recommender systems in E-commerce. Computers & Operations Research, 2012.

[7] Y. Cao and Y. Li: An intelligent fuzzy-based recommendation system for consumer electronic products. Expert Systems with Applications, vol. 33, pp. 230-240, 2007.

[8] K. Yoshii, M. Goto, K. Komatani, T. Ogata and H. G. Okuno: An Efficient Hybrid Music Recommender System Using an Incrementally Trainable Probabilistic Generative Model IEEE Transactions on Audio, speech, and language processing, Vol. 16, No. 2., 2008.

[9] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, and J. Riedl, "MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System," Proc. Int'l Conf. Intelligent User Interfaces, 2003.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. IEEE Computer, 42(8):30–37, 2009.

[11] S. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. Lam, A. Rashid, J. Konstan, and J. Riedl.: On the recommending of citations for research papers. In ACM Conference on Computer Supported Cooperative Work, 2002.

[12] P. Resnick et al., "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," Proc. Conf. Computer Supported Cooperative Work (CSCW 94), ACM Press, pp. 175–186, 1994.

[13] Y. Koren. Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, pages 447–456, New York, NY, USA, 2009. ACM.

[14] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In Proceedings of the Context-aware Recommender Systems Workshop at Recsys09, 2009.

[15] C. Biancalana, F. Gasparetti, A. Micarelli, and G. Sansonetti. An approach to social recommendation for context-aware mobile services. ACM Trans. Intell. Syst. Technol., 2011.

[16] T. Q. Lee, Y. Park, and Y.-T. Park. A time-based approach to effective recommender systems using implicit feedback. Expert Syst. Appl., 34:3055–3062, May 2008.

[17] C. Biancalana, F. Gasparetti, A.Micarelli, A. Miola, G. Sansonetti: Context-aware Movie Recommendation based on Signal Processing and Machine Learning, CAMRa2011, Chicago, IL, USA, October 2011.

[18] C. Christakou, L. Lefakis, S. Vrettos, A. Stafylopatis: A Movie Recommender System Based on Semi-supervised Clustering, Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Nov 2005.

[19] C. Christakou, S. Vrettos, A. Stafylopatis: A Hybrid Movie Recommender Sytem Based on Neural Networks, International Journal on Artificial Intelligence Tools, Vol. 16, No. 5, 771-792, 2007.

[20] E. A. Eyjolfsdottir, G. Tilak, N. Li: MovieGEN: A Movie Recommendation System, Computer Science Department, University of California Santa Barbara, unpublished

[21] B. N. Miller. Toward a Personal Recommender System. PhD thesis, University of Minnesota, 2002.

[22] Apache Mahout. http://mahout.apache.org/

[23] Zeno Gantner, Steen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011), 2011.

[24] G.K. Zipf. Human Behavior and the Principle of Least Effort, Addison-Wesley, 1949.

[25] Sambolec, Igor; Rukavina, Ivan; Podobnik, Vedran. RecoMMobile: A spatiotemporal recommender system for mobile users. (pdf) Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2011). Split, Croatia: IEEE, 2011.

[26] Podobnik, Vedran; Striga, Darko; Jandras, Ana; Lovrek, Ignac. How to Calculate Trust between Social Network Users?. (pdf) Proceedings of the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2012). Split, Croatia: IEEE, 2012.

[27] Jinni. http://www.jinni.com/

[28] B. Y. Ricardo and R. N. Berthier, Modern Information Retrieval, ACM Press, New York, 1999.

[29] Apache Lucene. http://lucene.apache.org

[30] GlassFish. http://glassfish.java.net/

[31] PostgreSQL. http://www.postgresql.org/