

# Big & Personal: data and models behind Netflix recommendations

Xavier Amatriain  
Netflix  
xavier@netflix.com

## ABSTRACT

Since the Netflix \$1 million Prize, announced in 2006, our company has been known to have personalization at the core of our product. Even at that point in time, the dataset that we released was considered "large", and we stirred innovation in the (Big) Data Mining research field. Our current product offering is now focused around instant video streaming, and our data is now many orders of magnitude larger. Not only do we have many more users in many more countries, but we also receive many more streams of data. Besides the ratings, we now also use information such as what our members play, browse, or search.

In this paper, we will discuss the different approaches we follow to deal with these large streams of data in order to extract information for personalizing our service. We will describe some of the machine learning models used, as well as the architectures that allow us to combine complex offline batch processes with real-time data streams.

## 1. INTRODUCTION

Recommender Systems are a prime example of the mainstream applicability of large scale data mining. Applications such as e-commerce, search, Internet music and video, gaming or even online dating make use of similar techniques to mine large volumes of data to better match their users' needs in a personalized fashion.

There is more to a good recommender system than the data mining technique. Issues such as the user interaction design, outside the scope of this paper, may have a deep impact on the effectiveness of an approach. But given an existing application, an improvement in the algorithm can have a value of millions of dollars. On the other hand, given an existing method or algorithm, adding more features coming from different data sources can also result in a significant improvement.

We will start by describing some of the lessons learned from the Netflix Prize (see Section 2), and then explain the current use of data, and models at Netflix in section 3. All

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

*BigMine'13* August 2013, Chicago, Illinois, USA

Copyright is held by the owner/author(s). Publication rights licensed to ACM. ...\$15.00.

of these Big Data Mining techniques would not be possible without a proper architecture that is able to leverage large amounts of data, and scale the computation. In section 4, I will give an overview of the Netflix architecture that combines offline, nearline, and real-time processing.

## 2. LESSONS FROM THE NETFLIX PRIZE

In 2006, Netflix announced the Netflix Prize, a machine learning and data mining competition for movie rating prediction. We offered \$1 million to whoever improved the accuracy of our existing system called Cinematch by 10%. We conducted this competition to find new ways to improve the recommendations we provide to our members, which is a key part of our business. However, we had to come up with a proxy question that was easier to evaluate and quantify: the root mean squared error (RMSE) of the predicted rating.

The Netflix Prize put the spotlight on Recommender Systems and the value of user data to generate personalized recommendations. It did so by providing a crisp problem definition that enabled thousands of teams to focus on improving a metric. While this was a simplification of the recommendation problem, there were many lessons learned.

A year into the competition, the Korbell team won the first Progress Prize with an 8.43% improvement. They reported more than 2000 hours of work in order to come up with the final combination of 107 algorithms that gave them this prize. And they gave us the source code. We looked at the two underlying algorithms with the best performance in the ensemble: Matrix Factorization (MF) [12]<sup>1</sup> and Restricted Boltzmann Machines (RBM) [16]. Matrix Factorization by itself provided a 0.8914 RMSE, while RBM alone provided a competitive but slightly worse 0.8990 RMSE. A linear blend of these two reduced the error to 0.88. To put these algorithms to use, we had to work to overcome some limitations, for instance that they were built to handle 100 million ratings, instead of the more than 5 billion that we have, and that they were not built to adapt as members added more ratings. But once we overcame those challenges, we put the two algorithms into production, where they are still used as part of our recommendation engine.

One of the most interesting findings during the Netflix Prize came out of a blog post. Simon Funk introduced an incremental, iterative, and approximate way to compute the

<sup>1</sup>The application of Matrix Factorization to the task of rating prediction closely resembles the technique known as Singular Value Decomposition used, for example, to identify latent factors in Information Retrieval. Therefore, it is common to see people referring to this MF solution as SVD.

SVD using gradient descent [7]. This provided a practical way to scale matrix factorization methods to large datasets. Another enhancement to matrix factorization methods was Koren *et. al.*'s SVD++ [10]. This asymmetric variation enables adding both implicit and explicit feedback, and removes the need for parameterizing the users.

The second model that proved successful was the Restricted Boltzmann Machine (RBM). For the Netflix Prize, Salakhutdinov *et al.* proposed an RBM structure with binary hidden units and softmax visible units with 5 biases only for the movies the user rated [16].

Many other lessons came out of the Prize. For example, the matrix factorization methods mentioned above were combined with the traditional neighborhood approaches [10]. Also, early in the prize, it became clear that it was important to take into account temporal dynamics in the user feedback [11].

Another finding of the Netflix Prize was the realization that user explicit ratings are noisy. This was already known in the literature. Herlocker *et al.* [8] coined the term “magic barrier” to refer to the limit in accuracy in a recommender system due to the natural variability in the ratings. This limit was in fact relatively close to the actual Prize threshold [3], and might have played a role in why it took so much effort to squeeze the last fractions of RMSE.

The final Grand Prize ensemble that won the \$1M two years later was a truly impressive compilation and culmination of years of work, blending hundreds of predictive models to finally cross the finish line [4]. The way that the final solution was accomplished by combining many independent models also highlighted the power of using ensembles. At Netflix, we evaluated some of the new methods included in the final solution. The additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment. Also, our focus on improving Netflix personalization had by then shifted from pure rating prediction to the next level. In the next section, I will explain the different methods and components that make up a complete personalization approach such as the one used by Netflix.

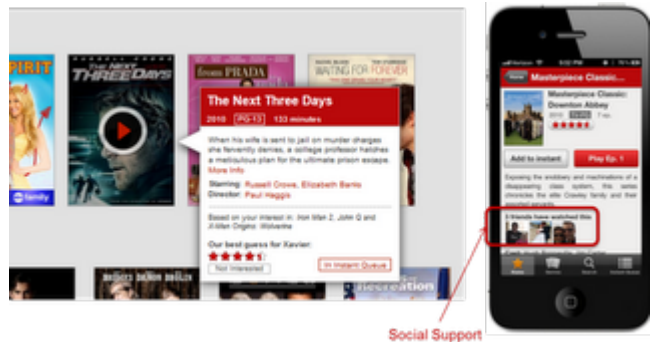
### 3. NETFLIX PERSONALIZATION: BEYOND RATING PREDICTION

Netflix has discovered through the years that there is tremendous value in incorporating recommendations to personalize as much of the experience as possible. This realization pushed us to propose the Netflix Prize described in the previous section. In this section, we will go over the main components of Netflix personalization.

#### 3.1 Everything is a Recommendation

Personalization starts on our homepage in any device. This page consists of groups of videos arranged in horizontal rows. Each row has a title that conveys the intended meaningful connection between the videos in that group. Most of our personalization is based on the way we select rows, how we determine what items to include in them, and in what order to place those items.

Take as a first example the Top 10 row (see Figure 1). This row is our best guess at the ten titles you are most likely to enjoy. Of course, when we say “you”, we really mean everyone in your household. It is important to keep



**Figure 2: Adding explanation and support for recommendations contributes to user satisfaction and requires specific algorithms. Support in Netflix can include your predicted rating, related shows you have watched, or even friends who have interacted with the title.**

in mind that Netflix’ personalization is intended to handle a household that is likely to have different people with different tastes. Even for a single person household we want to appeal to your range of interests and moods. To achieve this, in many parts of our system we are not only optimizing for **accuracy**, but also for **diversity**. Diversification of results is an important area of research for companies that is already attracting interest in the academic community [15, 13, 19]

Another important element in Netflix’ personalization is **awareness**. We want members to be aware of how we are adapting to their tastes. This not only promotes trust in the system, but encourages members to give feedback that will result in better recommendations. A different way of promoting trust with the personalization component is to provide **explanations**[18] as to why we decide to recommend a given movie or show (see Figure 2). We are not recommending it because it suits our business needs, but because it matches the information we have from you: your explicit taste preferences and ratings, your viewing history, or even your friends’ recommendations.

On the topic of friends, we recently released our Facebook connect feature. Knowing about your friends not only gives us another signal to use in our personalization algorithms, but it also allows for different rows that rely mostly on your social circle to generate recommendations. Combining social with collaborative filtering data is also a worthy area of research [20].

Some of the most recognizable personalization in our service is the collection of “genre” rows. These range from familiar high-level categories like “Comedies” and “Dramas” to highly tailored slices such as “Imaginative Time Travel Movies from the 1980s”. Each row represents 3 layers of personalization: the choice of genre itself, the subset of titles selected within that genre, and the ranking of those titles. Rows are generated using a member’s implicit genre preferences — recent plays, ratings, and other interactions —, or explicit feedback provided through our taste preferences survey (see Figure 3). As with other personalization elements, **freshness** and diversity is taken into account when deciding what genres to show from the thousands possible.

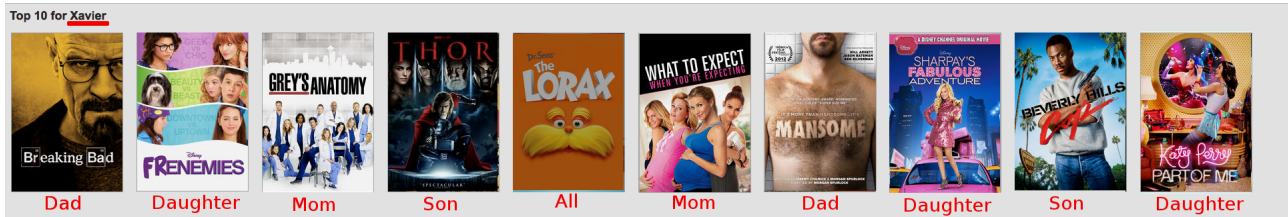


Figure 1: Example of a Netflix Top 10 row. We promote personalization awareness and reflect on the diversity of a household. Note though that personal labels are only the author’s guess since the system is uncertain about the true household composition.

Interesting, the topic of generating “lists” or “sets” of related items to recommend has had little focus in the literature except for some domain-specific research in generating music playlists[14].

**Similarity** is also an important source of personalization. We think of similarity in a very broad sense; it can be between movies or between members, and can be in multiple dimensions such as metadata, ratings, or viewing data. Furthermore, these similarities can be blended and used as features in other models. Similarity is used in multiple contexts, for example in response to generate rows of “ad hoc genres” based on similarity to titles that a member has interacted with recently.

In most of the previous contexts, the goal of the recommender systems is still to present a number of attractive items for a person to choose from. This is usually accomplished by selecting some items and sorting them in the order of expected enjoyment (or *utility*). Since the most common way of presenting recommended items is in some form of list, we need an appropriate **ranking** model that can use a wide variety of information to come up with an optimal sorting of the items. In the next section, we will go into some of the details of how to design such a ranking model.

### 3.2 Ranking

The goal of a ranking system is to find the best possible ordering of a set of items for a user, within a specific context, in real-time. We optimize ranking algorithms to give the highest scores to titles that a member is most likely to play and enjoy.

If you are looking for a ranking function that optimizes consumption, an obvious baseline is item popularity. The reason is clear: on average, a member is most likely to watch what most others are watching. However, popularity is the opposite of personalization: it will produce the same ordering of items for every member. Thus, the goal becomes to find a personalized ranking function that is better than item popularity, so we can better satisfy members with varying tastes. One way to approach this is to use the member’s predicted rating of each item as an adjunct to item popularity. At this point, we are ready to build a ranking prediction model using these two features by, for example, using a very simple scoring function to be a linear combination of popularity and predicted rating. This gives an equation of the form  $score(u, v) = w_1p(v) + w_2r(u, v) + b$ , where  $u$ =user,  $v$ =video item,  $p$ =popularity and  $r$ =predicted rating. This equation defines a two-dimensional space (see Figure ??).

Once we have such a function, we can pass a set of videos through our function and sort them in descending order ac-



Figure 4: Constructing a basic personalized two-dimensional ranking function based on popularity and predicted rating

ording to the score. First, though, we need to determine the weights  $w_1$  and  $w_2$  in our model. We can formulate this as a machine learning problem: select positive and negative examples from your historical data and let a machine learning algorithm learn the weights that optimize our goal. This family of machine learning problems is known as “Learning to Rank” and is central to application scenarios such as search engines or ad targeting. A crucial difference in the case of ranked recommendations is the importance of personalization: we do not expect a global notion of relevance, but rather look for ways of optimizing a personalized model.

As you might guess, the previous two-dimensional model is a very basic baseline. Apart from popularity and rating prediction, we have tried many other features at Netflix. Some have shown no positive effect while others have improved our ranking accuracy tremendously. Figure 5 shows the ranking improvement we have obtained by adding different features and optimizing the machine learning algorithm.

Many traditional supervised classification methods such as logistic regression or tree ensembles can be used for ranking when using this pointwise approach. Recent research, though, has allowed for more sophisticated pairwise [5, 6] or listwise approaches [17, 9].

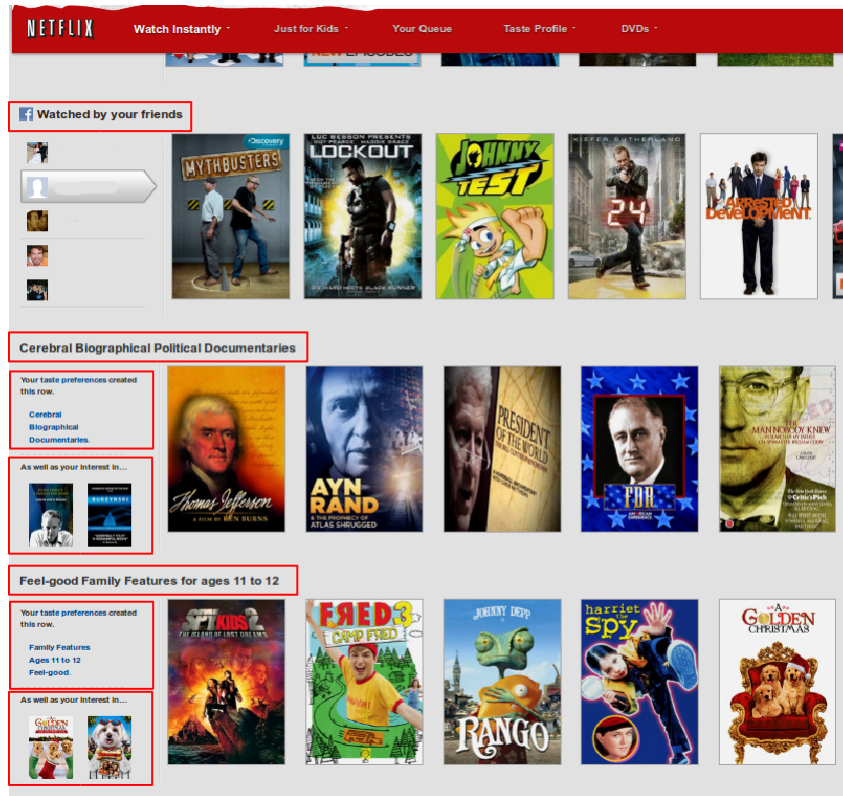


Figure 3: Netflix Genre rows can be generated from implicit, explicit, or hybrid feedback

### 3.3 Data

The previous discussion on the ranking algorithms highlights the importance of both data and models in creating an optimal personalized experience. The availability of high volumes of high quality user data allows for some approaches that would have been unthinkable just a few years back. As an example, here are some of the data sources we can use at Netflix to optimize our recommendations:

- We have several billion item **ratings** from members. And we receive millions of new ratings every day.
- We already mentioned the use of global item **popularity** for ranking. There are many ways to compute popularity such as over various time ranges or grouping members by region or other similarity metrics.
- Our members add millions of items to their **queues** each day. And they directly enter millions of **search terms** each day.
- Each item in our catalog has rich **metadata** such as actors, director, genre, parental rating, or reviews.
- Using presentation and **impression data**, we know what items we have recommended and where we have shown them, and can look at how that decision has affected the user's actions. We can also observe the member's interactions with the recommendations: scrolls, mouse-overs, clicks, or the time spent on a given page.

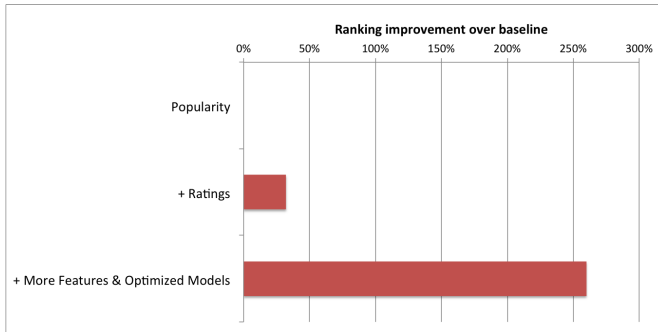


Figure 5: Performance of Netflix ranking system when adding features

- **Social** data has become our latest source of personalization features. Social data may include the social network connections themselves as well as interactions, or activities of connected nodes.
- We can also tap into **external data** such as box office performance or critic reviews to improve our features.
- And that is not all: there are many other features such as **demographics, location, language, or temporal data** that can be used in our predictive models.

### 3.4 Models

So, what about the models? Many different modeling approaches have been used for building personalization engines. One thing we have found at Netflix is that with the great availability of data, both in quantity and types, a thoughtful approach is required to model selection, training, and testing. We use all sorts of machine learning approaches: From unsupervised methods such as **clustering** algorithms to a number of supervised classifiers that have shown optimal results in various contexts. This is an incomplete list of methods you should probably know about if you are working in machine learning for personalization: **Linear regression, Logistic regression, Elastic nets, Singular Value Decomposition, Restricted Boltzmann Machines, Markov Chains, Latent Dirichlet Allocation, Association Rules, Matrix factorization, Gradient Boosted Decision Trees, Random Forests,** and Clustering techniques from the simple **k-means** to graphical approaches such as **Affinity Propagation**.

There is no easy answer to how to choose which model will perform best in a given problem. The simpler your feature space is, the simpler your model can be. But it is easy to get trapped in a situation where a new feature does not show value because the model cannot learn it. Or, the other way around, to conclude that a more powerful model is not useful simply because you don't have the feature space that exploits its benefits [1].

## 4. ARCHITECTURES

In order to build Big Data solutions you not only need to have large volumes of data, and be smart about the algorithms or models you use. It is also very important to design a software architecture that is scalable and reactive enough to allow the system to benefit from the incoming streams of data.

Of course, the simplest thing we can do with data is to store it for later offline processing. There are proven solutions to scale offline computations using, for example, Hadoop. *Offline computation* has almost no limitations on the amount of data and the computational complexity of the algorithms, since it runs in a batch manner with relaxed timing requirements. If we had no requirements related to the latency of the computation, and the reactivity to new user events, we could compute everything in an offline fashion. However, offline results can easily grow stale between updates because the most recent data is not incorporated. On the other end of the spectrum, *online computation* can respond quickly to events and use the most recent data. Online components are subject to an availability and response time Service Level Agreements (SLA) that specifies the maximum latency of the process in responding to requests from client

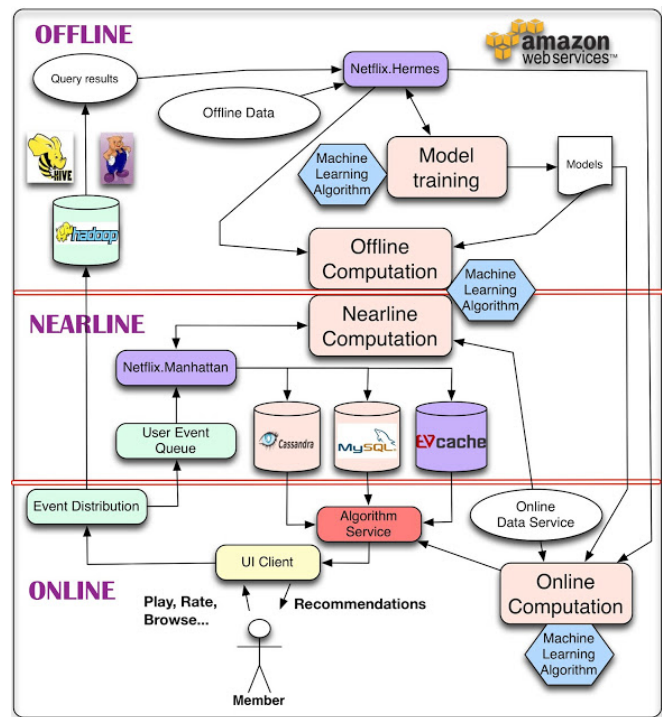


Figure 6: Netflix Architecture for Personalization

applications. This can make it harder to fit complex and computationally costly algorithms in this approach. *Nearline computation* can be seen as a compromise between the two previous modes. In this case, computation is performed exactly like in the online case. However, we remove the requirement to serve results as soon as they are computed and can instead store them, allowing it to be asynchronous. The nearline computation is done in response to user events so that the system can be more responsive between requests.

One of the key issues in a personalization architecture is how to combine and manage online and offline computation in a seamless manner. *Model training* is another form of computation that uses existing data to generate a model that will later be used during the actual computation of results. Another goal of the architecture is to define how the different kinds of events and data need to be handled by the event and data distribution system. A related issue is how to combine the different signals and models that are needed across the offline, nearline, and online regimes. Finally, we also need to figure out how to combine intermediate recommendation results in a way that makes sense for the user. Our solution at Netflix is outlined in Figure 6. See [2] for more details on all these components.

## 5. CONCLUSION

The Netflix Prize abstracted the recommendation problem to a proxy and simplified question of predicting ratings. But it is clear that the Netflix Prize objective, accurate prediction of a movie's rating, is just one of the many components of an effective recommendation system. We also need to take into account factors such as context, popularity, interest, evidence, novelty, diversity, or freshness. Supporting all

the different contexts in which we want to make recommendations requires a range of algorithms and different kinds of data.

Recommender systems in an application domain such as video streaming need to optimize the probability a member chooses an item and enjoys it enough to come back to the service. In order to do so, we should employ all the data that is available: from user ratings and interactions, to content metadata. More data availability enables better results. But in order to get those results, we need to have optimized models, appropriate metrics and system architectures that enable scalable and reactive solutions.

The availability of more and different sources of data has opened up new research avenues. As personalization algorithms keep improving, so does the experience of the users that use these systems. But the recommendation problem is far from solved. There are still many unexplored opportunities and lessons to be learned.

## 6. REFERENCES

- [1] X. Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013.
- [2] X. Amatriain and J. Basilico. System architectures for personalization and recommendation. In the Netflix Techblog: <http://techblog.netflix.com/2013/03/system-architectures-for.html>, March 2013.
- [3] X. Amatriain, J. M. Pujol, and N. Oliver. I Like It... I Like It Not: Evaluating User Ratings Noise in Recommender Systems. In *User Modeling, Adaptation, and Personalization*, volume 5535, chapter 24, pages 247–258. Springer Berlin, 2009.
- [4] R. M. Bell and Y. Koren. Lessons from the Netflix Prize Challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.
- [5] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval*, 13:201–215, June 2010.
- [6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, December 2003.
- [7] S. Funk. Netflix update: Try this at home. <http://sifter.org/~simon/journal/20061211.html>, 2006.
- [8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
- [9] M. Karimzadehgan, W. Li, R. Zhang, and J. Mao. A stochastic learning-to-rank algorithm and its application to contextual advertising. In *Proceedings of the 20th WWW*, 2011.
- [10] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD*, 2008.
- [11] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD*, 2009.
- [12] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, August 2009.
- [13] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA, 2010. ACM.
- [14] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *ISMIR*, volume 5, pages 634–637, 2005.
- [15] K. Raman, P. Shivaswamy, and T. Joachims. Online learning to diversify from implicit feedback. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 705–713, New York, NY, USA, 2012. ACM.
- [16] R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc of ICML '07*, 2007.
- [17] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. CLIMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proc. of the sixth Recsys*, 2012.
- [18] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2011.
- [19] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 109–116, New York, NY, USA, 2011. ACM.
- [20] X. Yang, H. Steck, and Y. Liu. Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1267–1275. ACM, 2012.