Mining Large Streams of User Data for Personalized Recommendations

Xavier Amatriain
Netflix
xamatriain@netflix.com

ABSTRACT

The Netflix Prize put the spotlight on the use of data mining and machine learning methods for predicting user preferences. Many lessons came out of the competition. But since then, Recommender Systems have evolved. This evolution has been driven by the greater availability of different kinds of user data in industry and the interest that the area has drawn among the research community. The goal of this paper is to give an up-to-date overview of the use of data mining approaches for personalization and recommendation. Using Netflix personalization as a motivating use case, I will describe the use of different kinds of data and machine learning techniques.

After introducing the traditional approaches to recommendation, I highlight some of the main lessons learned from the Netflix Prize. I then describe the use of recommendation and personalization techniques at Netflix. Finally, I pinpoint the most promising current research avenues and unsolved problems that deserve attention in this domain.

1. INTRODUCTION

Recommender Systems (RS) 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, and can even be the factor that determines the success or failure of a business. 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. I will describe the use of data, models, and other personalization techniques at Netflix in section 3. I will also discuss whether we should focus on more data or better models in section 4.

Another important issue is how to measure the success of a given personalization technique. Root mean squared error (RMSE) was the offline evaluation metric of choice in the Netflix Prize (see Section 2). But there are many other relevant metrics that, if optimized, would lead to different solutions - think, for example, of ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) or other information retrieval ones such as recall or area under the curve (AUC). Beyond the optimization of a given offline metric, what we are really pursuing is the impact of a method on the business. Is there a way to relate the goodness of an algorithm to more customer-facing metrics such as click-through rate (CTR) or retention? I will describe our approach to innovation called "Consumer Data Science" in section 3.1.

But before we understand the reasons for all these effects, and before we are ready to embrace the open research questions in the area of personalization described in Section 5, we need to understand some of the basic techniques that enable the different approaches. I will briefly describe them in the following paragraphs.

1.1 Approaches to the Recommendation problem

The most common approach to build a Recommender System is to use one of the many available Collaborative Filtering (CF) algorithms [1]. The underlying assumption of these methods is captured by the principle of *like mindedness*: users who are measurably similar in their historical preferences are likely to also share similar tastes in the future. In other words, CF algorithms assume that, in order to recommend content of any kind to users, information can be drawn from what they and other similar users have liked in the past. Historically, the k-Nearest Neighbor (kNN) algorithm was the most favored approach to CF, since it transparently captured this assumption of like-mindedness: it operates by finding, for each user (or item), a number of *similar* users (items) whose profiles can then be used to directly compute recommendations [55].

The main alternative to CF is the so-called content-based approach [46], which identifies similarities between items based on the features inherent in the items themselves. These recommender systems require a way to extract content descriptions and a similarity measure between items. Automatic content description is still only available for some kinds of content, and under some constraints. That is why some systems need to rely on experts to manually input and categorize the content [56]. On the other hand, content-based approaches can deal with some of the shortcomings of CF such as item cold-start - *i.e.* the initialization of new items that the system has no previous user preference data for.

CF and content-based methods can be combined in different ways using hybrid approaches [15]. Hybrid RS can combine

several different methods in a way that one method provides support whenever the other methods are lacking. In practice, most of the advanced recommendation systems used in the industry are based on some sort of hybridation, and are rarely purely CF or content-based.

1.2 Data Mining methods in Recommender Systems

No matter which of the previous approaches is used, a recommender system's engine can be seen as a particular instantiation of a traditional data mining task [4]. A data mining task typically consists of 3 steps, carried out in succession: *Data Preprocessing*, *Data Modeling*, and *Result Analysis*. Traditional machine learning techniques such as dimensionality reduction, classification, or clustering, can be applied to the recommendation problem. In the following paragraphs, I will describe some of the models that, beyond the classical kNN, can be used to build a recommender system.

Although current trends seem to indicate that other matrix factorization techniques are preferred (see Section 2.1), earlier works used **Principal Component Analysis** (PCA) [24] . **Decision Trees** may be used in a content-based approach for a RS. One possibility is to use content features to build a decision tree that models the variables involved in the user preferences [13]. **Bayesian classifiers** have been used to derive a model for content-based RS [23]. **Artificial Neural Networks** (ANN) can be used in a similar way as Bayesian Networks to construct content-based RS's [47]. ANN can also be used to combine (or hybridize) the input from several recommendation modules or data sources [20]. **Support Vector Machines** (SVM) have also shown promising recent results [30].

Clustering approaches such as k-*means* can be used as a pre-processing step to help in neighborhood formation [65]. Finally, **association rules** [3] can also be used[38].

2. 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.

2.1 Lessons from the Prize

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) [35] [1] and Restricted Boltzmann Machines (RBM) [54]. 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.

The standard matrix factorization decomposition provides user factor vectors $U_u \in R^f$ and item-factors vector $V_v \in R^f$. In order to predict a rating, we first estimate a baseline $b_{uv} = \mu + b_u + b_v$ as the user and item deviation from average. The prediction can then be obtained by adding the product of user and item factors to the baseline as $r'_{uv} = b_{uv} + U_u^T U_v$.

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 SVD using gradient descent [22]. 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++ [33]. This asymmetric variation enables adding both implicit and explicit feedback, and removes the need for parameterizing the users.

The second model that proved successful in the Netflix Prize was the Restricted Boltzmann Machine (RBM). RBM's can be understood as the fourth generation of Artificial Neural Networks - the first being the Perceptron popularized in the 60s; the second being the backpropagation algorithm in the 80s; and the third being Belief Networks (BNs) from the 90s. RBMs are BNs that restrict the connectivity to make learning easier. RBMs can be stacked to form Deep Belief Nets (DBN). For the Netflix Prize, Salakhutditnov *et al.* proposed an RBM structure with binary hidden units and softmax visible units with 5 biases only for the movies the user rated [54].

Many other learnings came out of the Prize. For example, the matrix factorization methods mentioned above were combined with the traditional neighborhood approaches [33]. Also, early in the prize, it became clear that it was important to take into account temporal dynamics in the user feedback [34].

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.*[27] 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 [6], 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 [11]. The way that the final

[1]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.

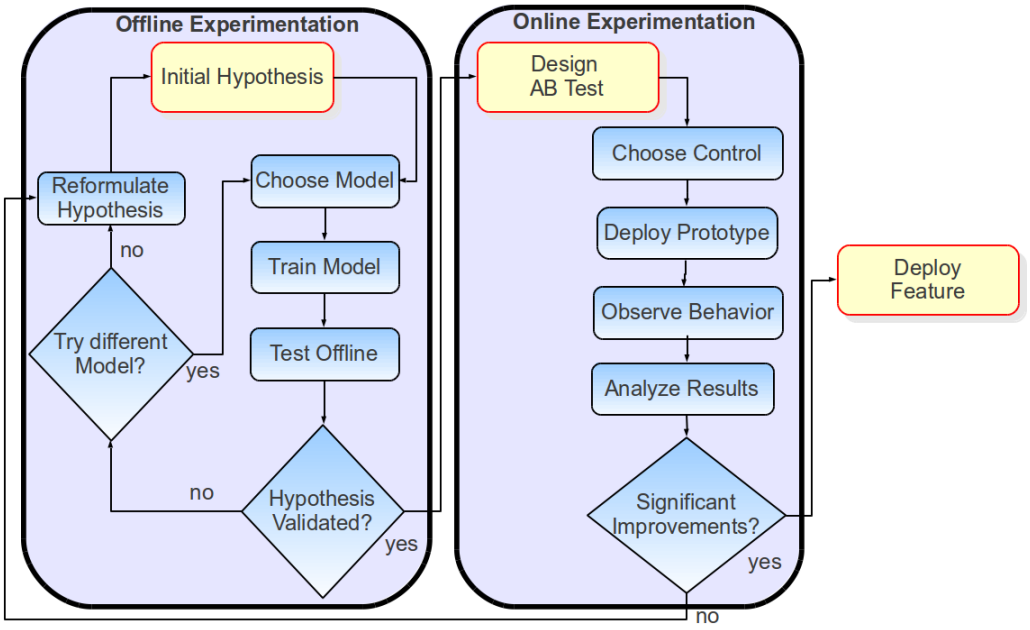

Figure 1: Following an iterative and data-driven offline-online process for innovating in personalization

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. But first let us take a look at how we manage innovation in this space.

3.1 Consumer Data Science

The abundance of source data, measurements and associated experiments allow Netflix not only to improve our personalization algorithms but also to operate as a data-driven organization. We have embedded this approach into our culture since the company was founded, and we have come to call it Consumer (Data) Science. Broadly speaking, the main goal of our Consumer Science approach is to innovate for members effectively. We strive for an innovation that allows us to evaluate ideas rapidly, inexpensively, and objectively. And once we test something, we want to understand why it failed or succeeded. This lets us focus on the central goal of improving our service for our members.

So, how does this work in practice? It is a slight variation on the traditional scientific process called A/B testing (or bucket testing):

1. **Start with a hypothesis**: Algorithm/feature/design X will increase member engagement with our service and ultimately member retention.
2. **Design a test**: Develop a solution or prototype. Think about issues such as dependent & independent variables, control, and significance.
3. **Execute the test**: Assign users to the different buckets and let them respond to the different experiences.
4. **Let data speak for itself**: Analyze significant changes on primary metrics and try to explain them through variations in the secondary metrics.

When we execute A/B tests, we track many different metrics. But we ultimately trust member engagement (e.g. viewing hours) and retention. Tests usually have thousands of members and anywhere from 2 to 20 cells exploring variations of a base idea. We typically have scores of A/B tests running in parallel. A/B tests let us try radical ideas or test many approaches at the same time, but the key advantage is that they allow our decisions to be data-driven.
An interesting follow-up question that we have faced is how to integrate our machine learning approaches into this data-driven A/B test culture at Netflix. We have done this with an offline-online testing process that tries to combine the best of both worlds (see Figure 1). The offline testing cycle is a step where we test and optimize our algorithms prior to performing online A/B testing. To measure model performance offline we track multiple metrics: from ranking measures such as normalized discounted cumulative gain, to classification metrics such as precision, and recall. We also

Figure 2: 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.

use the famous RMSE from the Netflix Prize or other more exotic metrics to track different aspects like diversity. We keep track of how well those metrics correlate to measurable online gains in our A/B tests. However, since the mapping is not perfect, offline performance is used only as an indication to make informed decisions on follow up tests.

Once offline testing has validated a hypothesis, we are ready to design and launch the A/B test that will prove the new feature valid from a member perspective. If it does, we will be ready to roll out in our continuous pursuit of the better product for our members. That is in fact how we came about to having the personalization experience I will describe in the next section.

3.2 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 2). 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 in mind that Netflix personalization is intended to handle a household that is likely to have different people with different tastes. That is why when you see your Top 10, you are likely to discover items for dad, mom, the kids, or the whole family. 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**.

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** as to why we decide to recommend a given movie or show (see Figure 3). 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.

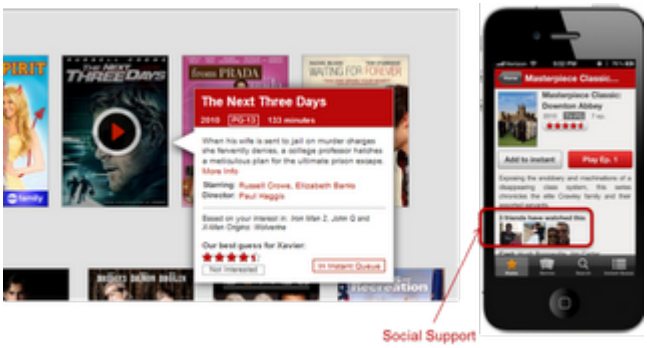

Figure 3: Adding explanation and support for recommendations contributes to user satisfaction and requires specific algorithms. Support in Netflix can include your predicted rating, related shows you have watched, or even friends who have interacted with the title.

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 4) . As with other personalization elements, **freshness** and diversity is taken into account when deciding what genres to show from the thousands possible.

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 "adhoc 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

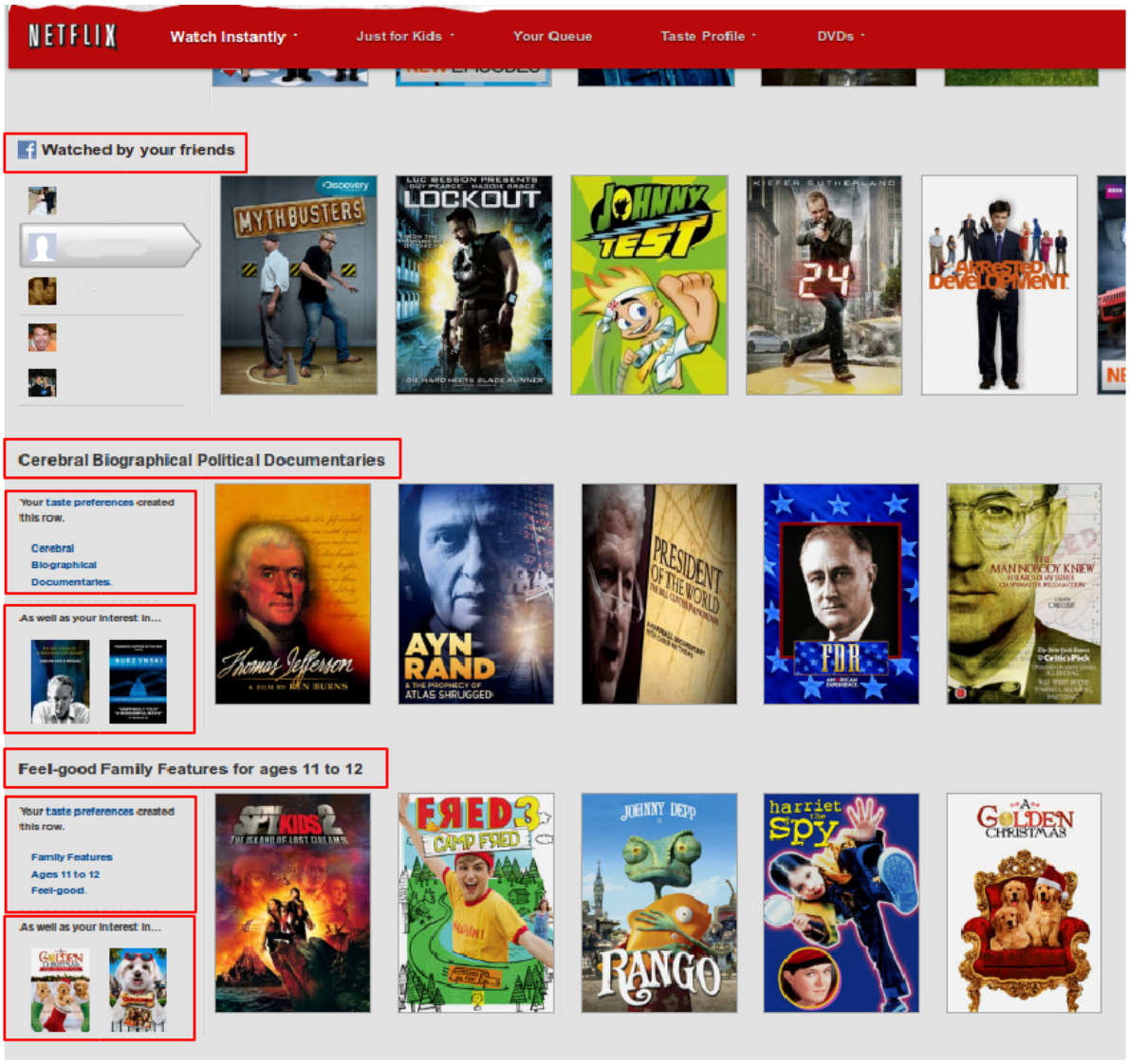

Figure 4: Netflix Genre rows can be generated from implicit, explicit, or hybrid feedback

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.

3.3 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.

Recall that our goal is to recommend the titles that each member is most likely to play and enjoy. One obvious way to approach this is to use the member's predicted rating of each item as an adjunct to item popularity. Using predicted ratings on their own as a ranking function can lead to items that are too niche or unfamiliar, and can exclude items that the member would want to watch even though they may not rate them highly. To compensate for this, rather than using either popularity or predicted rating on their own, we would like to produce rankings that balance both of these aspects. At this point, we are ready to build a ranking prediction model using these two features.

Let us start with a very simple scoring approach by choosing our ranking function to be a linear combination of popularity and predicted rating. This gives an equation of the form $score(u, v) = w_1 p(v) + w_2 r(u, v) + b$, where u=user, v=video item, p=popularity and r=predicted rating. This equation defines a two-dimensional space (see Figure 5).

Once we have such a function, we can pass a set of videos through our function and sort them in descending order according to the score. First, though, we need to determine the weights w_1 and w_2 in our model (the bias b is constant and thus ends up not affecting the final ordering). 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 6 shows the ranking improvement we have obtained by adding different features

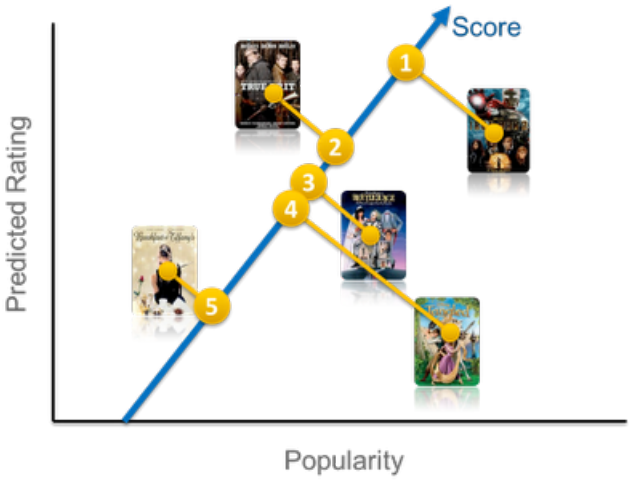

Figure 5: Constructing a basic personalized two-dimensional ranking function based on popularity and predicted rating

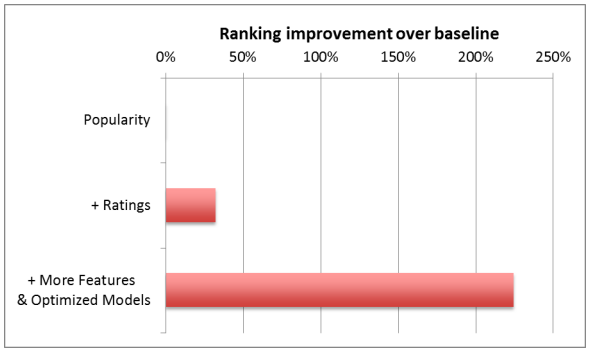

Figure 6: Performance of Netflix ranking system when adding features

and optimizing the machine learning algorithm.

Many supervised classification methods can be used for ranking. In section 5.2, we will explore some of the latest approaches to the learning to rank problem.

3.4 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.
- **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.

3.5 Models

So, what about the models? As we described in Section 1.2, 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.

4. DISCUSSION: MORE DATA OR BETTER MODELS?

The previous discussion on models vs. data has recently become a favorite - and controversial - topic. The improvements enabled thanks to the availability of large volumes of data together with a certain Big Data "hype" have driven many people to conclude that it is "all about the data". But in most cases, data by itself does not help in making our predictive models better.

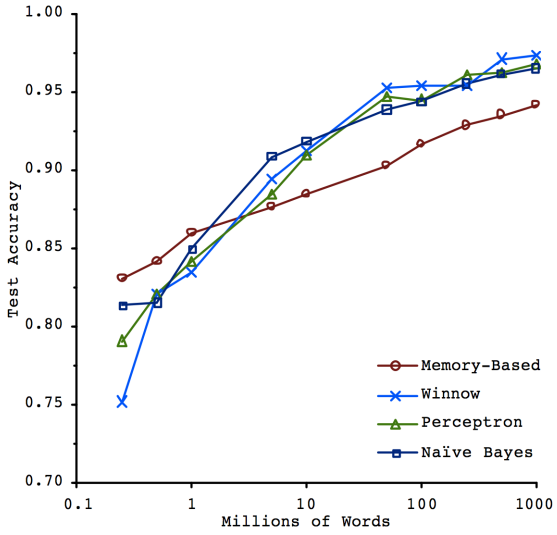

Figure 7: In some cases, accuracy does not depend so much on the model used, but on the amount of data used to train the model. (From "Scaling to Very Very Large Corpora for Natural Language Disambiguation" [Banko And Brill, 2001])

Probably one of the most famous quotes defending the power of data is that of Peter Norvig claiming that "We don't have better algorithms. We just have more data.". He is even misquoted as saying that "All models are wrong, and you don't need them anyway" (You can read Norvig's rebuttal and clarifications in his own webpage [2]). Norvig did co-author a paper entitled "The Unreasonable Effectiveness of Data" [26] in which the authors discuss the power of data over models. The effect that the authors were referring to had already been described years before in a now famous paper by Banko and Brill [9] where the authors included the plot in Figure 7. Both Norvig, and Banko and Brill are of course right... in a context. The problem is that they are now and again misquoted in contexts that are completely different from the original ones.

In order to understand why, we need to clarify the difference between models with high **variance** or high **bias**. The basic idea is that there are two possible (and almost opposite) reasons why a model might not perform well. In the first case, we might have a model that is too complicated for the amount of data we have. This situation, known as high variance, leads to model overfitting. We know that we are facing a high variance issue when the training error is much lower than the test error. High variance problems can be addressed by reducing the number of features, and by increasing the number of data points. Both Banko & Brill and Norvig were dealing with high variance models since they were working on language models in which roughly every word in the vocabulary makes a feature. These are models with many features as compared to the training examples. Therefore, they are likely to overfit. And yes, in this case adding more examples will help. In the opposite case, we might have a model that is too simple to explain the data

[2] http://norvig.com/fact-check.html

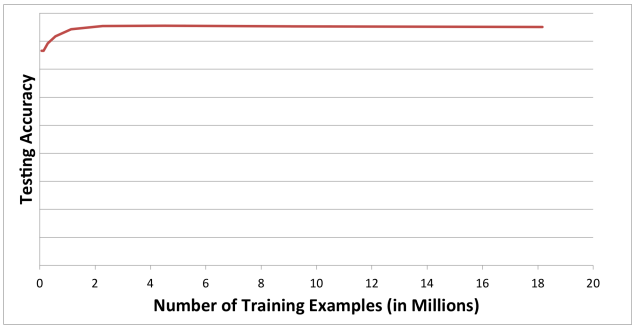

Figure 8: Adding more data to high bias models will not help. This plot illustrates a real case from a production system at Netflix

we have. In that case, known as high bias, adding more data will not help. See figure 8 illustrating the performance of a real production system at Netflix as we add more training examples.

So, no, more data does not always help. As we have just seen, there can be many cases in which adding more examples to our training set will not improve the model performance.

On the other hand, high bias models will not benefit from more training examples, but they might very well benefit from more features. Or will they? Well, again, it depends. Let's take the Netflix Prize, for example. Pretty early on, there was a blog post commenting on the use of extra features to solve the problem[3]. The post explains how a team of students got an improvement on the prediction accuracy by adding content features from IMDB. In retrospect, it is easy to criticize the post for making a gross over-generalization from a single data point. Many teams showed later that adding content features from IMDB or the like to an optimized algorithm had little to no improvement. Some of the members of the Gravity team, one of the top contenders for the Prize, published a detailed paper in which they showed how those content-based features would add no improvement to the highly optimized collaborative filtering matrix factorization approach [48]. Again: More data, even in the form of different features, does not always help.

So, is the Big Data revolution only hype? No way. Having more data, both in terms of more examples or more features, is a blessing. The availability of data enables more and better insights and applications. More data indeed enables better approaches. More than that: it requires better approaches! In other words, data is important. But data without a sound approach becomes noise.

5. RESEARCH DIRECTIONS

As I mentioned before, even though there were a lot of research advances in the context of the Netflix Prize, the prize was a simplification. In section 3, I illustrated the broader scope of the recommendation problem by presenting Netflix' comprehensive approach. In this section, I will describe some of the latest advances in Recommender Systems by highlighting some of the most promising research directions.

[3]http://anand.typepad.com/datawocky/2008/03/more-data-usual.html

Most of these directions are enabled thanks to the availability of larger amounts of different data such as implicit user feedback, contextual information, or social network interaction data.

5.1 Beyond explicit ratings

Explicit ratings are not the only or even the best kind of feedback we can get from our users. As already mentioned, explicit feedback is noisy. Another issue with ratings is that they do not represent a linear but rather an ordinal scale. Most traditional methods wrongly interpret ratings as being linear, for example by computing averages. This issue, however, has been addressed by some recent methods such as OrdRec [70] deal with this issue.

In any case, in most real-world situations, implicit and binary feedback is much more readily available and requires no extra effort on the user side. For instance, in a web page you will have users visiting a URL, or clicking on an ad as a positive feedback. In a music service, a user will decide to listen to a song. Or in a movie service, like Netflix, you will have users deciding to watch a title as an indication that the user liked the movie. That is why, besides trying to address some of the issues with explicit ratings, there have been many recent approaches that focus on the use of the more reliable and readily available implicit feedback . Bayesian Personalized Ranking (BPR) [51], for example, uses implicit feedback to compute a personalized ranking.

Implicit and explicit feedback can be combined in different ways [44]. Even the SVD++ approach explained in Section 2.1 and used during the prize can combine explicit and implicit feedback. Another way is to use logistic ordinal regression [45]. Matchbox, a Bayesian approach [62], also offers a framework to integrate different kinds of feedback such as ordinal ratings or implicit like/don't like preferences.

5.2 Personalized Learning to Rank

The traditional pointwise approach to learning to rank described in Section 3.3 treats ranking as a simple binary classification problem where the only input are positive and negative examples. Typical models used in this context include Logistic Regression, Support Vector Machines, or Gradient Boosted Decision Trees.

There is a growing research effort in finding better approaches to ranking. The pairwise approach to ranking, for instance, optimizes a loss function defined on pairwise preferences from the user. The goal is to minimize the number of inversions in the resulting ranking. Once we have reformulated the problem this way, we can transform it back into the previous binary classification problem. Examples of such an approach are RankSVM [17], RankBoost [21], or RankNet [14].

We can also try to directly optimize the ranking of the whole list by using a listwise approach. RankCosine [66], for example, uses similarity between the ranking list and the ground truth as a loss function. ListNet [16] uses KL-divergence as loss function by defining a probability distribution. RankALS [63] is a recent approach that defines an objective function that directly includes the ranking optimization and then uses Alternating Least Squares (ALS) for optimizing.

Whatever ranking approach we use, we need to use rank-specific information retrieval metrics to measure the performance of the model. Some of those metrics include Mean

Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR), or Fraction of Concordant Pairs (FCP). What we would ideally like to do is to directly optimize those same metrics. However, it is hard to optimize machine-learned models directly on these measures since they are not differentiable and standard methods such as gradient descent or ALS cannot be directly applied.

In order to optimize those metrics, some methods find a smoothed version of the objective function to run Gradient Descent. CLiMF optimizes MRR [60], and TFMAP [59], optimizes MAP in a similar way. AdaRank [68] uses boosting to optimize NDCG. Another method to optimize NDCG is NDCG-Boost [64], which optimizes expectation of NDCG over all possible permutations. SVM-MAP [69] relaxes the MAP metric by adding it to the SVM constraints. It is even possible to directly optimize the non-diferentiable IR metrics by using techniques such as Genetic Programming, Simulated Annealing [32], or even Particle Swarming [18].

5.3 Context-aware recommendations

Most of the work on recommender systems has traditionally focused on the two-dimensional user/item problem. But we know that in practice many other dimensions might be affecting the user's preference. All of those other dimensions (e.g. location, or time) are referred to as context. Using contextual variables represents having to deal with more data, and a higher dimensionality problem. However, this might prove effective for businesses [25].

Adomavicius and Tuzhilin [2] do a thorough review of approaches to contextual recommendations and categorize context-aware recommender systems (CARS) into three types: contextual pre-filtering, where context drives data selection; contextual post-filtering, where context is used to filter recommendations once they have been computed using a traditional approach; and contextual modeling, where context is integrated directly into the model. An example of contextual pre-filtering is the so-called user micro-profile, in which a single user is represented by a hierarchy of possibly overlapping contextual profiles [8]. Post-filtering methods can use traditional approaches and then apply filtering or weighting. In their experimental evaluation, Panniello et al. [43] found that the choice of a pre-filtering or post-filtering strategy depends on the particular method and sometimes a simple post-filter can outperform an elaborate pre-filtering approach.

Although some standard approaches to recommendation could theoretically accept more dimensions, the only models to report results in this category are Oku et al.'s Context-aware Support Vector Machines (SVM) [42]. Xiong et al. present a Bayesian Probabilistic Tensor Factorization model to capture the temporal evolution of online shopping preferences [67]. The authors show in their experiments that results using this third dimension in the form of a tensor does improve accuracy when compared to the non-temporal case. Multiverse is another multidimensional tensor factorization approach to contextual recommendations that has proved effective in different situations [31].

Factorization Machines [50] is a novel general-purpose regression model that models interactions between pairs of variables and the target by using factors. Factorization Machines have proved to be useful in different tasks and domains [52]. In particular, they can be efficiently used to model the interaction with contextual variables [53]. Another novel approach to contextual recommendations worth mentioning is the one based on the use of Sparse Linear Method (SLIM) [39].

5.4 Unbalanced class problems and presentation effects

In the traditional formulation of the "Recommender Problem", we have pairs of items and users and user feedback values for very few of those dyads. The problem is formulated as the finding of a utility function or model to estimate the missing values. But in cases where we have implicit feedback, the recommendation problem becomes the prediction of the probability a user will interact with a given item. There is a big shortcoming in using the standard recommendation formulation in such a setting: we don't have negative feedback. All the data we have is either positive or missing. And the missing data includes both items that the user explicitly chose to ignore because they were not appealing and items that would have been perfect recommendations but were never presented to the user [61].

A way that this unbalanced class problem has been addressed is to convert unlabeled examples into both a positive and a negative example, each with a different weight related to the probability that a random exemplar is positive or negative [19]. Another solution is to binarize the implicit feedback values: any feedback value greater than zero means positive preference, while any value equal to zero is converted to no preference [28]. A greater value in the implicit feedback value is used to measure the "confidence" in the fact the user liked the item.

In many practical situations, though, we have more information than the simple binary implicit feedback from the user. In particular, we might be able to know whether items not selected by the user were actually shown. This adds very valuable information, but slightly complicates the formulation of our recommendation problem. We now have three different kinds of values for items: positive, presented but not chosen, and not presented. This issue has been recently addressed by the so-called Collaborative Competitive Filtering (CCF) approach [71]. The goal of CCF is to model not only the collaboration between similar users and items, but also the competition of items for user attention.

Another important issue related to how items are presented is the so-called position bias: An item that is presented in the first position of a list has many more possibilities to be chosen than one that is further down [49].

5.5 Social Recommendations

One of the factors that has contributed the most to the recent availability of large streams of data is the explosion of social networks. Recommender systems have also jumped onto this new source of data [37]. Most of the initial approaches to social recommendation[4] relied on the so-called *trust-based model* in which the trust (or influence) of others is transmitted through the social network connections [41; 7]. It is still unclear whether users prefer recommendations from friends to those coming from other users. In a recent study [12], the authors found that the selection of

[4]It is important to note that the term "social recommendation" was originally used to describe collaborative filtering approaches [10; 58]

users where the recommendation came from did not make much difference, except if the recipients of the recommendation were made aware of it. In any case, it seems clear that at the very least social trust can be used as a way to generate explanations and support that have a positive impact on the user.

There are other uses of social information. For instance, social network data can be an efficient way to deal with user or item cold-start. Social information can, for instance, be used to select the most informative and relevant users or items [36]. And speaking of selecting users, some recent methods propose using social information to select experts [57] in a similar way as they can also be selected in collaborative filtering settings [5].

Social-based recommendations can also be combined with the more traditional content-based or collaborative filtering approaches [29]. As a matter of fact, social network information can be efficiently included in a pure collaborative filtering setting by, for example, including it in the matrix factorization objective function [40; 72].

6. 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 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 approaches, appropriate metrics and rapid experimentation.

This 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. Our team of researchers and engineers at Netflix do so every day. Make sure to visit our jobs page[5] if you are interested in joining us on this pursuit.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] G. Adomavicius and A. Tuzhilin. *Recommender Systems Handbook*, chapter Context-aware recommender systems, pages 217–253. Springer, 2011.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of 20th VLDB Conf.*, 1994.

[4] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol. Data Mining Methods for Recommender Systems. In *Recommender Systems Handbook*, pages 39–71. Springer US, 2011.

[5] X. Amatriain, N. Lathia, J. M. Pujol, H. Kwak, and N. Oliver. The wisdom of the few: a collaborative filtering approach based on expert opinions from the web. In *Proc. of 32nd ACM SIGIR*, 2009.

[6] 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.

[7] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: an axiomatic approach. In *Proc. of the 17th WWW*, 2008.

[8] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on Context-Aware Recommender Systems (CARS 2009)*, 2009.

[9] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proc. of ACL '01*, pages 26–33, 2001.

[10] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proc. of AAAI '98*, 1998.

[11] R. M. Bell and Y. Koren. Lessons from the Netflix Prize Challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007.

[12] S. Bourke, K. McCarthy, and B. Smyth. Power to the people: exploring neighbourhood formations in social recommender system. In *Proc. of Recsys '11*, pages 337–340, 2011.

[13] A Bouza, G Reif, A Bernstein, and H. Gall. Semtree: ontology-based decision tree algorithm for recommender systems. In *International Semantic Web Conference*, 2008.

[14] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd ICML*, pages 89–96, 2005.

[15] R. Burke. The adaptive web. chapter Hybrid Web Recommender Systems, pages 377–408. 2007.

[16] Z. Cao and T. Liu. Learning to rank: From pairwise approach to listwise approach. In *In Proceedings of the 24th ICML*, 2007.

[5] http://jobs.netflix.com

[17] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval*, 13:201–215, June 2010.

[18] E. Diaz-Aviles, M. Georgescu, and W. Nejdl. Swarming to rank for recommender systems. In *Proc. of Recsys '12*, 2012.

[19] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proc. of the 14th ACM SIGKDD*, 2008.

[20] S. Hsu et al. AIMED- A Personalized TV Recommendation System. In *Interactive TV: a Shared Experience*, 2007.

[21] 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.

[22] S. Funk. Netflix update: Try this at home. http://sifter.org/ simon/journal/20061211.html, 2006.

[23] R. Ghani and A. Fano. Building recommender systems using a knowledge base of product semantics. In *In 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems*, 2002.

[24] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Journal Information Retrieval*, 4(2):133–151, July 2001.

[25] M. Gorgoglione, U. Panniello, and A. Tuzhilin. The effect of context-aware recommendations on customer purchasing behavior and trust. In *Proc. of Recsys '11*, pages 85–92, 2011.

[26] A. Halevy, P. Norvig, and F. Pereira. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12, March 2009.

[27] 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.

[28] Yifan Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proc. of the 2008 Eighth ICDM*, pages 263–272, 2008.

[29] M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proc. of KDD '09*, 2009.

[30] H. Kang and S. Yoo. SVM and Collaborative Filtering-Based Prediction of User Preference for Digital Fashion Recommendation Systems. *IEICE Transactions on Inf & Syst*, 2007.

[31] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proc. of the fourth ACM Recsys*, 2010.

[32] 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.

[33] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD*, 2008.

[34] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD*, 2009.

[35] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, August 2009.

[36] N. N. Liu, X. Meng, C. Liu, and Q. Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *Proc. of RecSys '11*, 2011.

[37] H. Ma, H. Yang, M.R. Lyu, and I. King. Sorec: social recommendation using probabilistic matrix factorization. In *Proc. of the 17th Conf. on Information and knowledge management*, 2008.

[38] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *WIDM '01*.

[39] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proc. of the 21st WWW*, 2012.

[40] J. Noel, S. Sanner, K. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna. New objective functions for social collaborative filtering. In *Proc. of WWW '12*, pages 859–868, 2012.

[41] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proc. of IUI '05*, 2005.

[42] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware SVM for context-dependent information recommendation. In *Proc. of the 7th Conference on Mobile Data Management*, 2006.

[43] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of RecSys '09*, 2009.

[44] D. Parra and X. Amatriain. Walk the Talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. In *User Modeling, Adaption and Personalization*, volume 6787, chapter 22, pages 255–268. Springer, 2011.

[45] D. Parra, A. Karatzoglou, X. Amatriain, and I. Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. In *Proc. of the 2011 CARS Workshop*.

[46] M. Pazzani and D. Billsus. Content-based recommendation systems. In *The Adaptive Web*, volume 4321. 2007.

[47] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[48] I. Pilászy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *Proc. of RecSys '09*, 2009.

[49] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proc. of the 17th CIKM*, pages 43–52, 2008.

[50] S. Rendle. Factorization Machines. In *Proc. of 2010 IEEE ICDM*, 2010.

[51] S. Rendle, C. Freudenthaler, Z. Gantner, and L. S. Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th UAI*, 2009.

[52] S. Rendle, C. Freudenthaler, and L. S. Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *Proc. of the 19th WWW*, New York, NY, USA, 2010.

[53] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proc. of the 34th ACM SIGIR*, 2011.

[54] R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proc of ICML '07*, 2007.

[55] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *In Proc. of the 10th International WWW Conference*, 2000.

[56] Science. Rockin' to the Music Genome. *Science*, 311(5765):1223d–, 2006.

[57] X. Sha, D. Quercia, P. Michiardi, and M. Dell'Amico. Spotting trends: the wisdom of the few. In *Proc. of the Recsys '12*, 2012.

[58] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *Proc. of SIGCHI '95*, 1995.

[59] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. TFMAP: optimizing MAP for top-n context-aware recommendation. In *Proc. of the 35th SIGIR*, 2012.

[60] 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.

[61] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proc. of the 16th ACM SIGKDD*, 2010.

[62] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proc. of the 18th WWW*, 2009.

[63] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *Proc. of Recsys '12*, 2012.

[64] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to Rank by Optimizing NDCG Measure. In *Proc. of SIGIR '00*, 2000.

[65] Gui-Rong X., Chenxi L., Qiang Y., WenSi X., Hua-Jun Z., Yong Y., and Zheng C. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of the 2005 SIGIR*, 2005.

[66] F. Xia, T. Y. Liu, W. Wang, J.and Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. of the 25th ICML*, New York, NY, USA, 2008.

[67] L. Xiong, X. Chen, T. Huang, and J. G. Carbonell J. Schneider. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining*, 2010.

[68] J. Xu and H. Li. AdaRank: a boosting algorithm for information retrieval. In *Proc. of SIGIR '07*, 2007.

[69] J. Xu, T. Y. Liu, M. Lu, H. Li, and W. Y. Ma. Directly optimizing evaluation measures in learning to rank. In *Proc. of SIGIR '08*, 2008.

[70] Koren Y and J. Sill. OrdRec: an ordinal model for predicting personalized item rating distributions. In *RecSys '11*, pages 117–124, 2011.

[71] S.H. Yang, B. Long, A.J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proc. of the 34th ACM SIGIR*, 2011.

[72] X. Yang, H. Steck, Y. Guo, and Y. Liu. On top-k recommendation using social networks. In *Proc. of RecSys '12*, 2012.