Noname manuscript No.
(will be inserted by the editor)

Temporal influence over the Last.fm social network

Róbert Pálovics[1,2] **András A. Benczúr**[1,3]
[1]**Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)**
[2]**Technical University Budapest**
[3]**Eötvös University Budapest**
{rpalovics,benczur}@ilab.sztaki.hu

the date of receipt and acceptance should be inserted later

Abstract In a previous result, we showed that the influence of social contacts spreads information about new artists through the Last.fm social network. We successfully decomposed influence from effects of trends, global popularity, and homophily or shared environment of friends. In this paper we present our new experiments that use a mathematically sound formula for defining and measuring the influence in the network. We provide new baseline and influence models and evaluation measures, both batch and online, for real time recommendations with very strong temporal aspects. Our experiments are carried over the two-year "scrobble" history of 70,000 Last.fm users. In our results, we formally define and distill the effect of social influence. In addition, we provide new models and evaluation measures for real time recommendations with very strong temporal aspects.

1 Introduction

Last.fm became a relevant online service in music based social networking. For registered users, it collects "scrobbles", which is a word by Last.fm meaning that when you listen to a song, the name of the song is added to your music profile. Most user profiles are public, and each user of Last.fm may have friends inside the Last.fm social network.

In this paper we exploit the time series of information gathered by the Last.fm service. Our goal is to investigate how members of the social network may influence their friends' taste. For this purpose, we use data from users with public profile who allow

Research supported in part by the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956) and by the grant OTKA NK 105645.
Support from the "Momentum - Big Data" grant of the Hungarian Academy of Sciences.
The work of Robert Palovics reported in this paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME" project. This project is supported by the grant TAMOP - 4.2.2.B-10/1–2010-0009. Work conducted at the Eötvös University, Budapest was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013).

Address(es) of author(s) should be given

others to view their detailed timeline of listening history. Last.fm's service is unique since in the timeline and friendship information, we may catch immediate effects by matching the history of friends in time. In our data, we observe stronger temporal correlation of friends compared to pairs of random users. Based on modeling the difference of coupled events between friends vs. all users, we formally define a quantity to distinguish influence from coincidental pairs of users listening to the same music.

In our previous work [27] we showed that social contacts influence one another by showing that the observed similarity in taste and behavior is not only due to homophily: in a carefully designed experiment we subtracted external effects that may result in friends listening to similar music.

In this paper we concentrate on the timely aspects of the recommendation and the statistical foundations of the notion of the influence. We give two methods that take potential influence between friends into account.

The first lightweight method is a modified version of our previous result [27]. We recommend new artists to a user closely after a friend listened to the same artist. Since most users only have a few friends, the prediction can be very efficiently computed even in real time. Our model in this paper is based on a probabilistic notion of the influence conditional probabilities, unlike the heuristic approach of [27].

The second method is a highly time sensitive latent factor model. Compared to a standard collaborative filtering method, we process events only once and in the order they have appeared. This online learning method is based on stochastic gradient descent with high learning rate so that recent events have high contribution to the factor weights. The online factor model already incorporates not just popularity by using a high learning rate, but also part of friends' influence. Immediately after a user listens to an artist, the corresponding factor weights are adjusted by a high learning rate. If a friend has similar factor weights e.g. by homophily, the same artist will have high recommendation score after the learning step. The online factor model hence involves a latent variant of an influence recommender.

Our best recommender method is the combination of the online factor model with the lightweight recommender based on friends' past items. This latter influence based recommender combines very well with other methods and improves an additional near 1% even over the online factor model that already incorporates part of the influence effect. Over batch recommenders, we obtain a 4% of increase in quality, a strong result in view of the three-year Netflix Prize competition [5] to improve recommender quality by 10%. Note that we only give two methods that result in a stable strong improvement over the baselines that include batch matrix factorization, temporal popularity [27] and social regularization [22].

As part of our new results, we introduce quality measures for time aware recommender evaluation. As influence from friends has short, few hours effect only, we retrain part of our models after each event and hence potentially we give completely new top list of items for each event in the testing period. We show that discounted cumulative gain (DCG) computed individually for each event and averaged in time is an appropriate measure for real time recommender evaluation.

Throughout the paper, we use an anonymized data set of two years of artist scrobble timeline obtained from Last.fm. We selected a representative, well-connected, yet anonymous random sample of users by the following rule:

- User location is stated in UK;
- Age between 14 and 50, inclusive;

– Profile displays scrobbles publicly (privacy constraint);
– Daily average activity between 5 and 500.
– User has at least 10 friends that meet the first four conditions.

In our experiments we use 71,000 users with 285,241 friendship edges. The time series contain 979,391,001 scrobbles from 2,073,395 artists and were collected between 01 January 2010 and 31 December 2011. Note that one user can scrobble an artist at different times. The number of unique user-artist scrobbles is 57,274,158. Detailed statistics on the data set can be found in [27].

The rest of this paper is organized as follows. First we explore for measurable signs of influence by friends in Section 2. Our lightweight influence recommender is defined in Section 3 and our online matrix factorization method in Section 4. We define new online evaluation metrics in Section 5, and give the baseline algorithms in Section 6. Finally we show our measurements for improved recommendation quality in Section 7.

1.1 Related results

The Netflix Prize competition [5] has recently generated increased interest in recommender algorithms in the research community and put recommender algorithms under a systematic thorough evaluation on standard data [4]. The final best results combined a very large number of methods whose reproduction is out of the scope of this paper. As one of our baselines we selected a successful matrix factorization recommender described by Simon Funk in [14] that is based on an approach reminiscent of gradient boosting [13].

Bonchi [6] summarizes the data mining aspects of research on social influence. He concludes that "another extremely important factor is the temporal dimension: nevertheless the role of time in viral marketing is still largely (and surprisingly) unexplored", an aspect that is key in our result.

Closest to our results are the applications of network influence in collaborative filtering under the term of "social regularization" [22, 26, 32, 33]. These results add smoothing to that make friends' model similar. We use social regularization as one baseline model in our experiments. In other results, only ratings and no social contacts are given [11], or in [15], both friendship and view information was present over Flickr, but the main goal was to measure the strength of the influence and no measurements were designed to separate influence from other effects.

Since our goal is to recommend different artists at different times, our evaluation must be based on the quality of the top list produced by the recommender. This so-called top-K recommender task is known to be hard [10]. For a recent result on evaluating top-K recommenders is found in [9].

Music recommendation is considered in several results orthogonal to our methods that will likely combine well. Mood data set is created in [16]. Similarity search based on audio is given in [18]. Tag based music recommenders [12, 30, and many more], a few of them based on Last.fm tags, use annotation and fall into the class of content based methods as opposed to collaborative filtering considered in our paper. Best starting point for tag recommendation in general are the papers [17, 23, 24]. Note that the Netflix Prize competition put a strong vote towards the second class of methods [28].

As a social media service, Twitter is widely investigated for influence and spread of information. Twitter influence as followers has properties very different from usual

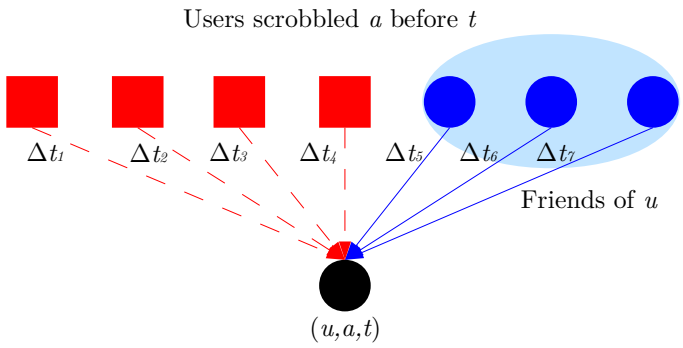

Fig. 1: Potential influence on u by some other user to scrobble (u, a, t).

social networks [20]: compared to our data set, for example, the life span of items in Twitter are too short to be used in recommender systems. Deep analysis of influence in terms of retweets and mentions is given in [7]. Notion of influence similar to ours is derived in [3,8] for Flickr and Twitter cascades, respectively. Note that by our measurement the Last.fm data contains only a negligible amount of cascades as opposed to Twitter or Flickr.

Finally we mention that several results that we list in [27] show the influence of friends and contacts to spread many properties in social networks. Others question the methodology of these experiments [21, **?**] by proposing that the measured effects may be due to homophily [25, **?**], the fact that people tend to associate with others like themselves, and a shared environment also called confounding or contextual influence.

In our experiment we subtract external effects that may result in friends listening to similar music. Homophily is handled by collaborative filtering, a method that is capable of learning patterns of similarity in taste without using friendship information. External events and information sources (e.g. mass media) may result in temporal increase in popularity. We filter external trends by a method that measures popularity at the given time and recommends based on the momentary popularity.

2 Network influence

The key concept in this paper is a user v influencing another u to scrobble some new artist. Whenever a user u *first time scrobbles* an artist a in her timeline, we investigate whether another user could have influenced u to listen a. We may well expect influence among the causes when u scrobbles artist a the first time at time t, after v last scrobbling the same artist at some time $t' < t$ before. The time difference $\Delta t = t - t'$ is the *delay*, as seen in Fig. 1. Our key assumption is that we observe such a subsequent first time scrobbling between non-friends only by coincidence while some of these events between friends are the result of certain interaction. Our goal is to prove that friends indeed influence each other and this effect can be exploited for recommendations.

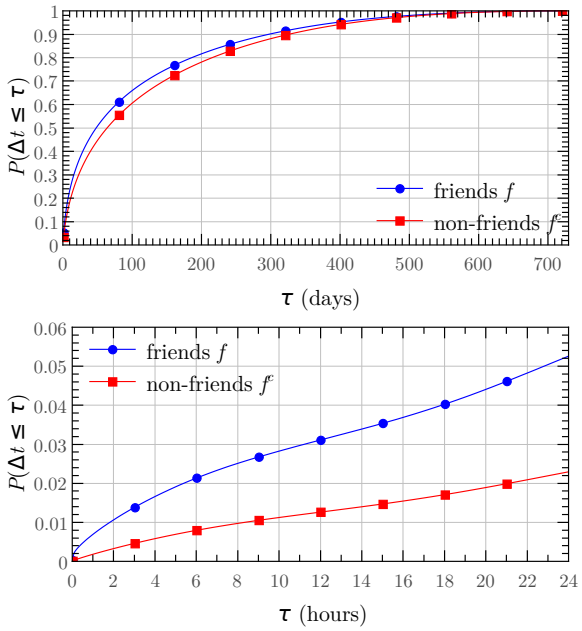

Fig. 2: Fraction of subsequent first time scrobbles with delay $\Delta t \leq \tau$ as the function of τ, in case of friends ($P(\Delta t \leq \tau \mid f)$) and non-friends ($P(\Delta t \leq \tau \mid f^c)$) over the entire timeline (**top**) and the first 24 hours (**bottom**).

Similar influence definitions are given in [3,8,15]. As detailed in [3], one main difference between these definitions is that in some papers t' is defined as the first and not the last time when user v scrobbles a. The smaller the delay Δt between the scrobbles of v and u, the more certain we are that u is affected by the previous scrobble of v. The distribution of delay with respect to friends and non-friends will help us in determining the frequency and strength of influence over the Last.fm social network.

In what follows, we consider *first time scrobbles*, events when a user scrobbles an artist for the first time in her timeline. Out of the 57,274,158 first-time scrobbles of a certain artist a by some user, we find a friend who scrobbled a before 10,993,042 times (19%) in the whole time series and 4,203,109 times in the second year. Note that one user can be influenced by more friends, therefore the total number of influences is 24,204,977. If we only consider influences with delay less than one week, this number reduces to 4,625,141. Note that there is no influencing user for the very first scrobbler of a in the data set. For other scrobbles there is always an earlier scrobble by some other user, however, that user may not be a friend of u. Some of the observed subsequent scrobbles may result by pure coincidence, especially when a new album is released or the popularity of the artist increases for some other reason.

In order to quantify real influence within the set of subsequent first time scrobbles, our goal is to determine the probability that the subsequent scrobbles are result of influence. If we condition this probability for friends and by the time of delay τ, we should obtain a monotonically decreasing function $\mathrm{Infl}(\tau)$.

To formalize, let us consider the probability space of subsequent first time scrobbles among all users. Let I denote the event that an subsequent first time scrobble is the result of an influence. I^c is the opposite, no influence occurs. Coincidence or other, external reason such as the overall increase in popularity causes the subsequent first time scrobble in the time series. Let f denote events between friends and f^c between non-friends. Finally let $\Delta t \leq \tau$ denote the set of events with delay at most τ. With these notations,

$$\mathrm{Infl}(\tau) = P(I \mid \Delta t \leq \tau, f) = \tag{1}$$

$$= \frac{P(I, \Delta t \leq \tau, f)}{P(\Delta t \leq \tau, f)} = \frac{P(\Delta t \leq \tau, I \mid f)P(f)}{P(\Delta t \leq \tau \mid f)P(f)} \tag{2}$$

$$= \frac{P(\Delta t \leq \tau, I \mid f)}{P(\Delta t \leq \tau \mid f)} = \frac{P(\Delta t \leq \tau \mid f) - P(\Delta t \leq \tau, I^c \mid f)}{P(\Delta t \leq \tau \mid f)}. \tag{3}$$

As non-friends f^c should not have any real influence on each other, we assume that

$$P(\Delta t \leq \tau, I^c \mid f) \approx P(\Delta t \leq \tau, I^c \mid f^c) = P(\Delta t \leq \tau \mid f^c). \tag{4}$$

Using this approximation, we can compute the probability of influences between friends as in (1) by expanding (3),

$$\mathrm{Infl}(\tau) = P(I \mid \Delta t \leq \tau, f) \approx \frac{P(\Delta t \leq \tau \mid f) - P(\Delta t \leq \tau \mid f^c)}{P(\Delta t \leq \tau \mid f)}. \tag{5}$$

By the above equation, the influence probability can be approximated by observing the cumulative density curves in Fig. 2. The estimate of this function as in (5) is shown in Fig. 3. As expected, $\mathrm{Infl}(\tau)$ is a monotonically decreasing function of τ. However, the decrease is slow unlike in some recent influence models that propose exponential decay in time [15]. Therefore, we approximate the influence probability with a slowly decreasing logarithmic function instead of an exponential decay,

$$\mathrm{Infl}(\tau) = 1 - c \log \tau, \tag{6}$$

where c is a constant.

3 Influence based recommendation

Based on the measurements in the previous section, we model the observed influences and give a method to apply them for recommendation.

To formalize, let $v \xrightarrow{a;\Delta t \in \mathcal{T}} u$ denote the event that user u scrobbles artist a the first time in her time series, and Δt time after her friend v also scrobbled a. The time difference Δt is restricted to be in a time interval $\mathcal{T}$. We would like to estimate the probability that $v \xrightarrow{a;\Delta t \in \mathcal{T}} u$ happens *and* the reason for this event is influence (I) between the users by a factor that only depends on Δt. First we decompose the full event into a conditional probability as

$$P(I, v \xrightarrow{a;\Delta t \in \mathcal{T}} u) = P(I \mid v \xrightarrow{a;\Delta t \in \mathcal{T}} u) \cdot P(v \xrightarrow{a;\Delta t \in \mathcal{T}} u). \tag{7}$$

In our simple model we consider the right term constant times the length of the time interval, independent of users and time,

$$P(I, v \xrightarrow{a;\Delta t \in \mathcal{T}} u) \approx P(I \mid v \xrightarrow{a;\Delta t \in \mathcal{T}} u) \cdot |\mathcal{T}|. \tag{8}$$

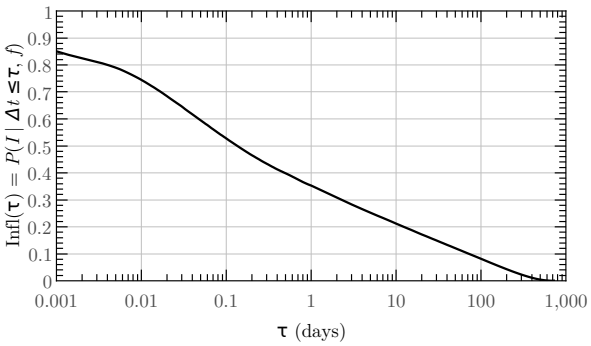

Fig. 3: The influence probability approximated by equation (5), the ratio of increase among friends compared to non-friends, very closely follows a logarithmic function of delay $\Delta t \leq \tau$.

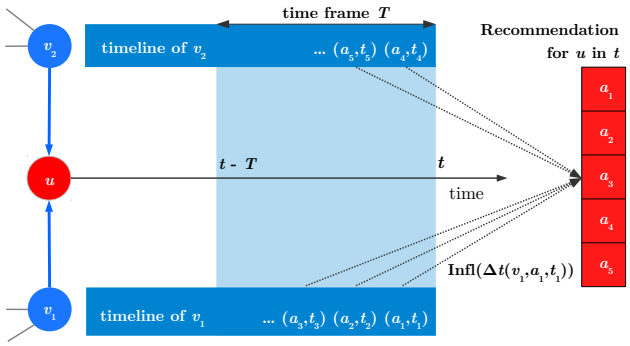

Fig. 4: Scheme of the influence based recommender algorithm.

Note that in our previous work [27], we modeled the left hand term of (7), the strength of the influence between users, by a heuristic formula that counted the number of influence events between the given pair of users. That formula has no mathematical foundation and in our new experiment, modeling by a constant performed better.

When a scrobble event happens at time exactly τ after the scrobble of v, the interval becomes a point and hence we are looking for the derivative of (8),

$$p(I, v \xrightarrow{a;\Delta t=\tau} u) := \lim_{\tau' \to \tau} \frac{P(I, v \xrightarrow{a;\Delta t \leq \tau'} u) - P(I, v \xrightarrow{a;\Delta t \leq \tau} u)}{\tau' - \tau} \tag{9}$$

$$\approx \lim_{\tau' \to \tau} \frac{P(I \mid v \xrightarrow{a;\Delta t \leq \tau'} u) \cdot \tau' - P(I \mid v \xrightarrow{a;\Delta t \leq \tau} u) \cdot \tau}{\tau' - \tau}. \tag{10}$$

We model the conditional probability of an influence in (10) independent of the users and the artist to get the influence function as in (5),

$$P(I \mid v \xrightarrow{a;\Delta t \leq \tau} u) \approx P(I \mid \Delta t \leq \tau) = \mathrm{Infl}(\tau). \tag{11}$$

By using (11) and (6), equation (10) becomes

$$p(I, v \xrightarrow{a;\Delta t = \tau} u) \approx \lim_{\tau' \to \tau} \frac{\mathrm{Infl}(\tau') \cdot \tau' - \mathrm{Infl}(\tau) \cdot \tau}{\tau' - \tau} = 1 - c(1 + \log \tau). \tag{12}$$

As illustrated in Figure 4, the final recommendation model for predicting influence events needs to aggregate the effect of all potential influencers. The events in (7) have very low probability since the right hand term is small. We may also assume independence of the influencers and approximate the effect of all influencers by the sum of the individual influence probabilities. Our final prediction score based on (12) is therefore

$$\hat{r}(u, a, t) = \sum_{v \in n(u)} 1 - c\left(1 + \log(t - t_v)\right), \tag{13}$$

where $n(u)$ denotes the friends of u and t_v is when v last scrobbled a before t.

In an efficient implementation, since the expression (12) decays with τ, we only need to retrieve the past scrobbles of all friends of u. This step is computationally inexpensive unless u has too many friends, when the recommendation is noisy anyway. To speed up computations, we only consider influences with delay not more than a predefined time frame T and hence we set

$$c = 1/\left(1 + \log T\right). \tag{14}$$

With a sufficiently small parameter of the time frame in the range of a few days, our algorithm can hence be implemented even to provide recommendations based on real time updated models.

4 Online matrix factorization

Batch recommender algorithms may iterate several times over the training set until convergence. In an online setting [1], the model needs to be retrained after each new event and hence reiterations over the earlier parts of the data is ruled out.

In this section, we give an online matrix factorization method and show that this method includes components that may learn user influences. In our algorithm, we allow a single iteration over the training data only, and this single iteration processes the events in the order of time. We use scrobbles as positive training instances and generated negative training instances by selecting three random artists uniformly at the time when a user scrobbled an artist.

We use the regularized matrix factorization method of [29], and use the k-factor model for prediction,

$$\hat{r}(u, a, t) = \sum_{i=1}^{k} p_{uit} q_{ait}, \tag{15}$$

where p and q contain the user and artist models, respectively. We optimize for mean squared error with an additional regularization term of weight λ,

$$F = \sum_{u,a,t} \left(r(u,a,t) - \sum_{i=1}^{k} p_{uit} q_{ait} \right)^2 + \lambda \sum_{i=1}^{k} p_{uit}^2 + \lambda \sum_{i=1}^{k} q_{ait}^2. \tag{16}$$

For a single event (u, a, t) that is either a scrobble with $r = 1$ or a generated negative sample with $r = 0$, we optimize the coefficients p_{uit} and q_{ait} for $i = 1, \ldots, k$ by gradient descent with learning rate η as

$$p_{uit} \leftarrow p_{ui(t-1)} + \eta \cdot \left(r(u,a,t) - \sum_{i=1}^{k} p_{ui(t-1)} q_{ai(t-1)} \right) q_{ai(t-1)} - \eta \cdot \lambda \cdot p_{ui(t-1)}; \tag{17}$$

$$q_{ait} \leftarrow q_{ai(t-1)} + \eta \cdot \left(r(u,a,t) - \sum_{i=1}^{k} p_{ui(t-1)} q_{ai(t-1)} \right) p_{ui(t-1)} - \eta \cdot \lambda \cdot q_{ai(t-1)}. \tag{18}$$

Online recommenders seem more restricted than those that may iterate over the data set several times and one would expect inferior quality by the online methods. Online methods however have the advantage of giving much more emphasis on recent events. In some sense, the online methods may incorporate the notion of influence from Section 2: if friends have similar taste and hence similar factor weights, a friend scrobbling some artist a will in the near future strengthen the weight for this artist for all users who have similar taste.

Indeed, assume that an influence $v \xrightarrow{a;\Delta t} u$ happens at time t. When we observe v scrobbling a at time $t - \Delta t$, we update the coefficients $q_{ai(t-\Delta t)}$ in (18). We may expect u and v, as friends, share their taste and hence their coefficients $p_{uit'}$ and $p_{vit'}$ are similar at all times t'. Thus the update of $q_{ai(t-\Delta t)}$ increases the score of artist a for user u at time t, as long as Δt remains small and the effect of the update is not diminished by more recent transactions.

5 Online evaluation

In this section, we describe our method to measure the accuracy of the best items recommended for a user in a timely manner, by looking at the scrobbles of the user at the given time. Influence depends on time and no matter how relative slow, the influential power of a friend scrobbling an artist decays as time passes by. For this reason, the influence based recommender must learn on the fly. Next we show why evaluating on the fly recommenders is challenging.

Recommender systems in practice need to rank the best K items for the user. In this top-K recommendation task [10,9] the goal is not to rate some of the individual items but to provide the best candidates. Despite the fact that only prediction for the top list matters in top-K evaluation, several authors propose models trained for RMSE with good top-K performance [19,31] and hence we follow their approach.

In a time sensitive or online recommender that potentially retrains its model after each and every scrobble, we have to generate new top-K recommendation list for every single scrobble in the test period. The online top-K task is hence different from the standard recommender evaluation settings, since there is always a single item only in the ground truth and the goal is to aggregate the rank of these single items over the

entire testing period. For our task, we need carefully selected quality metrics that we describe next.

One possible measure for the quality of a recommended top list of length K could be precision and recall [32,33]. Note that we evaluate against a single scrobble. Both the number of relevant (1) and the number of retrieved (K) items are fixed. Precision is $1/K$ if we retrieve the single item scrobbled and 0 otherwise. Recall is 0 if we do not retrieve the single relevant item and 1 otherwise. The value of K that maximizes precision is the rank of the item scrobbled and hence "maximal precision" follows the function of 1/rank.

Recently, measures other than precision and recall are preferred for measuring the quality of top-K recommendation [2]. The most common measure is nDCG that is a normalized version of the discounted cumulative gain (DCG) with threshold K

$$\mathrm{DCG@K}(a) = \begin{cases} 0 & \text{if } \mathrm{rank}(a) > K; \\ \dfrac{1}{\log_2(\mathrm{rank}(a)+1)} & \text{otherwise.} \end{cases} \tag{19}$$

Since DCG is a slower decreasing function of the rank than what we observed for maximal precision, DCG is more advantageous since we have a large number of artists of potential interest to each user. Our choice is in accordance with the observations in [2] as well.

Note that in our unusual setting of DCG evaluation, there is a single relevant item and hence for example no normalization is needed as in case of the DCG measure. Also note that the DCG values will be small since the nDCG of a relative short sequence of actual scrobbles will roughly be equal to the sum of the individual DCG values. The DCG measured over 100 subsequent scrobbles of different artists cannot be more than the ideal DCG, which is $\sum_{i=1}^{100} 1/\log_2(i+1) = 20.64$ in this case (the ideal value is 6.58 for $K = 20$). Hence the DCG of an individual scrobble will on average be less than 0.21 for $K = 100$ and 0.33 for $K = 20$.

6 Music Recommendation Baseline Methods

We describe one baseline method based on dynamic popularity in Section 6.1 and two more based on matrix factorization in the subsequent subsections. In Section 6.2 we describe the settings of a standard method, and finally in Section 6.3 we add regularization over friendship as in [22]. The first two methods were used as baseline in our preliminary experiments [27].

6.1 Dynamic popularity based recommendation

Given a predefined time frame T as in Section 3, equation (14), at time t we recommend an artist based on the popularity in time not earlier than $t - T$ but before t. In our algorithm we update the counts and store artists sorted by the current popularity. In one time step, we may either add a new scrobble event or remove the earliest one, corresponding to a count increment or decrement. For globally popular items, the sorted order can be maintained by a few changes in the order only. To speed up the procedure, we may completely ignore part of the long tail and for others update the position only after a sufficiently large change in count.

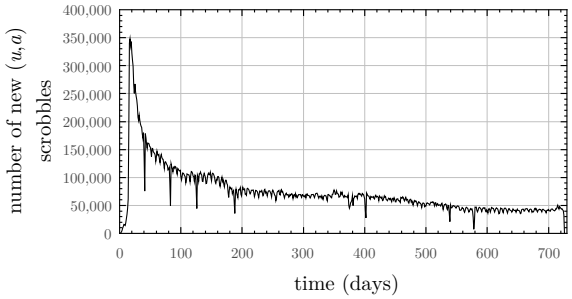

Fig. 5: Number of new (u, a) scrobbles as the function of time.

6.2 Batch factor model based recommendation

For our factor model based recommender, we use the standard regularized stochastic gradient descent implementation [14, 29]. In the testing period, we trained weekly models based on all data before the given week. For each user, we constructed three times as many negative training instances as positive by selecting random artists with probability proportional to their popularity in the training period. Each testing period lasted one week. For each user, we compute a top list of predictions once for the entire week and evaluate against the sequence of scrobbles in that week.

6.3 Social regularization

Ma et al. [22] propose a method to implement constraints in a factor model based recommender algorithm for keeping the profile of friends similar. We implemented both the average-based and the individual-based regularization of [22] and found the latter superior, hence we use individual-based regularization in our experiments. Note that these algorithms have no knowledge of time and hence cannot incorporate our notion of subsequent first time scrobbles as in Section 2, even though they may work very well for other, non-first-time scrobbles that we do not consider in this paper.

7 Experiments

In this section, we measure the quality of our recommendation methods. Out of the two year scrobbling data, we use the full first year as training period. The second year becomes the testing period where we consider scrobbles one by one. We allow a recommender algorithm to use part or full of the data before the scrobble in question for training and require a ranked top list of artists as output. We evaluate the given single actual scrobble a in question against the recommended top list of length K. As seen in Fig. 5, by the second year, the number of first-time scrobbles stabilize around 50,000 a day after the artificial peak in the beginning caused by the lack of earlier data. For the reason of stability, we measure our recommender methods in Year 2 of the timeline.

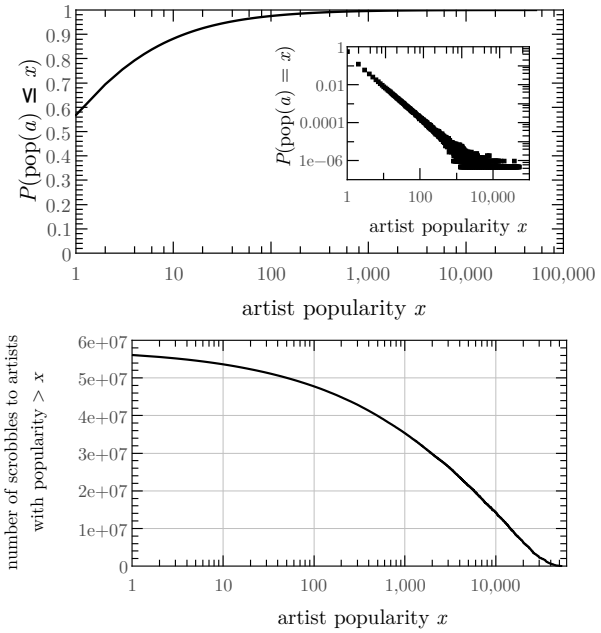

Fig. 6: **Top:** Distribution of scrobble count to a given artist and the cumulative distribution. **Bottom:** Fraction of scrobbles for artists with popularity at least a given value x, as the function of x.

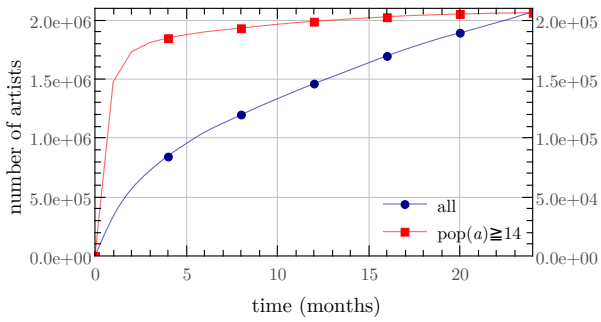

Fig. 7: The number of different artists scrobbled before a given time in the two year period of the data set.

In our evaluation we discard infrequent artists from the data set both for efficiency considerations and due to the fact that our item based recommenders will have too little information on them. As seen in Fig. 6, top, the number of artists with a given scrobble count follow a power-law distribution with near 60% of the artists appearing only once. While 90% of the artists gathered less than 20 scrobbles in two years, as seen in Fig. 6, bottom, they attribute to only less than 10% of the data set. In other

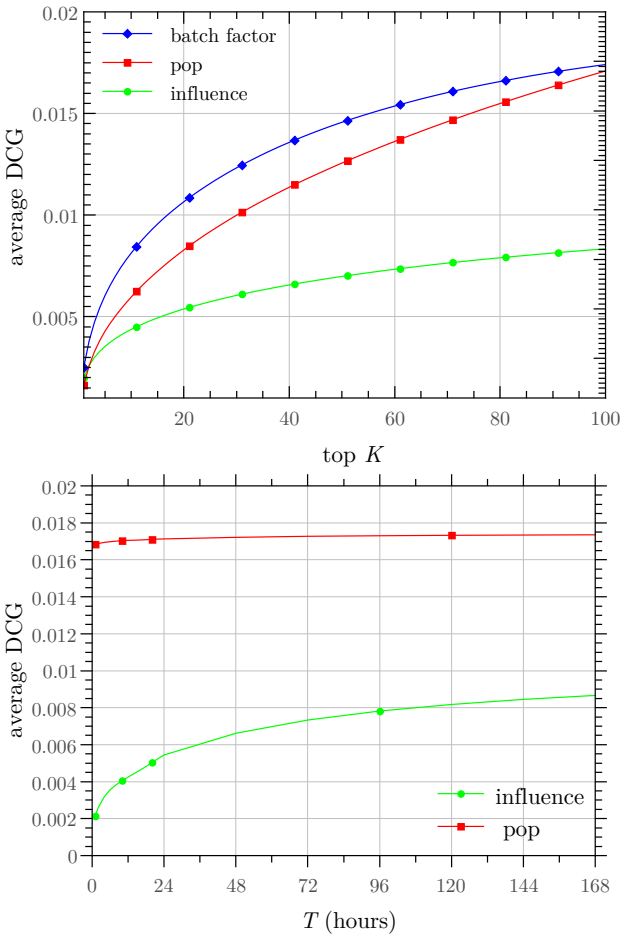

Fig. 8: **Top:** DCG@K as the function of K for the batch algorithms, for a time window T equal to one week. **Bottom:** DCG@100 defined by equation (19) as the function of the time window threshold T as in Section 3.

words, by discarding a large number of artists, we only lose a small fraction of the scrobbles. For efficiency we only consider artists of frequency more than 14.

As time elapses, we observe near linear increase in the number of artists that appear in the data set in Fig. 7. This figure shows artists with at least 14 scrobbles separately. Their count grows slower but still we observe a large number of new artist that appear in time and exceed the minimum count of 14. Very fast growth for infrequent artists may be a result of noise and unidentified artists from e.g. YouTube videos and similar Web sources.

Under various settings, we give daily, monthly and full year average DCG@K defined by equation (19). The final conclusion of the experiments is drawn by blending five recommenders, i.e. linearly combining their output. In our experiments we obtained the best results by linearly combining the values of 1/rank for each item instead of the

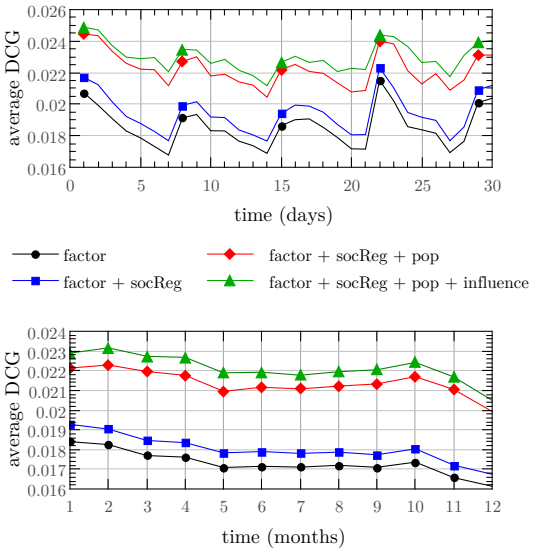

Fig. 9: DCG@100 as in equation (19) as the function of time. **Top**: Daily averages in a one month sample of the test period. **Bottom**: Monthly averages in the second year.

predicted score. As an advantage of 1/rank, we need no score normalization. The best combined results will be given in Figs. 9–14 and Tables 1–4. We give a detailed analysis of the parameters and the weights in the combinations in Section 7.1 for batch and in Section 7.2 for online matrix factorization.

7.1 Batch recommenders

We start by assessing the global parameters of the experiments. Parameter K in equation (19) controls the length of the top list considered for evaluation. In other words, K can be interpreted as the size of the list presented to the user. Practically K must be small in order not to flood the user with information. The dependence on the top list size K is measured in Fig. 8, top, for $K \leq 100$. We observe that our influence based method saturates the fastest. This is due to the fact that the number of items recommended to a given user is usually small unless the user has a large number of very active friends. For this reason, we give linear combination results not just for the value $K = 20$ that we consider practically feasible, but also for 100 for comparison.

The popularity and influence based methods depend on the time frame. The longer we look back in time, the more artists we can recommend. If we carefully set the rank as a function of time, wider time frames are advantageous for quality but put extra computational load. For the influence recommender T is the maximum delay Δt that we consider as influence while for the popularity one T is the time interval that we use for frequency computation. We ran measurements in the second year test period with different time frames T and computed the average DCG performance of the

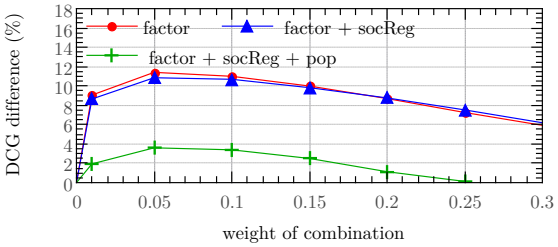

Fig. 10: Linear combination DCG@100 defined by (19) as the function of the linear combination weight for the influence recommender. Different curves correspond to different baseline methods.

recommender systems. Figure 8, bottom, shows the average DCG scores with different time frames. The performance only slowly increases for time frames longer than a day. In what follows we set T to be one week.

Next we set the parameters of the factor models. We use matrix factorization (see Section 6.2) and the social regularization variant [22] with the following parameters that turned out to perform best in our experiments. We set the learning rate $= 0.01$, $k = 20$, initial feature value $= 0.1$ and 20 iterations. We re-train the regularized matrix factors each week based on all past data. For this reason we see weekly periodicity in the one month timeline of Fig. 9: the batch factor model performs best immediately after the training period and slowly degrades in each one-week testing period. Since the online model has no retraining, we observe stable performance in time in Fig. 12 with a best learning rate of 0.1. Notice the contrast with the batch model that has, for best performance, ten times lower learning rate but 20 iterations instead.

First we improved the batch factor model based recommenders by combining them with the influence model. In all cases the combination of the influence model further improved our recommender systems.

In Fig. 9, we measure a stack of gradually stronger combinations of batch recommenders in a short daily and a full period monthly averaging. The batch factor model is slightly improved by social regularization. Major improvement is obtained by combining with temporal popularity. Finally, influence recommendation yields improvement over the full stack.

The relative improvement as the function of the influence weight in the combination is seen in Fig. 10. First we combined the influence recommender to a batch factor model and a factor model with social regularization. Finally, we selected the best combination (7:3) of our social regularization contained factor and popularity recommenders and combined it with the influence model.

In Tables 1–2 we summarize the average DCG@20 and DCG@100 curves in the testing period in case of the different combined batch recommenders.

7.2 Online models

Now we show our results using the online factor model. First of all, it turns out that this model is much stronger than its batch version. In addition, it already incorporates

	DCG@100	best combined DCG@100	improvement (%)
batch factor	0.017407	0.019393	11.41
batch factor + social regularization	0.018133	0.020101	11.40
batch factor + social reg. + popularity	0.021482	0.022256	3.75

Table 1: Best batch recommender accuracy and their combination with the influene recommender, K=100.

	DCG@100	best combined DCG@20	improvement (%)
batch factor	0.010660	0.011449	7.40
batch factor + social regularization	0.011079	0.011450	7.41
batch factor + social reg. + popularity	0.011114	0.011592	4.30

Table 2: Best batch recommender accuracy and their combination with the influene recommender K=20.

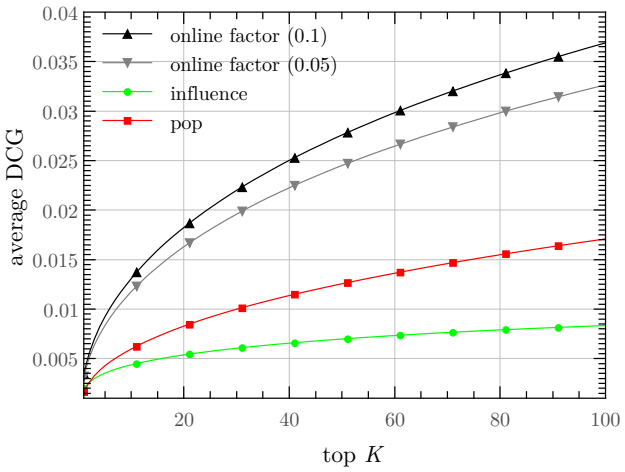

Fig. 11: DCG@K as the function of K for the online method with two different learning rates.

popularity, and also social regularization: these combinations gave no improvement for the entire range of combination weights.

As in Section 7.1, we first investigate the parameters of the recommenders. The dependence on the top list size K is similar to the batch methods, as seen in Fig. 11. We use the same values except we experiment with learning rate in the range of 0.01–0.5, as seen in Fig. 12. In this figure we also show the the improvement achieved by combining with the influence recommender. The online factor model cannot be improved by combining with any method other than the influence recommender and hence the combination of these two gives our strongest result. One can see that the final combination of the online methods outperforms the best combination of popularity and batch matrix factorization with social regularization for DCG@100.

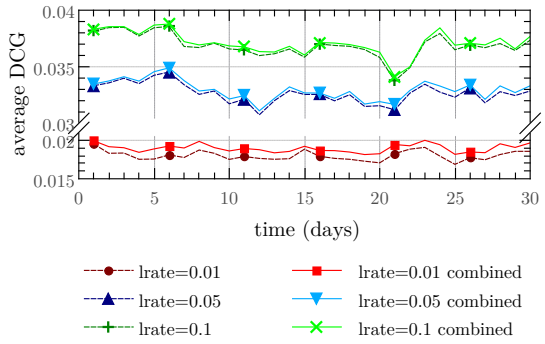

Fig. 12: DCG@100 as in equation (19) as the function of time for online factor models with different learning rates and their best combination with the influence recommender.

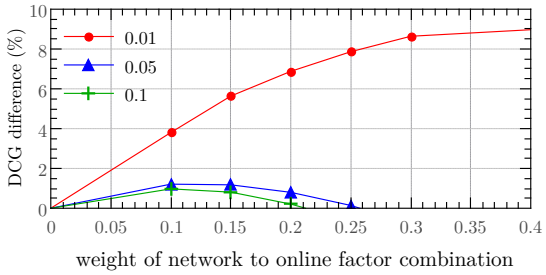

Fig. 13: Linear combination DCG@100 defined by (19) as the function of the linear combination weight for the network influence method. Different curves correspond to combined online factor models with different learning rates.

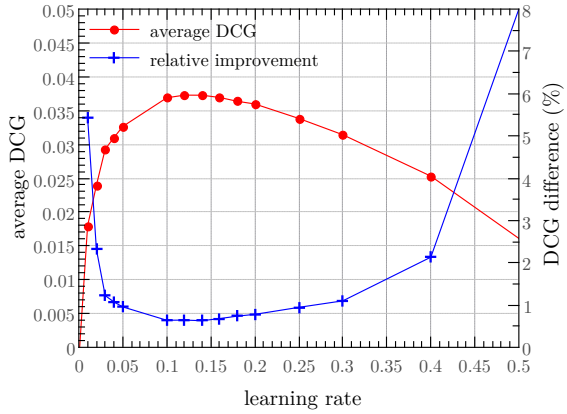

Fig. 14: Average DCG@100 (left axis) and combination relative improvement (right axis) for the online matrix factorization method as the function of the learning rate.

	DCG@100	best combined DCG@100	improvement (%)
online factor with learning rate 0.01	0.017950	0.018925	5.43
online factor with learning rate 0.05	0.032655	0.032959	0.93
online factor with learning rate 0.1	0.036890	0.037135	0.66

Table 3: The performance of the online factor model with different learning rates and their best combination with the influence recommender, K=100.

	DCG@100	best combined DCG@20	improvement (%)
online factor with learning rate 0.01	0.008478	0.009238	8.97
online factor with learning rate 0.05	0.016307	0.016504	1.21
online factor with learning rate 0.1	0.018248	0.018425	0.97

Table 4: The performance of the online factor model with different learning rates and their best combination with the influence recommender, K=20.

In our final experiment for obtaining the best recommender, we only combine the online factor model with network influence in Fig. 13. Note that as explained in Section 4, both models incorporate influence effects in their models. As expected, the stronger the factor model, the lower the improvement but it remains near 1% even in the strongest case. As the function of the learning rate, we can see both DCG and the relative improvement in Fig. 14.

Finally, Tables 3–4 summarize the average DCG@20 and DCG@100 curves in the testing period in case of the different combined recommenders. Note the strong improvement over the batch results in Tables 1–2.

Conclusions

Based on a 70,000-entry sample of Last.fm users, we were able to exploit the immediate temporal effect of users influencing the taste of friends for improving the quality of music recommendation. Over baseline recommenders, we achieved a 4% improvement in recommendation accuracy by combining them with presenting artists from friends' recent scrobbles that the given user had never seen before. Furthermore, we used our time-aware online matrix factorization method combined with our influence recommender and achieved significantly better results than in our batch experiments. Our system has very strong time sensitivity: when we recommend, we look back in the near past and combine friends' scrobbles with a factor model that is updated after each and every scrobble event. The influence from a friend at a given time is certain function of the observed influence in the past and the time elapsed since the friend scrobbled the given artist.

Our best methods learn online and provide top-K recommendation lists recomputed for each and every user query. Because of the inherent time dependence, we reviewed the available metrics and identified average DCG as a good candidate for time-aware recommender evaluation.

Acknowledgments

To the Last.fm team for preparing us this volume of the anonymized data set that cannot be efficiently fetched through the public Last.fm API. To Bálint Daróczy and Levente Kocsis for discussions and help in implementing the matrix factorization methods.

References

1. Abernethy, J., Canini, K., Langford, J., Simma, A.: Online collaborative filtering. University of California at Berkeley, Tech. Rep (2007)
2. Al-Maskari, A., Sanderson, M., Clough, P.: The relationship between ir effectiveness measures and user satisfaction. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 773–774. ACM (2007)
3. Bakshy, E., H., J.M., Mason, W.A., Watts, D.J.: Everyone's an influencer: quantifying influence on twitter. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp. 65–74. ACM (2011)
4. Bell, R.M., Koren, Y.: Lessons from the netflix prize challenge. ACM SIGKDD Explorations Newsletter **9**(2), 75–79 (2007)
5. Bennett, J., Lanning, S.: The netflix prize. In: KDD Cup and Workshop in conjunction with KDD 2007 (2007)
6. Bonchi, F.: Influence propagation in social networks: A data mining perspective. IEEE Intelligent Informatics Bulletin **12**(1), 8–16 (2011)
7. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, K.: Measuring user influence in twitter: The million follower fallacy. In: 4th International AAAI Conference on Weblogs and Social Media (ICWSM) (2010)
8. Cha, M., Mislove, A., Adams, B., Gummadi, K.P.: Characterizing social cascades in flickr. In: Proceedings of the first workshop on Online social networks, pp. 13–18. ACM (2008)
9. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the fourth ACM conference on Recommender systems, pp. 39–46. ACM (2010)
10. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS) **22**(1), 143–177 (2004)
11. Domingos, P., Richardson, M.: Mining the network value of customers. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 57–66. ACM (2001)
12. Eck, D., Lamere, P., Bertin-Mahieux, T., Green, S.: Automatic generation of social tags for music recommendation. Advances in neural information processing systems **20**, 385–392 (2007)
13. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics **29**(5), 1189–1232 (2001)
14. Funk, S.: Netflix update: Try this at home. http://sifter.org/˜ simon/journal/20061211.html (2006)
15. Goyal, A., Bonchi, F., Lakshmanan, L.V.: Learning influence probabilities in social networks. In: Proceedings of the third ACM international conference on Web search and data mining, pp. 241–250. ACM (2010)
16. Hu, X., Bay, M., Downie, J.: Creating a simplified music mood classification ground-truth set. In: Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR'07) (2007)
17. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. Knowledge Discovery in Databases: PKDD 2007 pp. 506–514 (2007)
18. Knees, P., Pohle, T., Schedl, M., Widmer, G.: A music search engine built upon audio-based and web-based similarity measures. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 447–454. ACM (2007)

19. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 426–434. ACM (2008)
20. Kwak, H., Lee, C., Park, H., Moon, S.: What is twitter, a social network or a news media? In: Proceedings of the 19th international conference on World wide web, pp. 591–600. ACM (2010)
21. Lyons, R.: The spread of evidence-poor medicine via flawed social-network analysis. Statistics, Politics, and Policy **2**(1), 2 (2011)
22. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp. 287–296. ACM (2011)
23. Markines, B., Cattuto, C., Menczer, F., Benz, D., Hotho, A., Stumme, G.: Evaluating similarity measures for emergent semantics of social tagging. In: 18th International World Wide Web Conference, pp. 641–641. Citeseer (2009)
24. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Ht06, tagging paper, taxonomy, flickr, academic article, to read. In: Proceedings of the seventeenth conference on Hypertext and hypermedia, pp. 31–40. ACM (2006)
25. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual review of sociology pp. 415–444 (2001)
26. Noel, J., Sanner, S., Tran, K.N., Christen, P., Xie, L., Bonilla, E.V., Abbasnejad, E., Della Penna, N.: New objective functions for social collaborative filtering. In: Proceedings of the 21st international conference on World Wide Web, pp. 859–868. ACM (2012)
27. Pálovics, R., Benczúr, A.A.: Temporal influence over the last. fm social network. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 486–493. ACM (2013)
28. Pilászy, I., Tikk, D.: Recommending new movies: even a few ratings are more valuable than metadata. In: Proceedings of the third ACM conference on Recommender systems, pp. 93–100. ACM (2009)
29. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Investigation of various matrix factorization methods for large recommender systems. In: Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, pp. 1–8. ACM (2008)
30. Tso-Sutter, K., Marinho, L., Schmidt-Thieme, L.: Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: Proceedings of the 2008 ACM symposium on Applied computing, pp. 1995–1999. ACM (2008)
31. Weimer, M., Karatzoglou, A., Smola, A.: Adaptive collaborative filtering. In: Proceedings of the 2008 ACM conference on Recommender systems, pp. 275–282. ACM New York, NY, USA (2008)
32. Yang, X., Steck, H., Guo, Y., Liu, Y.: On top-k recommendation using social networks. In: Proceedings of the sixth ACM conference on Recommender systems, pp. 67–74. ACM (2012)
33. Yuan, Q., Chen, L., Zhao, S.: Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In: Proceedings of the fifth ACM conference on Recommender systems, pp. 245–252. ACM (2011)