Exploiting Temporal Influence in Online Recommendation

Róbert Pálovics[1,2] András A. Benczúr[1,3] Levente Kocsis[1,4]
Tamás Kiss[1,3] Erzsébet Frigó[1,2]
[1]Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
[2]Technical University Budapest [3]Eötvös University Budapest [4]University of Szeged
{rpalovics, benczur, kocsis, kisstom, fbobee}@ilab.sztaki.hu

ABSTRACT

In this paper we give methods for time-aware music recommendation in a social media service with the potential of exploiting immediate temporal influences between users. We consider events when a user listens to an artist the first time and this event follows some friend listening to the same artist short time before. We train a blend of matrix factorization methods that model the relation of the influencer, the influenced and the artist, both the individual factor decompositions and their weight learned by variants of stochastic gradient descent (SGD). Special care is taken since events of influence form a subset of the positive implicit feedback data and hence we have to cope with two different definitions of the positive and negative implicit training data. In addition, in the time-aware setting we have to use online learning *and* evaluation methods. While SGD can easily be trained online, evaluation is cumbersome by traditional measures since we will have potentially different top recommendations at different times. Our experiments are carried over the two-year "scrobble" history of 70,000 Last.fm users and show a 5% increase in recommendation quality by predicting temporal influences.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

temporal recommendation and evaluation; social influence; online matrix factorization; Last.fm; music recommendation

1. INTRODUCTION

Part of the appeal of Web 2.0 is to find other people who share similar interests. Last.fm organizes its social network around music recommendation: users may automatically share their listening habits and at the same time grow their friendship. Based on the profiles shared, users may see what artists friends really listen to the most.

In a recent paper [22], we proved the existence of the influence of friends on musical taste by carefully decoupling trends and homophily, the fact that friends are a priori more likely to have similar taste. In this paper we exploit the timely information gathered by the Last.fm service on users with public profile to exploit the potential influence between friends for recommendation. Last.fm's service is unique in that we may obtain a detailed timeline and catch immediate effects by comparing the history of friends in time and comparing to pairs of random users instead of friends.

As our main contribution, we give a matrix factorization mixture model for influence between friends that yield improved collaborative filtering methods. In the simplest setting, we may recommend a new artist a to a user u closely after a friend v listened to the same artist. When we turn to modeling the tensor data $< u, v, a >$ that may even involve the time elapsed since v listening to a, we face a very sparse problem. Hence instead of modeling the tensor, we flatten out along the variables and define three matrices in addition to single-variable effects similar to the ones defined by the centralization procedures of [4].

Since influence from friends has a very strong time dependence in that only the events of the last few hours or days may have an effect on the user behavior, in this paper we consider online learning with very strong time sensitivity. Compared to standard collaborative filtering methods, we process events only once and in the order they have appeared. As baseline we use online stochastic gradient descent (SGD) 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 and involving an online updated item bias, but also part of friends' influence. Immediately after a user listens to an artist, the corresponding factor weights are relative strongly adjusted due to the 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 an implicit variant of an influence recommender by itself that we will further improve by a direct modeling of the influences.

To obtain the weighted combination of the baseline and the influence recommenders, we propose a new method for online learning user-dependent blending weights. If the derivatives of the individual models are available, a single SGD could optimize both the internal parameters and the blending weights. However as it turns out, the influence recom-

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645723.

mender requires a different set of implicit positive items and a procedure for generating a negative sample than the traditional online matrix factor model.

In our blend, we obtain a 5% of increase in quality, a strong result in view of the three-year Netflix Prize competition [6] to improve recommender quality by 10%. The fact that influences blend well with collaborative filtering and temporal effects prove that close events in the network bring in new information: friends' close events in the past can be exploited in a recommender system.

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

The rest of this paper is organized as follows. After describing our Last.fm data in Section 2, we explore for measurable signs of influence by friends in Section 3. Our main influence recommender is defined in Section 4, our online evaluation metric in Section 5, the online blending method in Section 6 and the baseline algorithms in Section 7. Finally we show our measurements for improved recommendation quality in Section 8.

1.1 Related results

The Netflix Prize competition [6] has recently generated increased interest in recommender algorithms in the research community and put recommender algorithms under a systematic thorough evaluation on standard data [5]. The final best results blended a very large number of methods whose reproduction is out of the scope of this paper.

Bonchi [7] 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. Notion of influence similar to ours is derived in [3, 8] for Flickr and Twitter cascades, respectively.

Closest to our results are the applications of network influence in collaborative filtering under the term of "social regularization" [18, 21, 25, 26]. These results add smoothing to 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 [13], 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]. 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 [14]. Similarity search based on audio is given in [16]. Tag based music recommenders [12, 23] 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 [15, 19, 20].

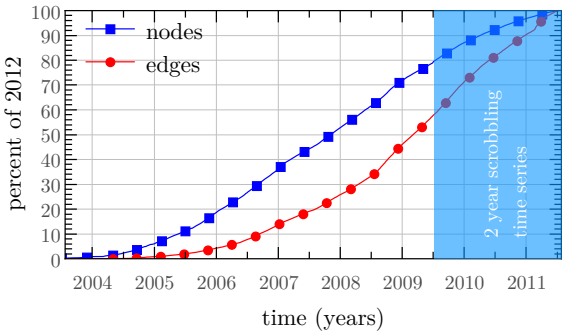

Figure 1: The number of the users and friendship edges in time as the fraction of the values at the time of the data set creation (2012).

2. THE LAST.FM DATA SET

Last.fm became a relevant online service in music based social networking. For registered users, it collects, "scrobbles"[1] what they have listened. Each user has its own statistics on listened music that is shown in her profile. Most user profiles are public, and each user of Last.fm may have friends inside the Last.fm social network. Therefore one relevant information for the users is that they see their own and their friends' listening statistics.

We investigate a data set that consists of the contacts and the implicit feedback timeline, the "scrobble history" of the users. Our goal is to exploit the influence of social contacts for recommendation. For privacy considerations, throughout our research, we selected an anonymous sample of users. Anonymity is provided by selecting random users while maintaining a connected friendship network. We set the following constraints for random selection:

- 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.
- At least 10 friends that meet the first four conditions.

The above selection criteria were set to select a representative part of Last.fm users and as much as possible avoid users who artificially generate inflated scrobble figures. In this anonymized data set of two years of artist scrobble timeline, edges of the social network are undirected and timestamped by creation date (Fig. 1). The number of users both in the time series and in the network is 71,000 with 285,241 edges; no edges are ever deleted from the network.

The time series contain 979,391,001 scrobbles from 2,073,395 artists and were collected between 01 January 2010 and 31 December 2011. The same user can scrobble an artist several times. The number of unique user-artist scrobbles is 57,274,158.

[1]The name "scrobbling" is a word by Last.fm, meaning the collection of information about user listening.

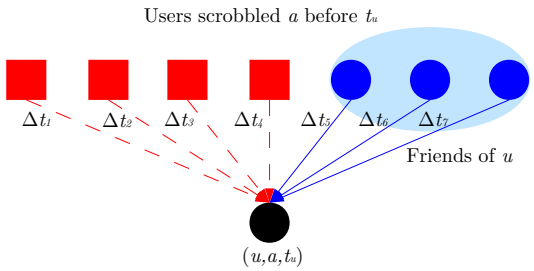

Figure 2: Potential influence on u by other users to scrobble (u, a, t_u).

3. NETWORK INFLUENCE

The key concept in this paper is a user v influencing another u to scrobble a. The sign of an influence is if u scrobbles artist a the first time at time t_u, after v last scrobbling the same artist at some time $t_v < t_u$ before. The time difference $\Delta t = t_u - t_v$ is the *delay*, as seen in Fig. 2. 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.

Similar influence definitions are given in [3, 8, 13]. As detailed in [3], one main difference between these definitions is that in some papers t_v 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.

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 a limit t on the delay, we should obtain a monotonically decreasing function $\text{Infl}(t)$.

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

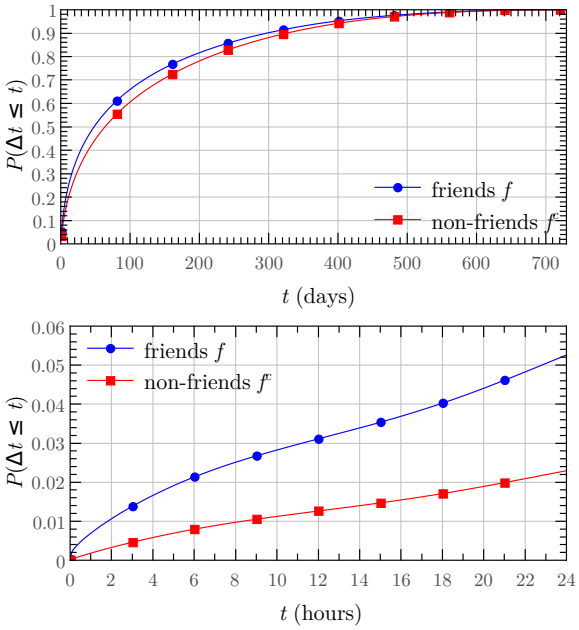

Figure 3: Fraction of subsequent first time scrobbles with delay $\Delta t \leq t$ as the function of t, in case of friends ($P(\Delta t \leq t \mid f)$) and non-friends ($P(\Delta t \leq t \mid f^c)$) over the entire timeline (**top**) and the first 24 hours (**bottom**).

the time series. Let f denote events between friends and f^c between non-friends. Finally let $\Delta t \leq t$ denote the set of events with delay at most t. With these notations,

$$\text{Infl}(t) = P(I \mid \Delta t \leq t, f) = \quad (1)$$

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

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

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

$$P(\Delta t \leq t, I^c \mid f) \approx P(\Delta t \leq t, I^c \mid f^c) = P(\Delta t \leq t \mid f^c). \quad (4)$$

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

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

By the above equation, the influence probability can be approximated by observing the cumulative density curves in Fig. 3. The estimate of this function as in (5) is shown in Fig. 4. As expected, $\text{Infl}(t)$ is a monotonically decreasing function of t. However, the decrease is slow unlike in some recent influence models that propose exponential decay in time [13]. Therefore, we approximate the influence probability with a slowly decreasing logarithmic function instead of an exponential decay,

$$\text{Infl}(t) = 1 - c \log t, \quad (6)$$

where c is a constant.

4. INFLUENCE BASED RECOMMENDATION

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

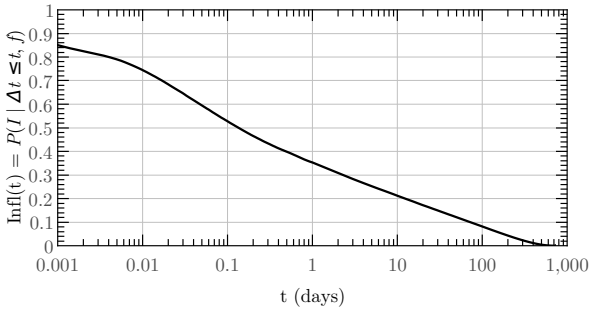

Figure 4: The influence probability approximated by equation (5), the ratio of increase among friends compared to non-friends very closely follows a logarithmic function of delay $\Delta t \leq t$.

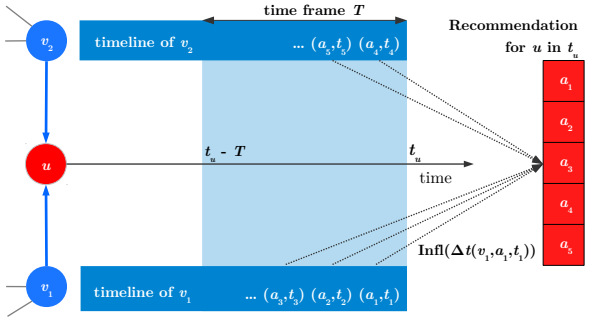

Figure 5: Scheme of the influence based recommender algorithm.

recommendation. 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 online.

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 $\mathcal{T}$. As illustrated in Fig. 5, we would like to decompose 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 into a factor that only depends on Δt and another one that is independent of Δt. First we decompose the full event into a conditional probability as

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

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

$$\lim_{\tau \to 0} \left(P(I, v \xrightarrow{a;\Delta t \leq t+\tau} u) - P(I, v \xrightarrow{a;\Delta t \leq t} u) \right) / \tau. \quad (8)$$

We model the right hand side of (7), the "strength" of the influence between users u and v, independent of time as

$$f(v \xrightarrow{a} u) := P(v \xrightarrow{a;\Delta t \in \mathcal{T}} u)/|\mathcal{T}|, \quad (9)$$

hence by equation (7) we may divide the derivative (8) by $f(v \xrightarrow{a} u)$ to get

$$\lim_{\tau \to 0} \left(P(I \mid v \xrightarrow{a;\Delta t \leq t+\tau} u)(t+\tau) - P(I \mid v \xrightarrow{a;\Delta t \leq t} u)t \right) / \tau. \quad (10)$$

We model influence conditional probabilities by a global function for all users that depend only on the time Δt elapsed,

$$P(I \mid v \xrightarrow{a;\Delta t \leq t} u) \approx P(I \mid \Delta t \leq t, f). \quad (11)$$

By using our function in (5) and (6), equation (10) becomes

$$\lim_{\tau \to 0} \left((1 - c \log(t+\tau)) \cdot (t+\tau) - (1 - c \log t) \cdot t \right) / \tau \quad (12)$$

$$= 1 - c(1 + \log t). \quad (13)$$

Now we give our matrix factorization model for $f(v \xrightarrow{a} u)$. We decompose the model into seven terms that give a global model for one or more of the three variables u, v, a in $(v \xrightarrow{a} u)$. By replacing variables considered globally by $\bullet$ and noting that the last term with all three variables global is a constant, we get

$$\begin{aligned} f(v \xrightarrow{a} u) \sim\ & w_0 + w(\bullet \xrightarrow{\bullet} u) + w(v \xrightarrow{\bullet} \bullet) + w(\bullet \xrightarrow{a} \bullet) \\ & + w(\bullet \xrightarrow{a} u) + w(v \xrightarrow{a} \bullet) + w(v \xrightarrow{\bullet} u). \end{aligned} \quad (14)$$

We have four global effects, a constant, an influencer, an influenced, and an artist, and three bivariate terms that can be modeled by matrix factorization as

$$\begin{aligned} f(v \xrightarrow{a} u) \sim\ & \alpha_0 + \alpha_1 b_v + \alpha_2 b_u + \alpha_3 b_a \\ & + \vec{U}\vec{A} + \vec{A'}\vec{V} + \vec{U'}\vec{V'}. \end{aligned} \quad (15)$$

The three bias terms b_v, b_u and b_a correspond to the frequencies of user v influencing, user u being influenced and influences occurred with artist a. $\alpha_1, \ldots, \alpha_3$ are learned weights of the biases, and α_0 is the global learned bias. The six vectors correspond to six different latent vectors.

The final prediction score $\hat{r}$ is based on (8), by using (15) and (13) we can write it in a form

$$\begin{aligned} \hat{r}(u, a, t_u) = \sum_{v \in n(u)} & (\alpha_0 + \alpha_1 b_v + \alpha_2 b_u + \alpha_3 b_a \\ & + \vec{U}\vec{A} + \vec{A'}\vec{V} + \vec{U'}\vec{V'})(1 - c(1 + \log(t_u - t_v))), \end{aligned} \quad (16)$$

where we sum up for all neighbors of u and t_v is when v last scrobbled a before t_u. For training, we only update $f(v \xrightarrow{a} u)$ by the actual positive events and a generated sample of negative events. In our algorithm we use SGD with respect to MSE to train the latent factors and the weights $\alpha_0, \ldots, \alpha_3$. Notice that the weight of the factor models is included within the factors, since the entire formula (15) is trained by a single SGD procedure. As we learn online, the weight of the effects are also trained by SGD and not by the least squares optimization procedure proposed in [4].

In an efficient implementation, since the expression (13) quickly decays with t, 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 influence with delay T not more than a predefined time frame and hence we set $c = 1/(1 + \log T)$. 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.

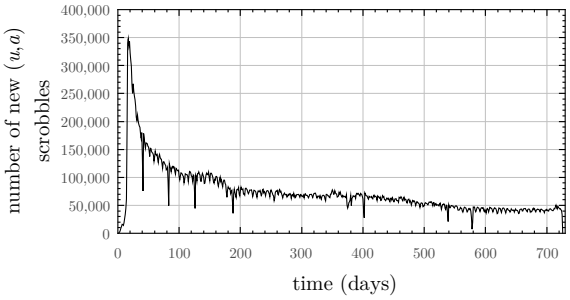

Figure 6: Number of new (u, a) scrobbles as the function of time.

5. ONLINE EVALUATION

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 [17, 24] 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.

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. 6, 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.

One possible measure for the quality of a recommended top list of length K could be precision and recall [25, 26]. 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

$$\text{DCG@K}(a) = \begin{cases} 0 & \text{if } \text{rank}(a) > K; \\ \dfrac{1}{\log_2(\text{rank}(a)+1)} & \text{otherwise.} \end{cases} \quad (17)$$

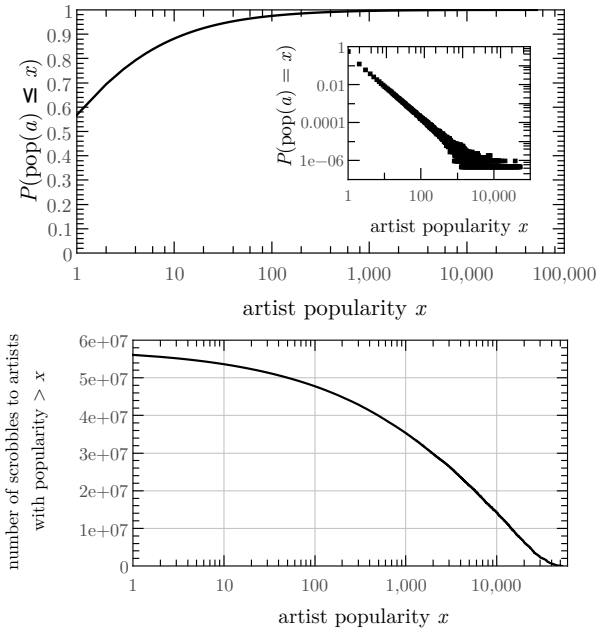

Figure 7: **Top:** Distribution of scrobble count to a given artist and the cumulative distribution. **Bottom:** Fraction of scrobbles for artists with popularity at least a given value x, as the function of x.

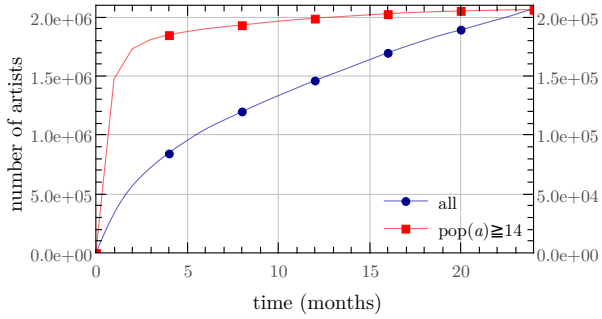

Figure 8: The number of different artists scrobbled before a given time in the two year period of the data set.

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 $K = 20$.

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. 7, top, the number of

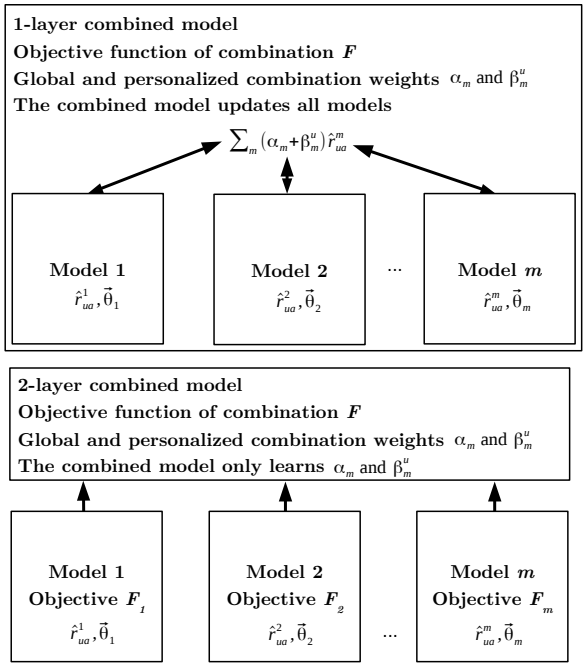

Figure 9: Scheme of the 1-layer(**top**) and 2-layer(**bottom**) online combination models.

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. 7, bottom, they attribute to only less than 10% of the data set. In other 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. 8. 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.

6. ONLINE BLENDING

We give two methods based on SGD that learn the online blending weight of recommender algorithms. Note that the algorithms may or may not themselves be based on SGD, i.e. the derivative of the individual models may or may not be available for the blending optimization procedure. Furthermore, we may blend methods with different definitions of the implicit feedback data sequence: the positive instances for the influence based recommender form a small subset of all the events and hence the influence recommender also needs different methods for generating negative training samples.

If the derivatives of the individual models are available for the top level optimizer, we may optimize in a single layer (top of Fig. 9) by minimizing

$$F(\hat{r}_{ua}) = F\left(\sum_m (\alpha_m + \beta_m^u)\hat{r}_{ua}^m\right), \tag{18}$$

where we sum over all models m, and F is the error measure, MSE in our case. Notice that we learn a user dependent blending weight vector β_m^u, hence for example the blending of a k and a k' factorization will in theory have at most as high F as a single $k+k'$ one, and in our experience performed only slightly better.

We may take the derivatives for both the constants α and β and the individual model parameters $\vec{\theta}_m$:

$$\frac{\partial F}{\partial \vec{\theta}_m} = \frac{\partial F}{\partial \hat{r}_{ua}} \frac{\partial \hat{r}_{ua}}{\partial \hat{r}_{ua}^m} \frac{\partial \hat{r}_{ua}^m}{\partial \vec{\theta}_m} = \frac{\partial F}{\partial \hat{r}_{ua}} (\alpha_m + \beta_m^u) \frac{\partial \hat{r}_{ua}^m}{\partial \vec{\theta}_m}; \tag{19}$$

$$\frac{\partial F}{\partial \alpha_m} = \frac{\partial F}{\partial \hat{r}_{ua}} \frac{\partial \hat{r}_{ua}}{\partial \alpha_m} = \frac{\partial F}{\partial \hat{r}_{ua}} \hat{r}_{ua}^m; \tag{20}$$

$$\frac{\partial F}{\partial \beta_m^u} = \frac{\partial F}{\partial \hat{r}_{ua}} \frac{\partial \hat{r}_{ua}}{\partial \beta_m^u} = \frac{\partial F}{\partial \hat{r}_{ua}} \hat{r}_{ua}^m. \tag{21}$$

If the derivatives are not available, the individual models are considered as black box for blending and we have to train in two layers (bottom of Fig. 9) and we may only use the last two derivatives (20) and (21).

If different models need different training samples, we cannot use the derivative (19) either. This is the case if we combine the baseline matrix factorization with the algorithm of Section 4. If the current positive event is not the result of an influence (i.e. not a first time scrobble or no friend scrobbling the same artist before), then we only update the baseline models. And if there is at least one possible influencer $v \xrightarrow{a} u$ for the current event (u, a), then we generate separate negative training instances for the baseline and the influence models. Notice that even a negative influence training data $v' \xrightarrow{a} u$ must satisfy that v' is a friend of u who scrobbled a and hence we usually have to choose from a restricted small set. Blending is meaningful only over this restricted set too, since for other events, the influence recommender has no t value in equation (16) to compute its prediction. Hence for blending, we have to use the same negative samples as for training the influence model.

7. MUSIC RECOMMENDATION BASELINE METHODS

We describe three baseline methods. The first one is based on dynamic popularity in Section 7.1. The second one in Section 7.2 is an online matrix factorization and the third one in Section 7.3 adds regularization over friendship as in [18].

All the methods discussed here are online algorithms, as opposed to the batch methods used in challenges such as Netflix. In some preliminary experiments the batch algorithms performed significantly worse in the online task compared to their online versions. We plan to compare the performance of batch and online versions of the algorithms in an online task more extensively in the future.

7.1 Dynamic popularity based recommendation

Given a predefined time frame T as in Section 4, at time t_u we recommend an artist based on the popularity in time not earlier than $t_u - T$ but before t_u. 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

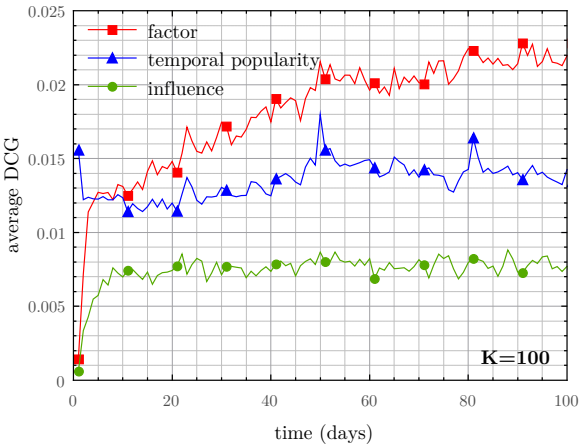

Figure 10: Online performance of the three different recommenders.

or decrement. For globally popular items, the sorted order can be maintained by a few changes in the order only.

7.2 Online matrix factorization

Stochastic gradient descent methods in batch setting 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. We may implement an online recommender algorithm by allowing a single iteration over the training data only, and this single iteration processes the events in the order of time. We used the first time scrobbles as positive training instances and generated negative training instances by selecting three random artists uniformly at the time when a user first scrobbled an artist.

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 3: 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.

7.3 Social regularization

Ma et al. [18] 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 [18] 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 3, even though they may work very well for other, non-first-time scrobbles that we do not consider in this paper.

8. EXPERIMENTS

In this section we describe the quality of our results for the second year testing period. Under various settings, we give daily average DCG@K defined by equation (17).

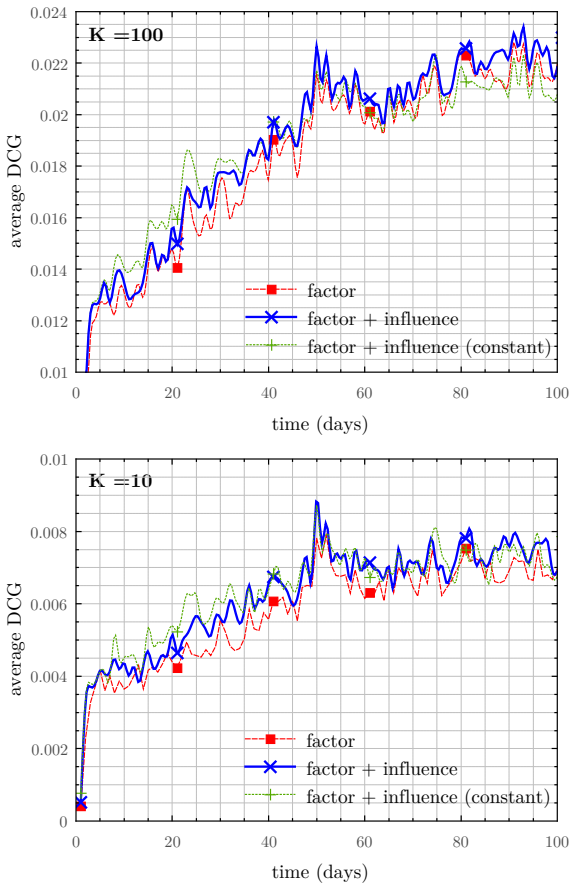

Figure 11: Combination of the influence and factor models.

Our experiments were carried out over the single core of an AMD based virtual server with 128GB RAM. On average, it took 28 minutes to process one day of scrobble history, update the online models and provide top-K recommendation corresponding to each user event.

Parameter K in equation (17) 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. We show results for $K = 10$ and 100. In Fig. 10, DCG@100 is shown for two baseline methods, matrix factorization and temporal popularity, as well as our influence model.

When combining variants of baseline and influence recommendation predictions, we observed that that social regularization did not improve matrix factorization and temporal popularity did not blend with online factorization. Indeed in Fig. 10 we may observe that peaks in temporal popularity performance immediately appear as peaks in matrix factorization performance, since online factorization learns temporal trends very well.

In Fig. 11, one can see that the online combination with influence recommendation improves over online matrix factorization both for DCG@10 and DCG@100. The average improvement is roughly 7% for DCG@10, and 3% for DCG@100. Over the same figure, we plot the performance of the constant term alone in equation (15). This simple recommender corresponds to adding up all the $(1 - c(1 + \log t))$ values for possible influencers without model building be-

yond learning the blending weight involved. At first this simple model blends best with the baseline, however, as the factor models get more training data, they become superior and the importance of the constant term α_0 in the model diminishes.

Conclusions

Based on a 70,000-entry sample of Last.fm users, we were able to exploit the effect of users influencing the taste of friends for improving the quality of music recommendation. Over static baseline recommenders, we achieved a 5% improvement in recommendation accuracy when presenting artists from friends' past scrobbles that the given user had never seen before.

Our system has very strong time-awareness: when we recommend, we look back in the near past and combine friends' scrobbles with the baseline methods. 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.

All of our methods learn online and provide top-K recommendation lists recomputed for every user query. Because of the inherent time dependence, we defined average DCG as our evaluation metric and gave a new online blending procedure that learns online user-dependent weights.

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.

The publication was supported by the KTIA_AIK_12-1-2013-0037 and the PIAC_13-1-2013-0205 projects. The projects are supported by Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund. Work conducted at the Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI) was supported in part by the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956), by the Momentum Grant of the Hungarian Academy of Sciences, and by OTKA NK 105645. Work conducted at the Technical University Budapest 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 TÁMOP - 4.2.2.B-10/1–2010-0009. Work conducted at University of Szeged 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).

9. REFERENCES

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma. Online collaborative filtering. *University of California at Berkeley, Tech. Rep*, 2007.

[2] A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proc. 30th SIGIR*, pages 773–774. ACM, 2007.

[3] E. Bakshy, J. M. H., W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proc. 4th WSDM*, pages 65–74. ACM, 2011.

[4] R. Bell and Y. Koren. Improved Neighborhood-based Collaborative Filtering. *KDD-Cup and Workshop*, 2007.

[5] R. Bell and Y. Koren. Lessons from the Netflix prize challenge. 2007.

[6] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD*, 2007.

[7] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16, 2011.

[8] M. Cha, A. Mislove, B. Adams, and K. P. Gummadi. Characterizing social cascades in flickr. In *Proc. first workshop on Online social networks*, pages 13–18. ACM, 2008.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. 4th RecSys*, pages 39–46. ACM, 2010.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM TOIS*, 22(1):143–177, 2004.

[11] P. Domingos and M. Richardson. Mining the network value of customers. In *Proc. 7th SIGKDD*, pages 57–66. ACM, 2001.

[12] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. *Advances in neural information processing systems*, 20:385–392, 2007.

[13] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *Proc. 3rd WSDM*, pages 241–250. ACM, 2010.

[14] X. Hu, M. Bay, and J. Downie. Creating a simplified music mood classification ground-truth set. In *Proc. 8th ISMIR*, 2007.

[15] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. *PKDD*, pages 506–514, 2007.

[16] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *Proc. 30th SIGIR*, pages 447–454. ACM, 2007.

[17] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. 14th SIGKDD*, pages 426–434. ACM, 2008.

[18] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proc. 4th WSDM*, pages 287–296. ACM, 2011.

[19] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *18th WWW*, pages 641–641. ACM, 2009.

[20] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *Proc. 17th Hypertext and Hypermedia*, pages 31–40. ACM, 2006.

[21] J. Noel, S. Sanner, K.-N. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna. New objective functions for social collaborative filtering. In *Proc. 21st WWW*, pages 859–868. ACM, 2012.

[22] R. Pálovics and A. A. Benczúr. Temporal influence over the last. fm social network. In *Proc. ASONAM*, pages 486–493. ACM, 2013.

[23] K. Tso-Sutter, L. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proc. Symp. on Applied Computing*, pages 1995–1999. ACM, 2008.

[24] M. Weimer, A. Karatzoglou, and A. Smola. Adaptive collaborative filtering. In *Proc. 2nd RecSys*, pages 275–282. ACM, 2008.

[25] X. Yang, H. Steck, Y. Guo, and Y. Liu. On top-k recommendation using social networks. In *Proc. 6th RecSys*, pages 67–74. ACM, 2012.

[26] Q. Yuan, L. Chen, and S. Zhao. Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In *Proc. 5th RecSys*, pages 245–252. ACM, 2011.