# 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

*Abstract*—**Several recent results show the influence of social contacts to spread certain properties over the network, but others question the methodology of these experiments by proposing that the measured effects may be due to homophily or a shared environment. In this paper we justify the existence of the social influence by considering the temporal behavior of Last.fm users. In order to clearly distinguish between friends sharing the same interest, especially since Last.fm recommends friends based on similarity of taste, we separated the timeless effect of similar taste from the temporal impulses of immediately listening to the same artist after a friend. We measured strong increase of listening to a completely new artist in a few hours period after a friend compared to non-friends representing a simple trend or external influence. In our experiment to eliminate network independent elements of taste, we improved collaborative filtering and trend based methods by blending with simple time aware recommendations based on the influence of friends. Our experiments are carried over the two-year "scrobble" history of 70,000 Last.fm users.**

## I. INTRODUCTION

Several results show the influence of friends and contacts to spread obesity [1], loneliness [2], alcohol consumption [3], religious belief [4] and many similar properties in social networks. Others question the methodology of these experiments [5] by proposing that the measured effects may be due to homophily, the fact that people tend to associate with others like themselves, and a shared environment also called confounding or contextual influence.

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. Companies such as Last.fm use this data to organize and recommend music to people.

In this paper we exploit the timely information gathered by the Last.fm service on users with public profile to investigate how members of the social network may influence their friends' taste. 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.

Our contribution to the dispute on whether social contacts influence one another or whether the observed similarity in taste and behavior is only due to homophily, we show a carefully designed experiment to 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. Another possible source for users listening to the same music may come from traditional media: news, album releases, concerts and ads. While the sources are hard to identify, common in them is that they cause temporal increase in popularity for the targeted artist. These effects are filtered by another method that measures popularity at the given time and recommends based on the momentary popularity.

We blend collaborative filtering and temporal popularity recommenders with a method for influence prediction that we describe in this paper. We consider events where a user listens to an artist for the first time closely after a friend listened to the same artist. We obtain a 4% of increase in recommendation quality, a strong result in view of the three-year Netflix Prize competition [6] to improve recommender quality by 10%. Note that we only give a single method that results in a stable strong improvement over the baselines.

Our new method is a lightweight recommender based on friends' past items that can be very efficiently computed even in real time. Part of the efficiency comes from the fact that potential items from influencing friends are relative rare. For this reason, the method in itself performs worse than the baselines, however it combines very well with them. Indeed, influence based predictions improve the accuracy of a traditional factor model recommender by nearly as much as measuring popularity at the given time, a prediction that is strong in itself. The fact that influences bend well prove that close events in the network bring in new information that can be exploited in a recommender system and also prove the existence of influence from friends beyond homophily.

## A. 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 [7]. The final best results blended 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 [8] that is based on an approach reminiscent of gradient boosting [9].

Closest to our results are the applications of network influence in collaborative filtering [10]. However in their data only ratings and no social contacts are given. In another result [11] over Flickr, both friendship and view information was present, but the main goal was to measure the strength of the influence and no measurements were designed to separate influence from other effects.

Bonchi [12] 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.

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 [13]. For a recent result on evaluating top-$k$ recommenders is found in [14].

Music recommendation is considered in several results orthogonal to our methods that will likely combine well. Mood data set is created in [15]. Similarity search based on audio is given in [16]. Tag based music recommenders [17], [18, 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 [19], [20], [21]. Note that the Netflix Prize competition put a strong vote towards the second class of methods [22].

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 social networks [23]. Deep analysis of influence in terms of retweets and mentions is given in [24]. Notion of influence similar to ours is derived in [25], [26] for Fickr 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.

## II. THE LAST.FM DATA SET

Last.fm became a relevant online service in music based social networking. The idea of Last.fm is to create a recommendation system based on plugins nearly for all kind of music listening platforms. 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.
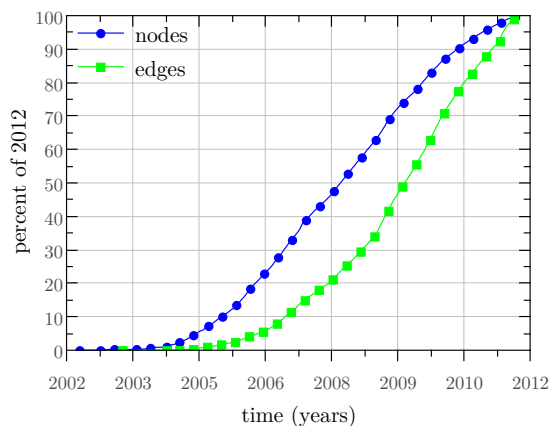
Fig. 1. The number of the users and friendship edges in time as the fraction of the values at the time of the data set creation (2012).
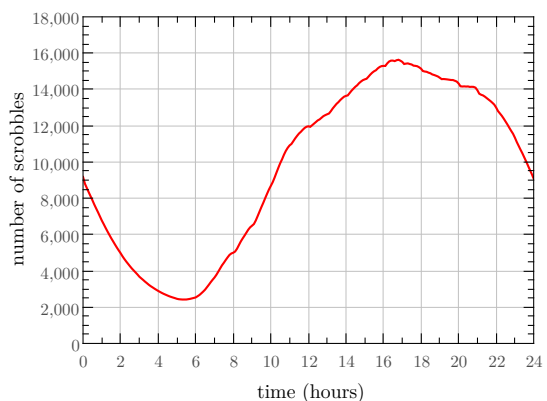


Fig. 2. Daily periodicity of scrobble count.

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 focus on two types of user information,

- the timeline information of users: user $u$ "scrobbled" artist $a$ at time $t$ $(u, a, t)$,
- and the social network of users.

Our data set hence consists of the contacts and the musical taste of the users. Our goal is to justify the existence of the influence of social contacts, i.e. certain correlation the taste of friends in the user network. 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.
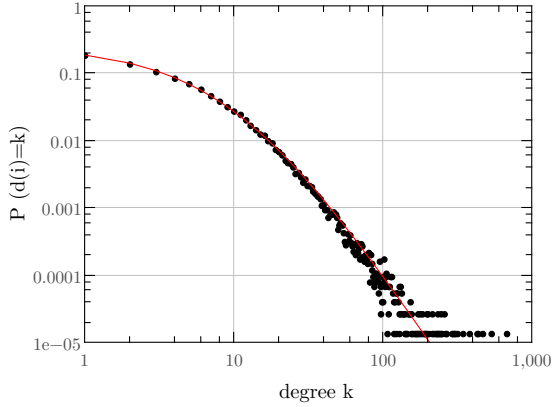- At least 10 friends that meet the first four conditions.

Fig. 3. Degree distribution in the friendship network.



Fig. 4. Potential influence on $u$ by some other user to scrobble $(u, a, t)$.

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). Note that no edges are ever deleted from the network.

The number of users both in the time series and in the network is 71,000 with 285,241 edges. The average degree is therefore 8, while the degree distribution follows shifted power-law as seen in Fig. 3

$$P(d(i) = k) \sim (x + s)^{-\alpha}$$

with exponent 3.8.

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. Fig. 2 shows the daily fluctuations in the users scrobbling activity.

## III. NOTION OF NETWORK INFLUENCE

The key concept in this paper is a user $v$ *influencing* another $u$ to scrobble $a$. This happens if $u$ 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* of the influence, as seen in Fig. 4. Our key assumption is that, in the above definition, we observe influences between non-friends only by coincidence while some of the observed influence between friends is the result of certain interaction between them. 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 [11], [25], [26]. As detailed in [26], 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 $a$.

For smaller influence delay $\Delta t$, we are more certain that $u$ is affected by the previous scrobble of $v$. The distribution of dela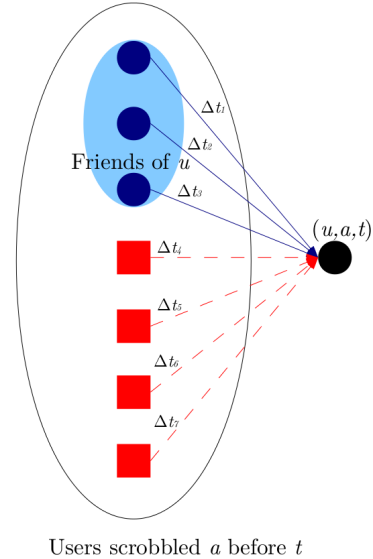y with respect to friends and non-friends will help us in determining the frequency and strength of influence over the Last.fm social network. Each time user $u$ first scrobbles $a$, we compute the delay $\Delta t$ for all users $v$ who scrobbled $a$ before $u$, if such users exist (see Fig. 4).

Out of the 57,274,158 first-time scrobbles of certain artist $a$ by some user, we find a friend who scrobbled $a$ before 10,993,042 times (19%). Note that one user can be influenced by more friends therefore the total number of influences is 24,204,977. 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 influences may result by pure co-incidence, especially when a new album is released or the popularity of the artist increases for some other reason. In order to identify real influence, we compare the frequency of influence from friends and from non-friends along delay $\Delta t$ as parameter. We compute the cumulative distribution function of all influences as a function of the delay,

$$\text{CDF}_A(t) = \text{fraction of influences with delay } \Delta t \leq t$$
$$\text{among all influences.} \tag{1}$$

Similarly, $\text{CDF}_F(t)$ stands for the same function among influences between friends only. Fig. 5 shows the functions for all users and friends. The function of friends is above that of all users, i.e. we observe shorter delay more frequently among friends.

Next we quantify the importance of friendship in influencing others as the *effectivity* function. The effectivity at $\Delta t$ is defined as the increase of influenced scrobbles among friends relative to all users that happen with delay at most $t$:

$$\text{Eff}(t) = \frac{\text{CDF}_F(t) - \text{CDF}_A(t)}{\text{CDF}_F(t)}. \tag{2}$$
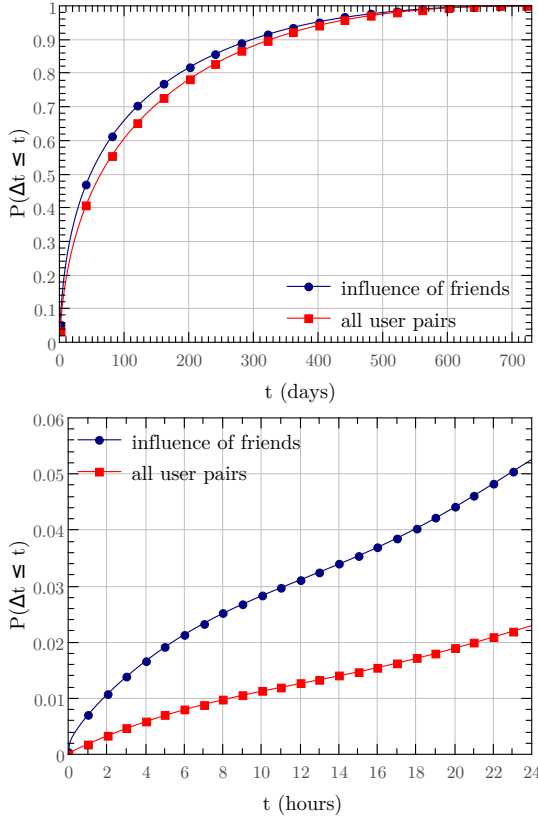
Fig. 5. Fraction of influences with delay $\Delta t \leq t$ as the function of $t$ as in (1), in case of friends (CDF$_F$) and non-friends (CDF$_A$) over the entire timeline (**top**) and the first 24 hours (**bottom**).
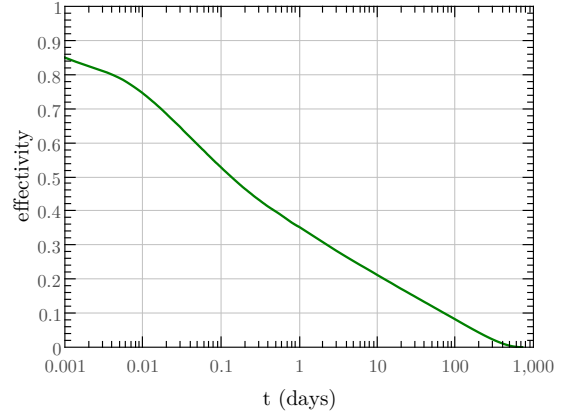


Fig. 6. The measured effectivity of the influence (ratio of increase among friends compared to all users) as in (2) very closely follows a logarithmic function of delay $\Delta t$.
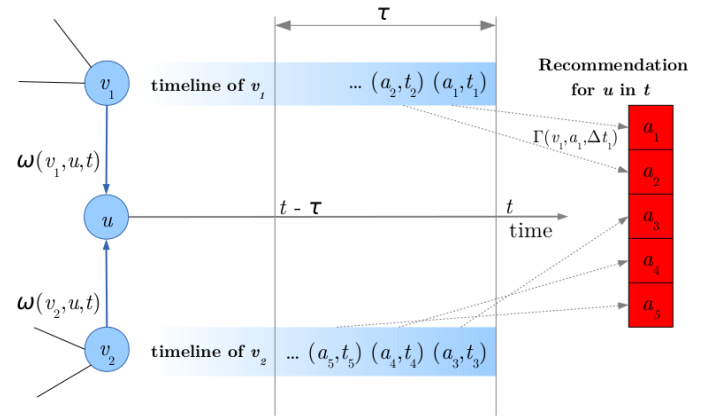


Fig. 7. Scheme of the influence based recommender algorithm.

Fig. 6 shows the measured effectivity curve in the community. As expected, Eff$(t)$ is a monotonically decreasing function of $t$. However, the decrease is slow unlike in some recent influence models that propose exponential decay in time [11]. Therefore, we approximate Eff$(t)$ with a slowly decreasing logarithmic function instead of an exponential decay.

## IV. INFLUENCE BASED RECOMMENDATION

Next we use our notion of influence in the task of artist recommendation. Influence depends on time and no matter how relative slow but the effectivity of a friend scrobbling an artist decays. For this reason the influence based recommendation must be updated more frequently than traditional collaborative filtering methods. Also note that for a given user, our recommendation can be computed very efficiently by a pass over the recent history of friends.

Based on the measurements in the previous Section, we give a temporal network influence based recommender algorithm. For a user $u$ at time $t$, we recommend based on friends' scrobbles before $t$. The predicted score $\hat{r}(u, a, t)$ of an artist $a$ is based on a function $\Gamma$ of the time elapsed since the friend $v$ scrobbling $a$ (the delay $\Delta t$) and a function $\omega$ of the observed frequency of $v$ influencing $u$ in the past, as summarized in

Fig. 7. Formally the predicted rating becomes

$$\hat{r}(u, a, t) = \sum_{v \in n(u)} \Gamma(v, a, \Delta t) \omega(v, u, t), \qquad (3)$$

where $n(u)$ denotes the friends of $u$, $\omega(v, u, t)$ is the strength of the influence between users $u$ and $v$, and $\Gamma(v, a, \Delta t)$ is the weight between user $v$ and artist $a$ based on the delay.

Our implementation depends on the two functions $\omega$ and $\Gamma$ defined in the next two subsections. In an efficient algorithm, the value of $\omega$ can be stored in memory for all pairs of friends. Alternately, $\omega$ can only be batch updated as the strength between two users are less time sensitive. The values of $\Gamma$, however, depend on the actual time when the recommendation is requested. As $\Gamma$ quickly decays with $\Delta t$, we only need to retrieve the past srobbles of all $v$, the friends of $u$. This step can be efficiently implemented unless $u$ has too many friends. In this latter case we could select only a few influencing friends based on the values of $\omega$, otherwise the recommendation is noisy anyway. Our algorithm can hence be implemented even in real time.

## A. Influence as function of delay

The potential of influence decays as time elapses since the influencer $v$ scrobbled the given artist $a$. Based on the effectivity curve (see Fig. 6) we approximate the strength of the influence with a monotonically decreasing logarithmic function

$$\Gamma(v, a, \Delta t) = 1 - C \cdot \log(\Delta t), \qquad (4)$$

where C is a global constant.

## B. Strength of influence between user pairs

We recommend a recent scrobble by a friend by taking both the recency of the scrobble and the observed relation between the two users. For each pair of users $u$, the influenced and $v$, the influencer, we define the strength $\omega(v, u, t)$ as a step function in time as follows:

- We initialize $\omega(v, u, 0) = 0$ for all pairs.
- Assume that $u$ and $v$ become friends at time $t_0$. We take a step and set $\omega(u, v, t_0) = \omega(v, u, t_0) = 1$.
- If we observe an influence from $v$ to $u$ at time $t > t_0$ with time difference $\Delta t$, we take another step and increase $\omega(v, u, t)$ by

$$\omega(v, u, t) \leftarrow \omega(v, u, t) + (1 - C \cdot \log(\Delta t)), \quad (5)$$

where $C$ is a global constant. For simplicity we use the same logarithmic function of the delay as in (4).

To speed up computations, we only consider influence with delay not more than a predefined time frame $\tau$. We apply $\tau$ for defining both $\omega$ in (5) and $\Gamma$ in (4) and hence in both cases we set

$$C = 1/\log(\tau). \qquad (6)$$

## V. REAL TIME RECOMMENDATION EVALUATION

Recommender systems in practice need to rank the best $k$ items for the user in real time. In the so-called top-$k$ recommendation task [13], [14], potentially we have to compute a new top list for every single scrobble in the test period. The top-$k$ task is different from the standard recommender evaluation settings and needs carefully selected 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 by computing the discounted cumulative gain with treshold $K$

$$\text{DCG@K}(a) = \begin{cases} 0 & \text{if rank } (a) > K; \\ \dfrac{1}{\log_2(\text{rank}(a) + 1)} & \text{otherwise.} \end{cases}$$
(7)

Note that in this unusual setting there is a single relevant item and hence for example no normalization is needed as in case of the NDCG measure. Also note that the DCG values will be small since the NDCG of a relative short
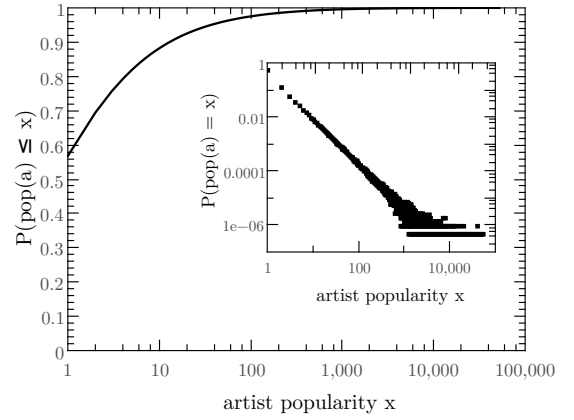


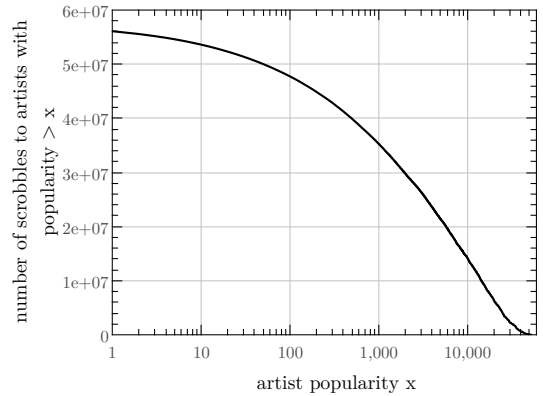Fig. 8. Distribution of scrobble count to a given artist and the cumulative distribution.



Fig. 9. Fraction of scrobbles for artists with popularity at least a given value $x$, as the function of $x$.

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. 8, 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. 9, they attribute to only less than 10% of the data set. In other words by discarding a large number of artists, we only loose 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. 10. This
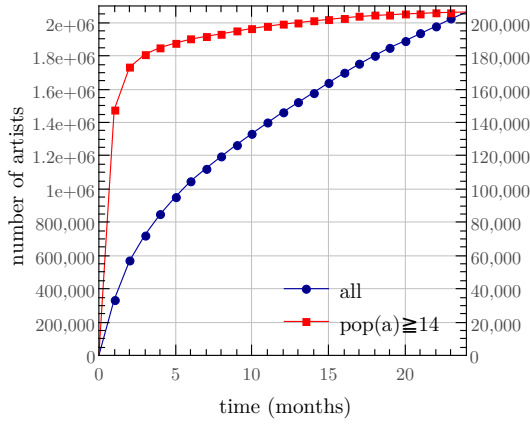
Fig. 10. The number of different artists scrobbled before a given time in the two year period of the data set.

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.

## VI. MUSIC RECOMMENDATION BASELINE METHODS

We describe one baseline method based on dynamic popularity in Section VI-A and one based on factorization in Section VI-B.

### A. Dynamic popularity based recommendation

Given a predefined time frame $\tau$ as in Section IV, at time $t$ we recommend an artist based on the popularity in time not earlier than $t - \tau$ 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. As future work we could also consider bursts and predict the popularity increase or decrease.

### B. Factor model based recommendation

For our factor model based recommender we selected the implementation of Funk [8]. 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.
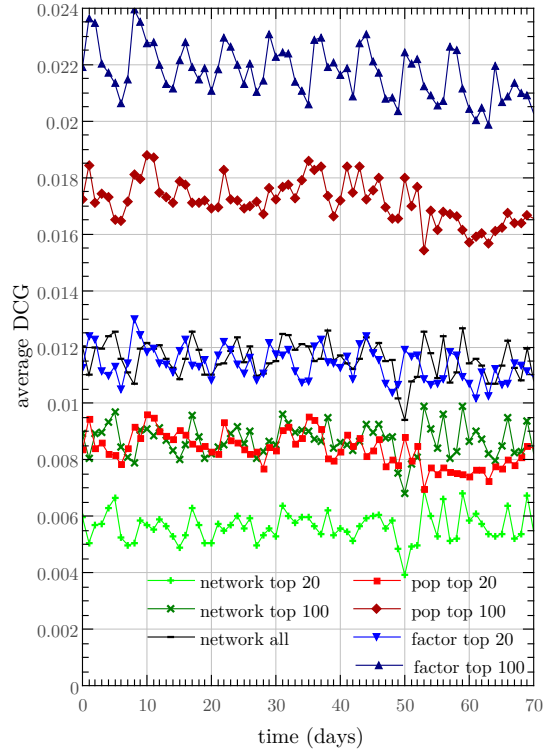


Fig. 11. Daily average DCG@K as in (7) in a 70-day sample of the test period. We show the three basic methods, from strongest to weakest, the factorization, temporal popularity, and the network influence recommenders. For $K$ we measure two values, 20 and 100, except for network influence where we also show $K = \infty$ as the entire ranked list can be efficiently computed in this case.
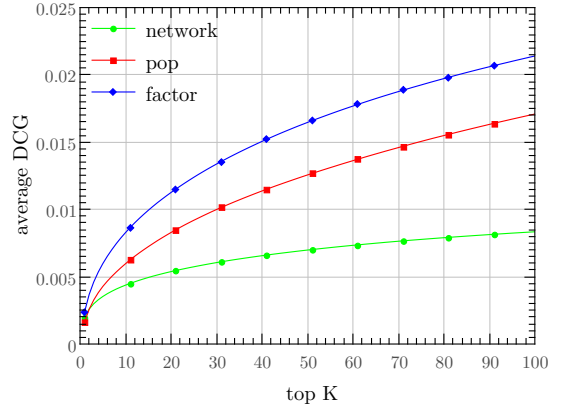


Fig. 12. DCG@K as the function of $K$ for the three basic algorithms, for a time window $\tau$ equal to one week.

## VII. EXPERIMENTS

First we give the daily average DCG@K defined by equation (7) in the second year testing period for the influence based and the two baseline recommenders. Parameter $K$ in equation (7) controls the length of the top list considered for evaluation.
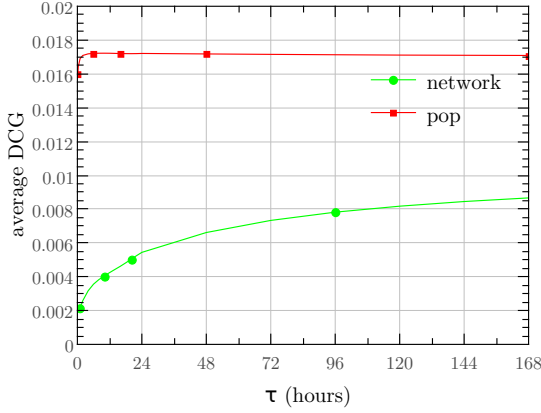
Fig. 13. DCG@100 defined by equation (7) as the function of the time window threshold $\tau$ as in Section IV-A.

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 performance of the three basic methods is shown in Fig. 11 for $K = 20$ and 100 and a time window $\tau$ in Section IV-A equal to one week.

The dependence on the top list size $K$ is measured in Fig. 12 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 blending results not just for the value $K = 20$ that we consider practically feasible but also for 100 for comparison.

Next we investigate the parameters of the individual algorithms. For a matrix factorization based method we use Funk's algorithm [8] with the following parameters that turned out to perform best in our experiments: learning rate = 0.001, feature number = 20, and initial feature value = 0.1. We re-train the algorithm each week based on all past data. For this reason we see weekly periodicity in the 10-week timeline of Fig. 11: the factor model performs best immediately after the training period and slowly degrades in the testing period.

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 $\tau$ is the maximum delay $\Delta t$ that we consider as influence while for the popularity one $\tau$ is the time interval that we use for frequency computation. We ran measurements in the second year test period with different time frames $\tau$ and computed the average DCG performance of the recommender systems. Figure 13 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 $\tau$ to be one week.

The final conclusion of the experiments is drawn by blending the three recommenders as shown in Figs. 15–14. In our experiments we obtained the best results by linearly combining 1/rank instead of the predicted score. As an advantage of 1/rank, we need no score normalization.
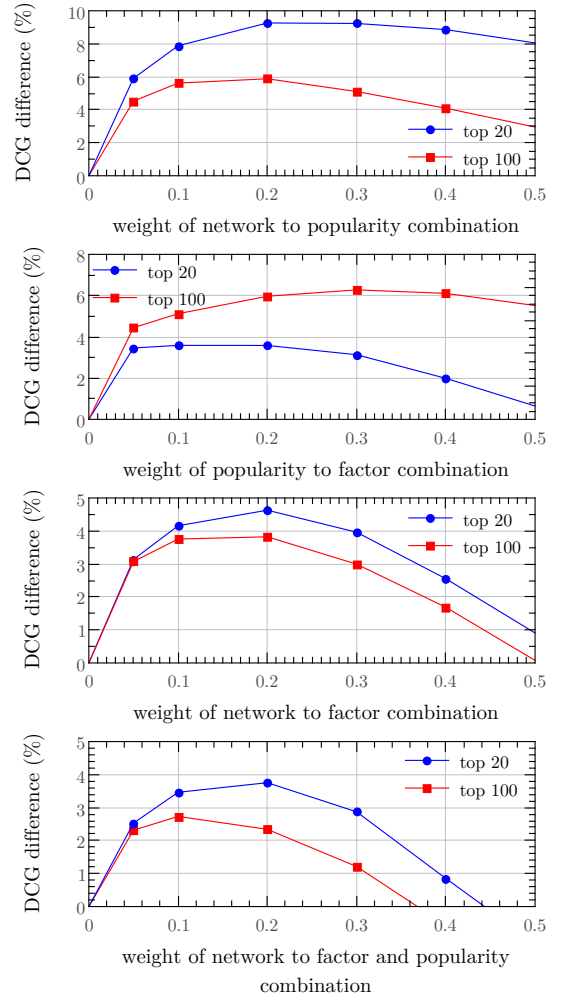


Fig. 14. Blending DCG@K defined by (7) as the function of the linear combination weight. From top to bottom: network influence and factor model; temporal popularity and factor model; network influence and factor model; finally network influence and the strogest combination of factor with popularity.

Figure 14 shows the relative improvement of the recommenders as the function of the blending weights. After blending the recommenders pairwise, we selected the strongest popularity-factor combinations (3:7 and 2:8) and blended it with the network recommender. One can see that the influence recommender not only improves the results of the factor and popularity recommenders, but combines well with their best blended result: the combination of the three methods outperforms the best blend of the factor and popularity models both for DCG@20 and DCG@100. The improvement is roughly 4%. Figure 15 shows the monthly average DCG@20 and DCG@100 curves in the testing period in case of the different blended recommenders. Each curve shows the result of the best combination of the corresponding recommenders. In each case we observe stable improvement over the entire testing period.
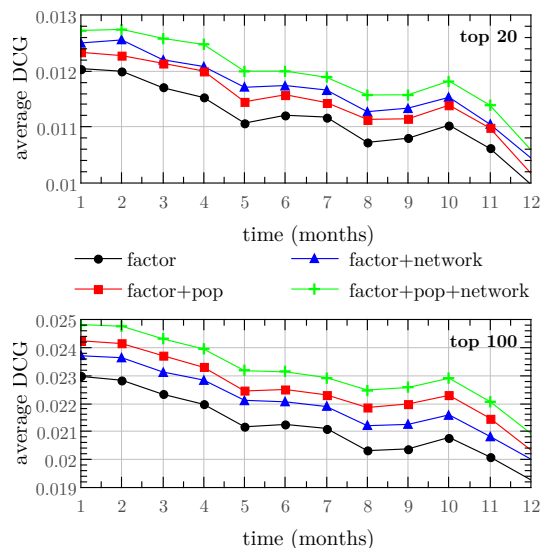
Fig. 15. Monthly average DCG@20 (**top**) and DCG@100 (**bottom**) as defined by (7) in the test period for the factor model and its combinations.

## Conclusions

Based on a 70,000 sample of Last.fm users, we were able to measure the effect of certain user recommending an artist to her friends. Our results confirm the existence of influence through the social network as opposed to the pure similarity of taste between friends. We disproved the opinion that homophily could be the reason for friends listening to the same music or behave similarly by constructing a baseline that takes homophily and temporal effects into account. Over the baseline recommender, we achieved a 4% 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. In addition, our method can efficiently be computed even in real time.

For future work we plan to investigate whether the temporal social influence is specific to Last.fm dataset or can match to other kind of social network, e.g. Twitter. We also plan to break down the analysis of influence spread by type of music, by age range, or by artist.

## Acknowledgements

## References

[1] N. Christakis and J. Fowler, "The spread of obesity in a large social network over 32 years," *New England Journal of Medicine*, 357(4):370–379, 2007.

[2] J. Cacioppo, J. Fowler, and N. Christakis, "Alone in the crowd: The structure and spread of loneliness in a large social network.," *Journal of Personality and Social Psychology*, vol. 97, no. 6, p. 977, 2009.

[3] J. Rosenquist, J. Murabito, J. Fowler, and N. Christakis, "The spread of alcohol consumption behavior in a large social network," *Annals of Internal Medicine*, vol. 152, no. 7, p. 426, 2010.

[4] S. Stroope, "Social networks and religion: The role of congregational social embeddedness in religious belief and practice," *Sociology of Religion*, 2011.

[5] R. Lyons, "The spread of evidence-poor medicine via flawed social-network analysis," *Statistics, Politics, and Policy*, 2(1), p. 2, 2011.

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

[7] R. Bell and Y. Koren, "Lessons from the Netflix prize challenge," 2007.

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

[9] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, 29(5):1189–1232, 2001.

[10] P. Domingos and M. Richardson, "Mining the network value of customers," in *SIGKDD*, pp. 57–66, ACM, 2001.

[11] A. Goyal, F. Bonchi, and L. V. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, pp. 241–250, ACM, 2010.

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

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

[14] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *RecSys*, pp. 39–46, ACM, 2010.

[15] X. Hu, M. Bay, and J. Downie, "Creating a simplified music mood classification ground-truth set," in *ISMIR*, 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 SIGIR*, pp. 447–454, ACM, 2007.

[17] 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.

[18] K. Tso-Sutter, L. Marinho, and L. Schmidt-Thieme, "Tag-aware recommender systems by fusion of collaborative filtering algorithms," in *ACM symposium on Applied Computing*, pp. 1995–1999, ACM, 2008.

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

[20] C. Marlow, M. Naaman, D. Boyd, and M. Davis, "Ht06, tagging paper, taxonomy, flickr, academic article, to read," in *Conf. on Hypertext and Hypermedia*, pp. 31–40, ACM, 2006.

[21] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, "Evaluating similarity measures for emergent semantics of social tagging," in *WWW*, pp. 641–641, 2009.

[22] I. Pilászy and D. Tikk, "Recommending new movies: even a few ratings are more valuable than metadata," in *RecSys*, pp. 93–100, ACM, 2009.

[23] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?," in *WWW*, pp. 591–600, ACM, 2010.

[24] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi, "Measuring user influence in Twitter: The million follower fallacy," in *ICWSM*, 2010.

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

[26] E. Bakshy, J. M. H., W. A. Mason, and D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," in *WSDM*, pp. 65–74, ACM, 2011.