PROJECT PERIODIC REPORT

Grant Agreement number: 288956

Project acronym: NADINE

Project title: New tools and Algorithms for Directed Network analysis

Funding Scheme: Small or medium-scale focused research project (STREP)

Periodic report: **1st 2nd X**

Period covered: **from 1.11.2013 to 30.04.2015**

Name, title and organisation of the scientific representative of the project's coordinator[1]:

Dr. Dima Shepelyansky

Directeur de recherche au CNRS

Lab de Phys. Theorique, Universite Paul Sabatier, 31062 Toulouse, France

Tel: +331 5 61556068, Fax: +33 5 61556065, Secr.: +33 5 61557572

E-mail: dima@irsamc.ups-tlse.fr; URL: www.quantware.ups-tlse.fr/dima

Project website address: www.quantware.ups-tlse.fr/FETNADINE/

[1] Usually the contact person of the coordinator as specified in Art. 8.1. of the grant agreement

NADINE DELIVERABLE D3.2.

It is based on milestones M5* [WP3.1,WP3.2] (promised to be finished), M12 [WP3.3, WP3.4] with deliverable publications:

[41] P1.21 Young-Ho Eom and D.L.Shepelyansky, **"Opinion formation driven by PageRank node influence on directed networks"**, submitted to Physica A Feb (2015) (arXiv:1502.02567[physics.soc-ph]) [M12-WP3.3,WP3.4]

[53] P3.9 R.Palovics, F. Ayala-Gomez, B. Csikota, B.Daroczy, L. Kocsis, D. Spadacene, A.A. Benczur, **"RecSys Challenge 2014: an ensemble of binary classifiers and matrix factorization"**, Proceedings of the 2014 Recommender Systems Challenge (p. 13) ACM (2014) [M12-WP3.3,WP3.4]

[54] P3.10 Andrea N. Ban, Levente Kocsis, Robert Palovics, **"Peer-to-peer Online Collaborative Filtering"**, preprint (2015) [M12-WP3.3,WP3.4]

[60] P3.16 Robert Palovics, Balint Daroczym Andras A. Benczur, Julia Pap, Leonardo Ermann, Samuel Phan, Alexei D. Chepelianskii, Dima L. Shepelyansky, , **"Statistical analysis of NOMAO customer votes for spots of France"**, preprint arXiv (2015) [M12-WP3.3,WP3.4]

[61] P3.17 Robert Palovics, Ferenc Beres, Nelly Litvak, Frederick Ayala-Gomez and Andras A. Benczur, **"Centrality prediction in temporally evolving networks"**, preprint arXiv (2015) [M12-WP3.3,WP3.4]

*[71] P4.18 Paolo Boldi, Corrado Monti, Massimo Santini, and Sebastiano Vigna, **"Liquid FM: Recommending Music through Viscous Democracy"**, submitted to CoRR (2015) (arXiv:1503.08604[cs.SI], 2015) [M12-WP3.3,WP3.4] **AND [M5-WP3.1-WP3.2 promised to be finished in report; the software is open and available herehttps://github.com/corradomonti/fbvoting]**

[72] P4.19 Paolo Boldi and Corrado Monti, "LlamaFur: Learning Latent Category Matrix

to Find Unexpected Relations in Wikipedia", submitted to CoRR (2015) [M12-WP3.3,WP3.4]

Opinion formation driven by PageRank node influence on directed networks

Young-Ho Eom[a,b], Dima L. Shepelyansky[b]

[a] *IMT Institute for Advanced Studies Lucca, Piazza San Francesco 19, Lucca 55100, Italy*
[b] *Laboratoire de Physique Théorique du CNRS, IRSAMC, Université de Toulouse, UPS, F-31062 Toulouse, France*

Abstract

We study a two states opinion formation model driven by PageRank node influence and report an extensive numerical study on how PageRank affects collective opinion formations in large-scale empirical directed networks. In our model the opinion of a node can be updated by the sum of its neighbor nodes' opinions weighted by the node influence of the neighbor nodes at each step. We consider PageRank probability and its sublinear power as node influence measures and investigate evolution of opinion under various conditions. First, we observe that all networks reach steady state opinion after a certain relaxation time. This time scale is decreasing with the heterogeneity of node influence in the networks. Second, we find that our model shows consensus and non-consensus behavior in steady state depending on types of networks: Web graph, citation network of physics articles, and LiveJournal social network show non-consensus behavior while Wikipedia article network shows consensus behavior. Third, we find that a more heterogeneous influence distribution leads to a more uniform opinion state in the cases of Web graph, Wikipedia, and Livejournal. However, the opposite behavior is observed in the citation network. Finally we identify that a small number of influential nodes can impose their own opinion on significant fraction of other nodes in all considered networks. Our study shows that the effects of heterogeneity of node influence on opinion formation can be significant and suggests further investigations on the interplay between node influence and collective opinion in networks.

Keywords: Opinion formation, Directed networks, Centrality, PageRank, Node influence

1. Introduction

Each individual has her/his own opinion about political, social, and economical issues based on her/his own belief, information, and perspective. Individuals also exchange, discuss, and reconcile their opinions with others through social contacts or networks. Through these interactions, collective opinions emerge from our society. The recent advent of social media such as Twitter or Facebook accelerates the emergence of collective opinions on global scale. Understanding how collective opinions are formed on various types of social networks has critical importance in the era of information technology.

Statistical physics community has provided quantitative tools to reveal the underlying mechanisms that govern the collective opinion formation through social interactions [1]. Various opinion formation models (see Refs. [1, 2] for details) on networks including voter models [3, 4, 5, 6], majority rule model [7], bounded confidence model [8], and Sznajd model [9] were suggested and extensively studied. These models have given us analysis tools of how network structure affects opinion dynamics and have provided us mathematical understanding of collective opinion formation.

In order to expand our understanding of collective opinion formation on networks further we can consider the following two directions. First we can consider opinion formation on real social networks rather than on artifact network models such as regular lattices or small-world networks which are mainly considered in previous studies [1, 2] and far from real networks. Second, in most of real situations, there are opinion leaders or elites who have strong influence and lead collective opinions in social systems. The roles of these leaders or elites on opinion formation is still elusive. In short, it is necessary to understand how heterogeneous individual influence affects on collective opinion formation on real networks.

In a recent study [10], PageRank is proposed as a node influence measure in an opinion formation model on large-scale real networks such as Web graphs and social media including LiveJournal and Twitter. The PageRank opinion formation (PROF) model, introduced in [10], takes into account a node influence in the pro-

cess of opinion formation. In the PROF model, the opinion of a node is updated by the weighted sum of neighbor nodes' opinions and the weight of the neighbor nodes are given by their PageRank (see the next section for details). It is found that a group of top influential elites in the networks (i.e., nodes with high PageRank) can impose their own opinion on a significant fraction of the considered networks [10]. The PROF model is also considered on Ulam networks [11], generated by the intermittency map and the Chirikov typical map, showing a similar behavior with the case of World Wide Web (WWW).

In the present work we consider how heterogeneous node influence affects the collective opinion formation using the modified PageRank opinion formation (PROF) model to go beyond previous works [10, 11]. Our goal is to examine how the PROF model behaves on real directed networks if we adjust the heterogeneity of node influence (i.e., the PageRank of nodes). The original PROF model considered only linear case of PageRank as a node influence, it is necessary to consider opinion formation driven by node influence under more general conditions. To do this we modified the PROF model considering sublinear PageRank of nodes such that the influence of node i is given by $P_i{}^g$ where P_i is the PageRank of node i and $0 \leq g \leq 1$. Extensive numerical study of the model shows various features of considered opinion formation. First we observed that all networks reach a steady state opinion and the relaxation time to this state is decreasing with the heterogeneity of node influence in the networks. Second we found our model shows consensus and non-consensus behavior in steady state depending on types of networks: Web graph, citation network of physics articles, and LiveJournal social network show non-consensus behavior while Wikipedia article network shows consensus behavior. Third we found that the more heterogeneous distribution of node influence the network has (i.e., higher g), the more uniform opinion state we can observe in Web graph, Wikipedia, and Livejournal. However, in the citation network, the more heterogeneous distribution of node influence leads to the less uniform opinion. Finally we observed that a small number of influential nodes can impose their own opinion on significant fraction of other nodes in all considered networks.

The paper is organized as follows. The modified PROF model is described in Section 2. The description of considered empirical directed networks is given in Section 3. The extensive numerical studies on empirical networks are presented in Section 4. A discussion of the result is given in Section 5.

2. Opinion formation by the modified PROF model

We consider a directed network $G(N, L)$ with N nodes and nodes in the network are connected by L directed links. Based on the network structure, the PageRank probability $P_i(t)$ of node i at iteration time t is given by

$$P_i(t) = (1-\alpha)/N + \alpha \sum_j A_{ij} P_j(t-1)/k_{out}(j), \quad (1)$$

where A_{ij} is the adjacency matrix of the network G and $A_{ij} = 1$ if there is a directed link from node i to j and $k_{out}(j)$ is the out-degree of node j (i.e., number of out-links from node j). We take the stationary state $P(i)$ of $P(i,t)$ as the PageRank of node i.

PageRank is a widely used node centrality to quantify influence of nodes in a given directed network. Originally PageRank was introduced for Google web search engine to rank web pages in World Wide Web based on the idea of academic citations [13]. Currently PageRank is used to rank nodes in various types of directed networks including citation networks of scientific papers [14, 15], social network services [16], world trade network [17], biological systems [18], and Wikipedia [19, 20, 21].

In this work each node i has a binary opinion $\sigma_i \in \{-1, +1\}$ and has PageRank P_i as a node influence based on network structure and Eq. (1). At each opinion update, a node i is randomly chosen and its opinion is updated considering its neighbor nodes' opinions. Each time step consists of N updates. Thus one time step corresponds to one update for each node on average. The opinion updating rule considers node influence of each neighbor node. Adopted from the original PageRank opinion formation (PROF) model [10, 11], the update rule reads: if the following function $H(i)$ for the chosen node i is positive, then $\sigma_i = +1$ otherwise $\sigma_i = -1$. The function $H(i)$ is given by:

$$H(i) = a \sum_{j \in \Lambda_{i,in}} \sigma_j P_j{}^g + b \sum_{j \in \Lambda_{i,out}} \sigma_j P_j{}^g, a + b = 1 \quad (2)$$

where $\Lambda_{i,in}$ is the group of in-neighbor nodes of node i (i.e., nodes have out-links to node i) and $\Lambda_{i,out}$ is the group of out-neighbor nodes of node i (i.e., nodes have out-links from node i), respectively. The parameter g quantifies the heterogeneity of node influence. If $g = 0$. then every node in the network has same node influence. If $g = 1.0$ then every node in the network can influence other nodes' opinion as much as its PageRank and and thus this case is reduced to the original PROF model [10]. Thus, $H(i)$ is the weighted summation of opinions of node i's neighbor nodes. In this study we use $a = b = 0.5$ for simplicity of analysis.

3. Empirical networks

We consider the following four empirical directed networks. (1) *Web graph*: we consider Web graph of University of Cambridge [22, 23]; here each node corresponds to a Web page and a link is hyper-link between the Web pages in the domain of University of Cambridge. (2) *Citation network*: we consider Physical Review citation network [15]; here a node corresponds to an article published in Physical Review journal of American Physical Society from 1897 to 2009 and the links correspond to the citation relations between the articles. (3) *Wikipedia*: we consider the network of articles in French Wikipedia [21]; the nodes correspond to articles in French Wikipedia (fr.wikipedia.org) and the links are the inter-articles hyper-links between the articles. (4) *LiveJournal*: we consider the social network of LiveJounral (livejournal.com) users; here the nodes are users of LiveJournal and the links are social relationship between the users; a more detail information on the network data are given in [24].

Statistical properties of the considered empirical networks are represented in Table 1. It is notable that unlike typical networks such as regular lattices or small-world networks considered in opinion formation models, all considered networks in this work have complex structural properties including broad degree distributions and broad distribution of PageRank [22, 15, 21, 10].

Table 1: Basic statistics of empirical directed networks, N gives the total number of nodes and L gives the total number of links.

Network	N	L
Web graph	212710	1831542
Citation	463349	4690897
Wikipedia	1352825	34431943
LiveJournal	3577166	44913072

4. Results

With the modified PROF model on described empirical networks, we investigate dynamics of collective opinion formation. First we consider evolution of the fractions of (+1) opinion, $f(t,+1)$, by time t to investigate whether considered networks can reach the steady state or not and whether they reach consensus opinion or not if the networks can reach the steady state. For simplicity, we represent $f(t) = f(t,+1)$. By definition, we can consider the fraction of (-1) opinion $f(t,-1) = 1-f(t)$ easily. Starting with same initial fraction of two opinions (i.e., $f(0,+1) = f(0,-1) = 0.5$), we numerically investigate how fractions of each opinion state evolve by time t. As shown in Fig. 1, all considered networks have reached the steady states. Subfigures located in the bottom row of Fig. 1 represent the evolution of the fraction of (+1) opinion nodes $f(t)$ along with time t and $g = 1$ (10 realizations for each network). For Wikipedia case (the third column of Fig. 1), we can observe "consensus" behavior (i.e., most of nodes have single major opinion whether (+1) or (-1)). However, we observed that Web graph (the first column of Fig. 1), Citation network (the second column of Fig. 1), and LiveJournal social network (the fourth column of Fig. 1) show non-consensus behavior (i.e., two finite values of opinion co-exist in the steady states). Here we define that if a given network have reached either $f_s > 0.95$ or $f_s < 0.05$, the network shows consensus behavior where f_s is the fraction of (+1) opinion in the steady state. We find that Web graph and Wikipedia relax to the steady state (either consensus or non-consensus) in short time ($t < 30$) as shown in Fig. 1 while more longer times ($t > 40$) are necessary to reach the steady states in cases of Citation and LiveJournal networks. Sub-linear g values cases (figures from the first to fourth row) show similar behaviors of reaching steady state with the linear cases. But it is notable that for Web graph and Wikipedia, the differences between each steady state fractions of (+1) opinions are bigger with growing g. We can consider this observation as a sign of growing polarization of steady state opinion. However, other networks give no clear signs. A further more quantitative analysis for these gaps between the fraction of steady state opinions are required.

To quantify the effects of g value on the relaxation time to the steady state of the collective opinion, first we define $\langle f(t)\rangle_{10}$ as an average fraction of (+) state for 10 consecutive time steps from time t to $t+9$ as following.

$$\langle f(t)\rangle_{10} = \frac{1}{10}\sum_{t}^{t+9} f(t) \tag{3}$$

We define time T_c of reaching the steady state for each network such that the standard deviation $\sigma(10)$ of above ten consecutive fraction $f(t)$ of (+1) opinion nodes from time $t = T_c$ to $t = T_c + 9$ is less than 0.0002. (i.e., $\sigma(10) < 0.0002$). Fig. 2 represents the relation between steady state relaxation time T_c and the influence exponent g. We can observe a clear tendency that bigger g (more heterogeneous influence the network has) leads to shorter time to reach the steady states for all networks. As Fig.1. implies, Web graph and Wikipedia

have shorter relaxation times $T_c < 30$ for various g while Citation and LiveJournal networks have significantly longer $40 < T_c < 110$ and effects of g variation are more pronounced.

In order to analyze opinion formation in the steady states and study polarization of steady state opinions, we investigate distributions of fraction of (+1) opinion f_s in steady state for each network. Fig. 3 represents the distributions of fraction of (+1) opinion in the steady states for each case of empirical network starting with $f(0,+1) = f(0,-1) = 0.5$. For the cases of Web graph, Wikipedia, and LiveJournal, increasing g resulted in more uniform opinion states (i.e., the fractions of majority opinion state whether (−1) or (+1) are getting higher with g). This indicates that a more heterogeneous node influence distribution in networks may lead to a more "totalitarian" society. However, the Citation network shows the opposite pattern. It is notable that the Citation network has different structural property from other directed networks. Unlike the other considered networks, reciprocal links (i.e., bi-directed links connecting from node i to node j and from node j to i.) are very rare in the citation networks due to time-ordering of citation relationships between scientific articles (i.e., it is practically not possible to cite publications in future). Thus this distinctive structure might affect behaviors of collective opinion on the network.

So far we considered only evolution of opinion states starting from the same fractions of initial opinion states (i.e., $f(0,+1) = f(0,-1) = 0.5$). If initial fraction of two opinions are different, then how collective opinions on networks are formed? In order to find out how the steady state fraction f_s of nodes with (+1) opinion depends on its initial fraction $f_i = f(0,+1)$, we investigate opinion formation with varying initial fraction of (+1) opinion and varying g. Fig. 4 represents a fraction of (+1) opinion in the steady state f_s versus an initial fraction of (+1) opinion f_i for each empirical network. Each row in Fig. 4 represents each network and each column represents each value of g.

In the case of Web graph, we can observe the emergence of bistability as g is increasing. Here bistability means there exist two steady state fractions of (+1) opinion. The bistability of Web graphs is also observed in [10] in the case of University of Cambridge and Oxford Web graph with original PROF model (i.e., $g = 1.0$). When g is small ($g \leq 0.25$), the fraction of (+1) opinion f_s in the steady state reached single value of fraction with some fluctuations. Meanwhile, when $g \geq 0.5$, there are two values of f_s in the steady state. For LiveJournal network, there are signs of multiple steady state fractions of (+1) opinion as shown in Fig. 3(D). This phenomenon is also observed in Fig 4 but only for $f_i = 0.5$. If $f_i \neq 0.5$, we cannot observe such multistability in the steady state. On the other hand, there is no such bistability for the case of Citation network and Wikipedia. In particular the Wikipedia network shows if the initial fraction of (+) opinion is less (more) than 0.45 (0.55), the final fraction is always less (more) than 0.05 (0.95). Based on the observation, the initial fraction of the opinion states can be critical for opinion formation in these networks but the detail behaviors can be different depending on the types of networks.

To characterize the effects of influential nodes on opinion formation, we investigate how a group of selected nodes with a fixed opinion can impose their own opinion on the entire network. We compare two opinion implanting strategies of n seed nodes with a fixed opinion.

In the *random implanting strategy*, we choose n nodes as seed nodes from a given network randomly and assign (+1) opinion to them. The opinions of seed nodes are fixed. We assign (−1) opinion to the rest of nodes (i.e., non-seed nodes) in the networks. The opinions of the non-seed nodes are flexible thus their opinions can be changed by the modified PROF rule at each update. Meanwhile in the *targeted implating strategy*, we choose n nodes as seed nodes in order of PageRank of the nodes and assign (+1) opinion to them. The opinions of seed nodes are also fixed. We assign (−1) opinion to the rest of nodes in the network and update the opinions of non-seed nodes by modified PROF rule as in the random implanting strategy at each update.

Fig 5 compares the fraction of (+1) opinion nodes in the steady state by two implanting strategies. Regardless of networks and value of g, targeted implanting cases are much more effective to lead collective opinion states of the networks to (+1) opinion. Even when $g = 0.0$ (i.e., every node has the same node influence), targeted implanting is more effective than random implanting strategy to change the nodes in the networks to (+1) opinion. The tendency is getting stronger with g. For the Citation, Wikipedia, and LiveJournal networks, even a very small fraction of top influential nodes with fixed (+1) opinion (i.e., $f(0) \leq 0.01$) can lead to the significant fraction of (+1) opinion in the steady state on the networks. For the Web graph, the tendency is weaker partially due to the "bistability" we observed above. In [10], it was observed that imposing (+1) opinion on small initial fraction ($\sim$ 1 percent of nodes) of top PageRank nodes can lead 40 percent of (+) opinion states. Our analysis indicates this "elite" effect can exist even when every node has the same influence but

the elite effect can be much stronger when node influence are heterogeneously distributed with a larger value of g.

5. Discussion

Opinion formation in social systems is mediated by social interactions between the individuals in the systems and at the same time it is affected by influence of interacting nodes. Thus understanding this interplay between individuals' influence and network structure of social interactions is a salient issue. In this study we used the modified PageRank opinion formation (PROF) model to consider how heterogeneous node influence affects collective opinion formation on real networks and analyzed effects of heterogeneity of node influence on opinion formation. We found that the relaxation time to reach the steady state is decreasing with the heterogeneity of node influence in the networks. We also identified that a small number of influential nodes can impose their opinion on significant fraction of nodes, and the impacts of these social elites on collective opinion is growing with the heterogeneity of node influence.

All of considered networks reach a steady opinion state. However, it is not clear why only Wikipedia shows consensus and the other networks do not. Since we considered directed networks, asymmetric nature of links could be the obstacle to reach consensus. To check the effect of the asymmetric nature of links, we considered undirected version of empirical networks but observed the same non-consensus behaviors. Thus we can rule out this explanation. On the other hand, a strong local structure such as communities or modules [25, 26] can prohibit to reach the consensus opinion state. Since communities in networks are typical composed of a group of tightly connected nodes, such a densely connected group of nodes may persist the influence from other parts of the networks. It would be interesting to study an interplay between influential nodes and community structure. The Citation network also displays the opposite behaviors from the other networks such that the other networks show more uniform opinions states with growing g while Citation network shows less uniform steady state opinion. It is interesting to check if other citation networks show similar behaviors with our Citation network.

In this study we used PageRank and its sub-linear power as node influence. However, other node centralities on directed network can be considered as node influence including in-degree, betweenness centrality [27], CheiRank [28], 2DRank [29], or non-structural node attributes. Since PageRank is positively correlated with in-degree, the study of considering node influence which is positively correlated with in-degree can be interesting. As described above, community or core-periphery structures may also significantly affect the collective opinion formation with a local structure-based influence measure.

Due to the advent of information technology and growing usage of social media, the problem of collective opinion formation is getting more and more complicated going to a global scale. A quantitative understanding of opinion formation on large-scale networks becomes of crucial importance. Our study sheds a new light on how the node influence and network structure together affect the collective opinion in directed networks.

Acknowledgments

We thank V.Kandiah for useful discussions and American Physical Society for letting us use their citation database for Physical Review journals. This work is supported in part EC FET Open project “New tools and algorithms for directed network analysis (NADINE)” - No. 288956. Y.-H. Eom also thanks for supporting of the EC FET project “Financial Systems SIMulation and POLicy Modelling (SIMPOL)” - No. 610704.

References

[1] C. Castellano, S. Fortunato, V. Loreto, Rev. Mod. Phys. 81 (2009) 591
[2] H. Xia, H. Wang, Z. Xuan, Int. J. Knowl. Syst. Sci. 2 (2011) 72.
[3] S. Galam, J. Math. Psych. 30 (1986) 426.
[4] S. Galam, Int. J. Mod. Phys. C 19 (2008) 409.
[5] V. Sood, S. Redner, Phys. Rev. Lett. 94 (2005) 178701.
[6] K. Suchecki, V.M. Eguíluz, M. San Miguel, Phys. Rev. E 72 (2005) 036132.
[7] S. Galam, Eur. Phys. J. B 25 (2002) 403.
[8] G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch, Adv. Compl. Syst. 03 (2000) 87.
[9] K. Sznajd-Weron, J. Sznajd, Int. J. Mod. Phys. C 11 (2000) 1157.
[10] V. Kandiah, D.L. Shepelyansky, Physica A 391 (2012) 5779.
[11] L. Chakhmakhchyana, D.L. Shepelyansky, Phys. Lett. A 377 (2013) 3119.
[12] A.M. Langville, C.D. Meyer, Googles PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, Princeton, 2006.
[13] S. Brin, L. Page, Comput. Netw. ISDN Syst. 30 (1998) 107.
[14] P. Chen, H. Xie, S. Maslov, S. Redner, Jour. Informetrics 1 (2007) 8.
[15] K.M. Frahm, Y.-H. Eom, D.L. Shepelyansky, Phys. Rev. E 89 (2014) 052814.
[16] H. Kwak, C. Lee, H. Park, S. Moon, What is Twitter, a social network or a news media? in: Proc. 19th Int. Conf. WWW2010, ACM, New York, NY, 2010, p. 591.
[17] L. Ermann, D.L. Shepelyansky, Acta Physica Polonica A 120 (2011) 6A.
[18] V. Kandiah, D.L. Shepelyansky, PLoS ONE 8 (2013) e61519.
[19] Y.-H. Eom, D.L. Shepelyansky, Euro. Phys. Jour. B 86 (2013) 492.
[20] Y.-H. Eom, D.L. Shepelyansky, PLoS ONE 8 (2013) 74554.
[21] Y.-H. Eom, P. Aragón, D. Laniado, A. Kaltenbrunner, S. Vigna, D.L. Shepelyansky, (2014) arXiv:1405.7183.
[22] K.M. Frahm, B. Georgeot, D.L. Shepelyansky, J. Phys, A: Math. Theor. 44 (2011) 465101.
[23] Academic Web Link Database Project. http://cybermetrics.wlv.ac.uk/database/.
[24] M. Kurucz, A. Benczur, A. Pereszlenyi, Large-scale principal component analysis on livejournal friends network, in: Proc. Workshop on Social Network Mining and Analysis Held in Conjunction with 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, Las Vegas NV, August 24.27, 2008. http://dms.sztaki.hu/en/letoltes/livejournal-data.
[25] M. Girvan, M. E. J. Newman, Proc. Natl. Acad. Sci. USA 99 (2002) 7821.
[26] S. Fortunato, Phys. Rep. 486 (2010) 75.
[27] S. Wasserman, K. Faust, Social Networks Analysis, Cambridge University Press, Cambridge, 1994.
[28] A. D. Chepelianskii, (2010) arXiv:1003.5455.
[29] A.O. Zhirov, O.V. Zhirov, D.L. Shepelyansky, Eur. Phys. J. B 77 (2010) 523.

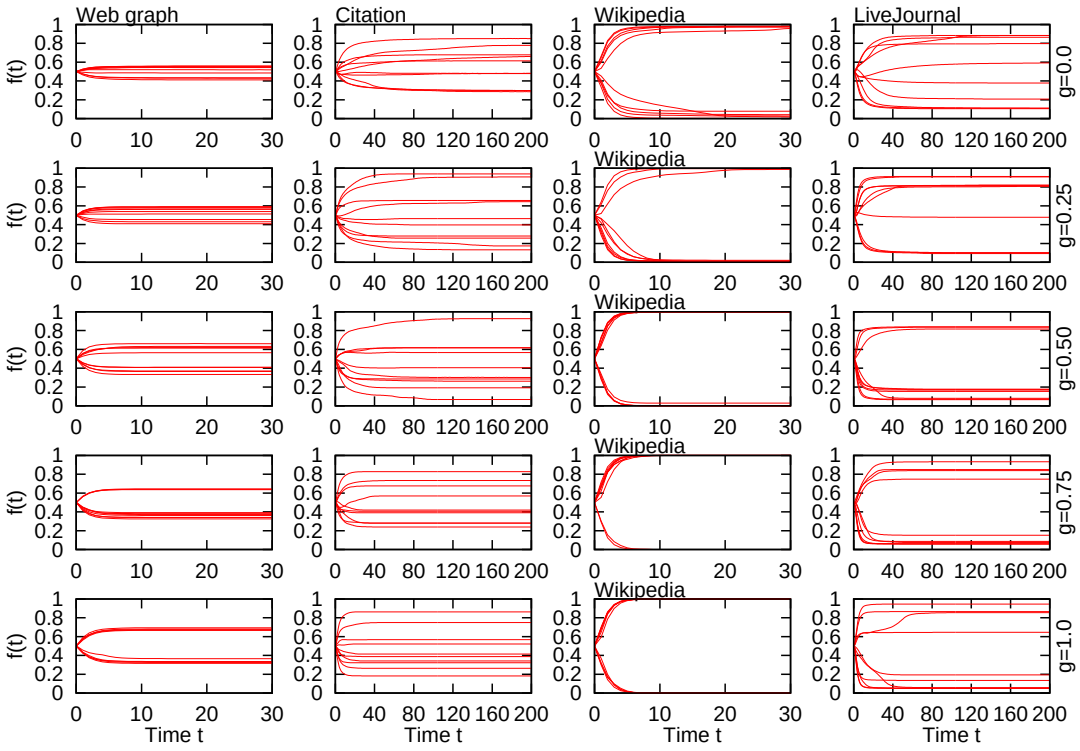


Figure 1: Evolution of the fractions of (+1) opinion $f(t)$ in time t; here 10 realizations per each network and each value of g are represented. Each column corresponds to the network and each row corresponds to g.

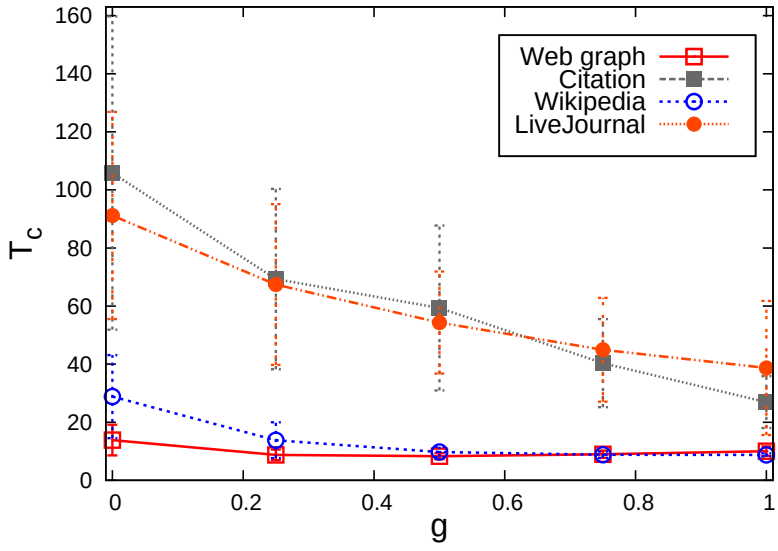


Figure 2: Dependence of the relaxation time T_c to a steady state on the influence exponent g for considered networks.

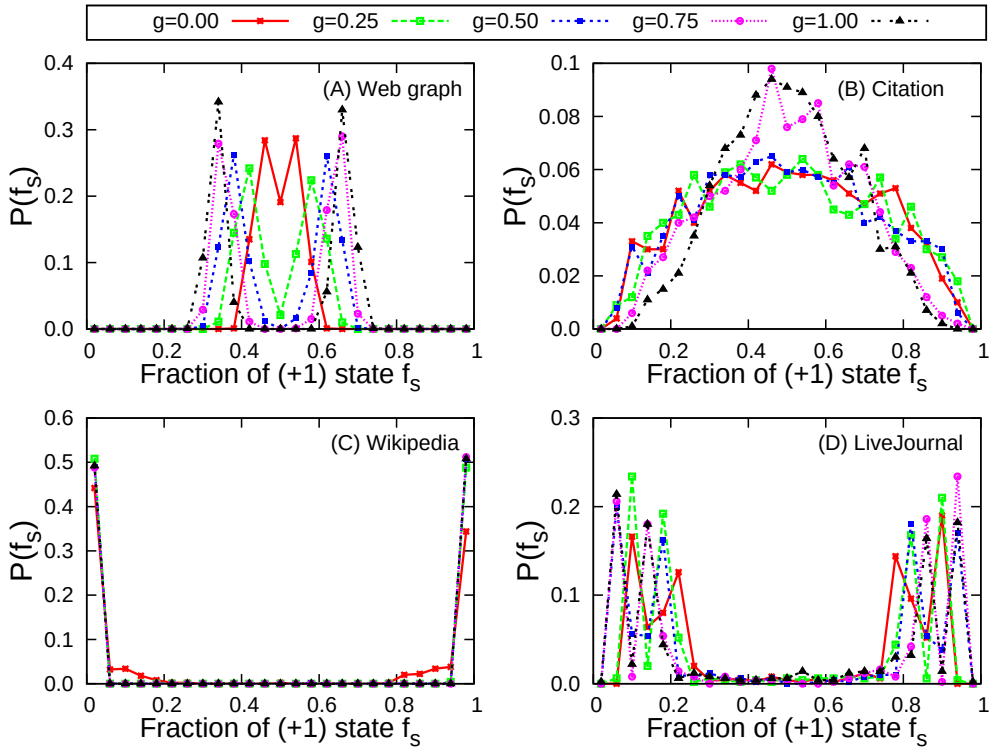


Figure 3: Distributions of (+1) opinion fraction in the steady state for each empirical network. Here f_s is the fraction of (+1) opinion in steady state and $P(f_s)$ is the probability distribution function of f_s. All the cases start with initial fraction of $f(+1,0) = f(-1,0) = 0.5$ with 1000 realizations for Web graph and Citation networks and 500 realizations for Wikipedia and LiveJournal.

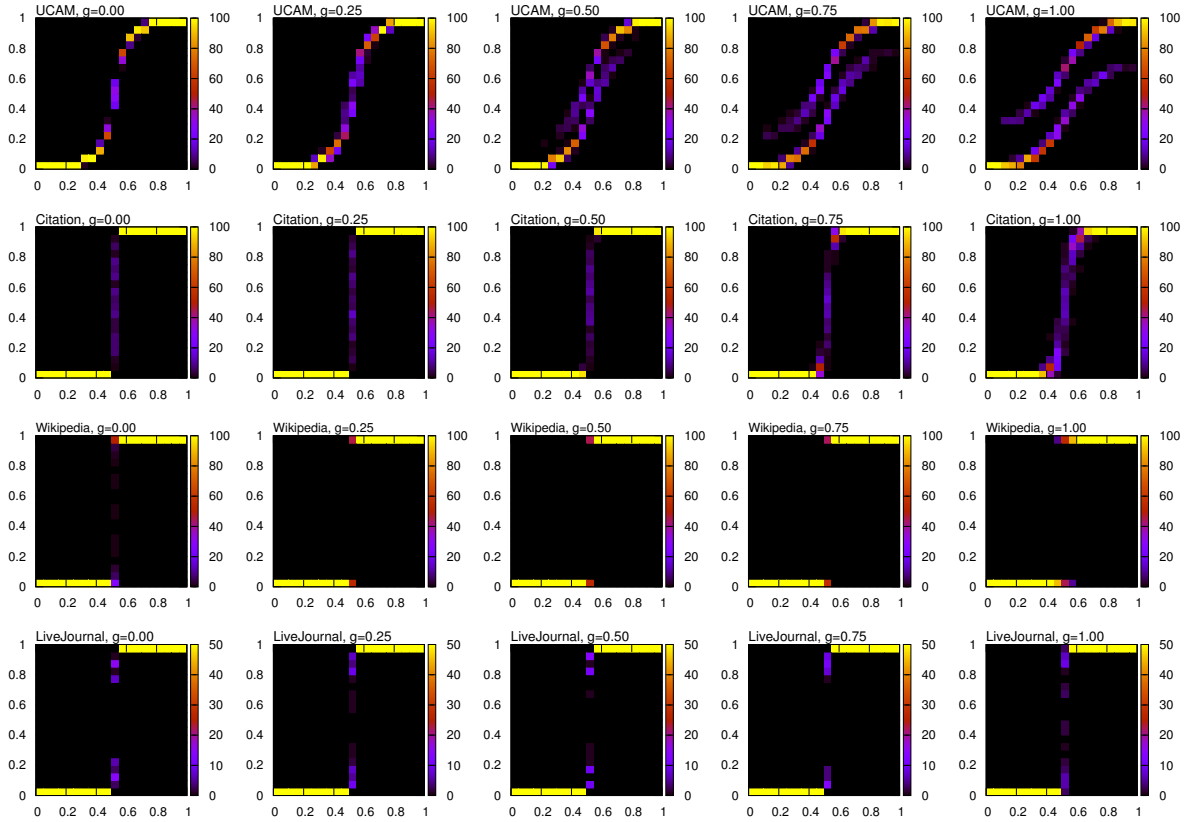


Figure 4: Fraction of (+1) opinion states f_s ($y-$axis) in the steady state as function of initial fraction f_i ($x-$axis) of (+1) opinion state for given network and g. Each row corresponds to each network and each column corresponds to the value of g. Here there are 100 realizations for Web graph, Citation networks, and Wikipedia and 50 realizations for LiveJournal. Here the color marks the relative number of cases obtained for give values (f_i, f_s), the color changes from black (zero) to red (maximal number of cases).

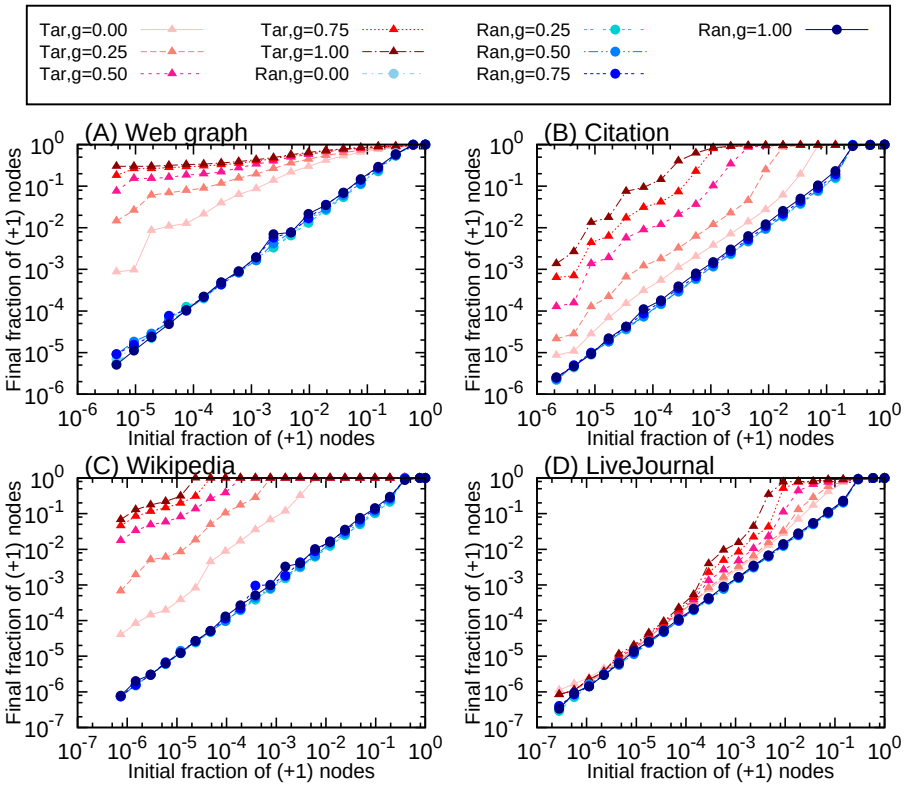


Figure 5: Comparisons between the target implanting strategy and random implanting strategies. Here f_s and f_i represent the fraction of (+1) opinion nodes for steady state and initial state on the network respectively. "Tar" represents the targeted implanting strategy and "Ran" represents the random implanting strategy. For targeted implanting strategy (filled triangles), pink, salmon, dark-pink, red, and dark-red colors represent $g = 0.0$, $g = 0.25$,$g = 0.5$,$g = 0.75$, and $g = 1.00$, respectively. For random implanting strategy (filled circles), skyblue, dark-turquoise, web-blue, blue, and navy represent $g = 0.0$, $g = 0.25$,$g = 0.5$,$g = 0.75$, and $g = 1.00$, respectively. Here there are 100 realizations for Web graph and Citation networks and 50 realizations for Wikipedia and 25 realizations for LiveJournal.

RecSys Challenge 2014: an ensemble of binary classifiers and matrix factorization

Róbert Pálovics[1,2] Frederick Ayala-Gómez[1,3] Balázs Csikota[1] Bálint Daróczy[1,3]
Levente Kocsis[1,4] Dominic Spadacene[1] 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 [4]University of Szeged
{rpalovics, fayala, bcsikota, daroczyb, kocsis, domspad, benczur}@ilab.sztaki.hu

ABSTRACT

In this paper we give our solution to the RecSys Challenge 2014. In our ensemble we use (1) a mix of binary classification methods for predicting nonzero engagement, including logistic regression and SVM; (2) regression methods for directly predicting the engagement, including linear regression and gradient boosted trees; (3) matrix factorization and factorization machines over the user-movie matrix, by using user and movie features as side information. For most of the methods, we use the GraphLab Create implementation. Our current nDCG achieves 0.877.

Categories and Subject Descriptors

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

Keywords

Recommender systems, RecSys Challenge; GraphLab Create; Twitter

1. INTRODUCTION

The RecSys Challenge 2014 data set consists of movie ratings automatically tweeted by the IMDb application. In this unusual prediction task, for each user, the top-K tweets based on predicted engagement is requested, where engagement consists of favorites and retweets. Evaluation is 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].

Predicting the number of retweets is a known task: [13] investigates the problem of predicting the popularity of messages as measured by the number of future retweets and [18] finds that performance is dominated by social features, but that tweet features add a substantial boost. In our work we use ideas from these papers for defining user and tweet features.

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

Since movie rating tweets are defined over a pair of a user and a movie, we face a dyadic prediction task where recommender and learning-to-rank approaches may also be applicable. In our method we blend recommender methods with side information and direct predictors for the engagement, which are based on a pool of user, movie and tweet features.

Our first class of methods consists of recommenders over the data set considered as a user-movie matrix. The Netflix Prize competition [5] put recommender algorithms through a systematic evaluation on standard data [3]. The final best results blended a very large number of methods whose reproduction is out of the scope of this experiment. Among the basic recommender methods, we use matrix factorization [22, 15].

A twist in the data set over the user-movie matrix is rich side information, both for users and for movies. In addition, some users and movies appear only in the test set and hence there we face the cold start problem [20]. Some methods that use the side information include stochastic gradient descent [17] and the product of user and item kernels [4]. In our experiments we use the factorization machine [19] as a very general toolkit for expressing relations within side information.

We noticed that the most important part is to distinguish between zero and nonzero engagement. We find that the solution of this simplified version of the problem still results in a very high nDCG score of 0.986. In this sense, the task is a mix of a recommender and a binary classification tasks. We use linear models including regression and Support Vector Machines (SVM) [21].

Our final prediction relies on the ensemble of a large number of methods. Classifier ensembles are known to offer a significant improvement in prediction accuracy [23, 8, 7]. Ensembles include changing the instances used for training through techniques such as Bagging [2] and Boosting [12]. In our results, we use gradient boosted trees [26] and we combine classifiers, regressors and recommenders both by averaging and by learning their weigth by linear regression.

In most of our models, we use the GraphLab Create implementation[1] [16].

[1]`http://graphlab.com/products/create/`

Table 2: Users and movies in the training and testing sets.

	Users	Movies
Training set	22,079	13,618
Test set	5,717	4,226
Unique to training set	17,838	10,090
Unique to test set	1,476	698
Apperaing in both	4,241	3,528

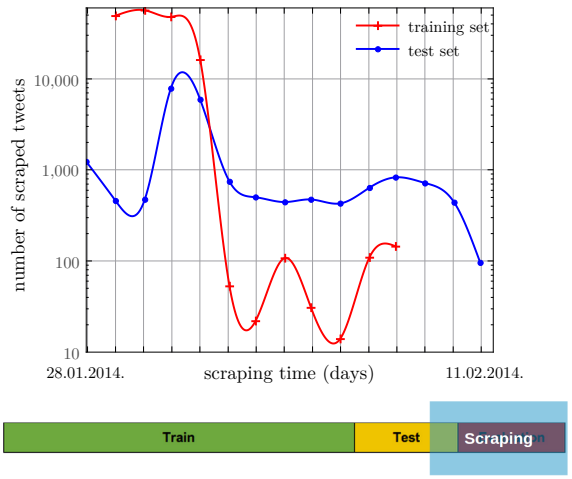


Figure 1: **Top**: Daily crawling frequencies. **Bottom**: The scraping period with respect to the training, testing and evaluation sets.

2. THE RECSYS CHALLENGE 2014 DATA SET

In this section we overview the key properties of the data set that we use for feature generation and modeling. For general information on the number and date of users, movies and tweets we refer to Table 2.

2.1 Training and test set statistics

The training and test sets greatly differ. Table 1 summarizes their properties. While in both data sets, the percentage of tweets with zero engagement is high, the ratio is significantly higher in the training set. Users may belong only to training, only to testing or to both sets (Table 2). We observe large difference between the training, testing and evaluation sets in the time elapsed between the tweet appeared and crawled due to the different timing of the training, testing and evaluation sets, as described in Fig. 1.

2.2 IMDb movie metadata

We collected IMDb related movie features from two different sources. The original MovieTweetings dataset [11] contains release year, movie genre(s) and title for each movie in the dataset. We transformed genres into a binary feature vector. Besides these features, we used the IMDBbPY Python package[2] to crawl two elementary features of the movies in the dataset: their average rating in the system, and the number of ratings they have.

[2]`http://imdbpy.sourceforge.net/`

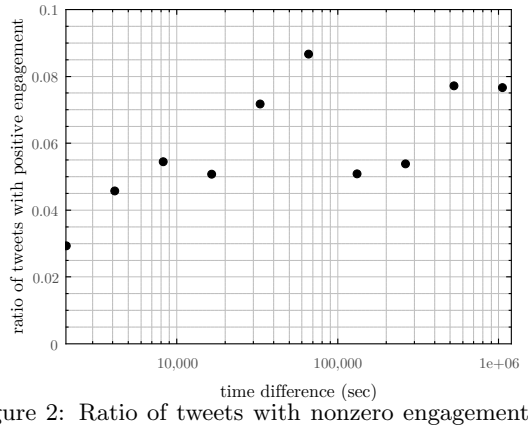


Figure 2: Ratio of tweets with nonzero engagement as the function of the time difference between the creation and scraping timestamps.

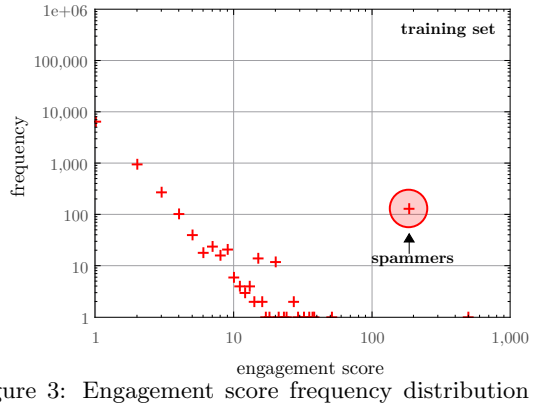


Figure 3: Engagement score frequency distribution of the training set.

2.3 Spam bots

While the frequency of engagement scores appears to follow a power-law relationship, we found within the training set a large number of tweets (130) which had an engagement score of 185, see Fig. 3. They were found to all have been a retweet of a tweet from one of Justin Bieber's friends, Ryan Butler, who also appears to have a large following. Many of the screen names of these users included variations of the name 'Justin Bieber' (e.g. 'thatkidbiebuh', 'BiebsDevotion', 'BelievingJ',...), leading us investigate celebrity posts. In fact the retweet times of such posts can occur within seconds of the post so that we term them 'spam bots'.

One strong indicator of celebrity appears to be the 'verified user' boolean, which is included in every status. In the training data, there are 16 tweets from verified users, all but 3 of which received some form of engagement. In the test data, only one such tweet exists, which too received engagement.

In light of these, we decided to use to remove the Ryan Butler-related tweets from the training data before we trained our models. Compared to the original engagement scores in Fig. 3, after cleansing, the training and testing distribution appears similar in Fig. 4.

Table 1: Training and test set properties.

	Training set	**Test set**
Number of users	22,079	5,717
Number of movies	13,618	4,226
Number of tweets	170,285	21,285
Number of zero engagement scores	162,108 (**95.19%**)	19,727 (**92.68%**)
Earliest creation time	28/02/2013 14:43	8/01/2014 22:06
Latest creation time	8/01/2014 22:06	11/02/2014 15:49
Minimum number of days between creation and scraping	**23**	**0**

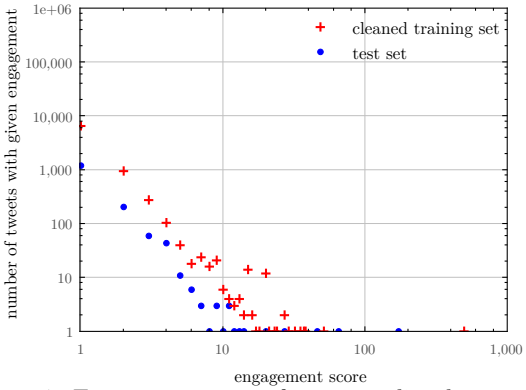


Figure 4: Engagement score frequency distributions in the cleansed training set and the test set.

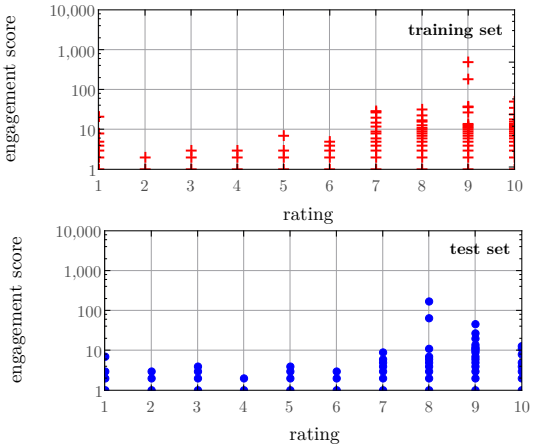


Figure 5: Engagement score as the function of rating for the training set (**top**), and for the test set (**bottom**).

Table 3: Properties of the cleansed training set.

Number of users	21,950
Number of movies	13,618
Number of tweets	170,155
Number of tweets with zero engagement score	162,107 (**95.27%**)

2.4 Extreme ratings

In Fig. 5 a U-shaped relationship can be made out within both datasets. In other words, tweets with extremist ratings generate higher engagement scores in average. In fact in terms of linear regressions based on only one feature, we found that rating performed the best, achieving as high as 0.818 nDCG on the test set. However, we found tweets containing ratings that fell outside of the "standard" range determined by the IMDb tweet template (1-10, inclusive). In the training set, 73 ratings were given outside of this range. The higher "extreme rating" tweets also more often receive engagement than regular tweets.

2.5 Cleansed training set

Table 3 summarizes the properties of our cleansed training dataset. We removed tweets with rating larger than 10 or less than 1. We also excluded the spammers from the dataset.

2.6 Features

Here we list the features we extracted from the original datasets.

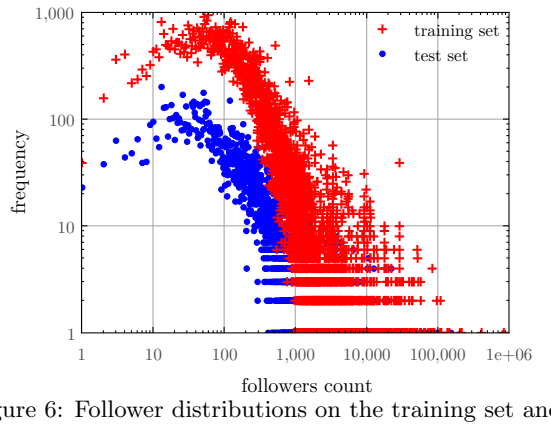


Figure 6: Follower distributions on the training set and the test set.

User features extracted from the Twitter json data: creation time, followers and friends count (see Fig. 6, favourites count, statuses count

Tweet features extracted from the Twitter json data: creation time, scraping time, time difference between creation and scraping, number of hashtags / URLs / mention in the tweet, tweet is retweet, tweet has retweet in the dataset

Table 4: Best nDCG@10 results of the invidual algorithms, separate for resubstitution over the training set and for the test set. In both cases we give the best result not using and using retweet information ("no RT" and "with RT", respectively).

Method		**Training set**		**Test set**	
		no RT	with RT	no RT	with RT
Random prediction	(RN)	≈ 0.61	-	≈ 0.75	-
Retweet based prediction	(RT)	-	0.73038	-	0.80098
Rating based prediction	(RA)	0.69386	0.75670	0.82125	0.84980
Graphlab Create Item Means Model	(IM)	0.74821	0.80729	0.79957	0.84340
Graphlab Create Popularity Model	(POP)	0.75935	0.81684	0.79467	0.83862
SVM	(SVM)	0.63976	0.73112	0.81850	0.86777
Graphlab Create Gradient Boosted Tree	(GCGT)	0.71009	0.79220	0.83336	0.87127
Graphlab Create Linear Regression	(GCLR)	0.67782	0.75374	0.82726	0.86089
Graphlab Create Logistic Regression	(GCGR)	0.67724	0.75270	0.82807	0.86078
Graphlab Create Matrix Factorization	(GCMF)	0.70343	0.77598	0.82444	0.86291
Graphlab Create LibFM based recommender	(GCFM)	0.68005	0.75878	0.82019	0.84139
NDCGboost	(NB)	0.68663	0.77032	0.80532	0.86282

Movie features extracted from the MovieTweetings dataset and our IMDb crawl:
movie average rating, number of ratings on IMDb, binary vector of movie genres from the movieTweetings dataset.

Rating based features: Besides the original user rating, we take the difference of the user's rating and the movie's average rating on IMDb. We also give a score for each tweet based on its extremeness (see 2.4). Moreover, we have created feature vectors from the ratings. We use the average engagement value as the function of the rating (1–10) as shown in Fig. 5.

2.7 Zero and nonzero engagement

In order to simplify the task, we considered separating those tweets that received engagement from those that did not. Assuming that this binary classification task could be perfectly solved, we measured an nDCG@10 of 0.988 over the test set. The fact that the actual engagement count is less important compared to separating zero from nonzero is in part due the shear number of tweets with no engagement (see Table 1), as well as the fact that most of the users have no more than 5 tweets: in the training data, 7,012 of the 22,079 users have 5 or more tweets. Note that of the 7012 users in the training set with more than 5 tweets, only 2285 of them have at least one tweet with nonzero engagement. The advantage of the zero-nonzero task is that binary classifiers are applicable, some of which giving the strongest performance with respect to nDCG, see Table 4.

2.8 Information leakage through retweets

Retweets of movie rating tweets were also fetched by the Twitter API. A retweeted message has by definition nonzero engagement. In addition, the retweet of a message received the engagement score of the original message, possibly as a side effect of how the Twitter API gives information on retweets. Moreover, if we see a retweet in the dataset, if its root tweet is in our dataset, we immediately know that the root tweet's engagement score is higher than 0. We use retweet information in two ways. First, we may include the fact that a retweet exists as a binary feature. Second, since retweets and retweeted messages have nonzero engagement, we increased the predicted score by a large constant for these tweets. For reference, we included the performance of all methods without using the retweet information in any way.

3. THE EVALUATION METRIC

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 or AUC with good top-K performance [14, 25] and hence we follow their approach.

The RecSys Challenge 2014 task is evaluated by nDCG, one of the most common measures for top-K recommender evaluation [1]. Note that we train our binary classifiers optimized for RMSE or AUC, both evaluated as a macro measure by globally ranking all tweets. The challenge evaluation, in contrast, defines a user micro average.

4. ELEMENTS OF OUR ENSEMBLE AND EXPERIMENTAL RESULTS

We describe two groups of methods. The first one is based on binary classification and regression in Section 4.2. The details of the features and parameters used in each model are described in Table 5. The second one in Section 4.3 is an online matrix factorization. The performance overview is found in Table 4 both for resubstitution on the training set and for the testing set. We show results not using any retweet based information separate. The columns that use retweet information include the best performing combination of using retweet as a binary feature for training and moving retweets and their root tweets ahead in the ranking. The final blending results are in Table 6.

4.1 Baseline measurements

We defined five nDCG baselines for both the training and testing set to benchmark the result of our models. The first is the random prediction (RN) for the models that does not use the retweet features. This method randomly sorts the tweets for each user and then calculates the nDCG. The second is the retweet based prediction (RT) for the models that uses retweet features. This model gives score 1 to tweets that are retweets or root tweets, and score 0 to all other

Table 5: Main parameters of our algorithms.

Method	Parameters
GCGT	num_trees: 18, step_size: 0.3, max_depth: 4, num_iterations: 18, min_child_weight: 0.1, min_loss_reduction: 0
GCLR	L2_penalty: .10, solver: newton, max_iter: 10
GCGR	L2_penalty: .10, solver: newton, max_iter: 10
GCMF	n_factors: 8 linear_regularization: 0, regularization: 0.0001
GCFM	n_factors: 8 linear_regularization: 0, regularization: 0.0001
SVM with RT Features	C : 0.5, kernel: linear
SVM no RT Features	C : 0.25, kernel: polynomial, degree: 2
NB	num_trees : 20, num_leaves: 8

Table 6: Best blended nDCG@10 results.

Blending Method	Test set	
	no RT features	with RT features
Best combination achieved by grid search	0.83922	0.87713
Average of RA, GCGT, GCGR	0.38973	0.87340
Scikit-learn Linear Regression	0.84027	0.87435

tweets. Our rating based prediction (RA) uses the U-shape based extremeness to predict the ranking of the tweets. We also used the popularity model and the item means model in Graphlab Create as baseline measures. The values of the baselines are shown in Table 4. It turns out that the nDCG changes because of the different properties of each dataset (e.g. engagement frequency, users engaged.).

4.2 Binary classification

In order to apply the binary classification methods, we created a binary feature that expresses if a tweet had engagement or not. We applied linear regression, logistic linear regression, Gradient Boosting Trees [26] and SVM [21].

For both linear regression (GCLR) and logistic linear regression (GCGR), the Newton method and stochastic sradient descent (SGD) solvers were used. However, the Newton method solver led to a better nDCG@10 than SGD. The features were rescaled using the L2-norm to ensure that the features have the same norm. The strongest three features were the rating, the difference of the rating and the movie's average rating in IMDb, and the number of personal mentions in the tweet. Note that one can edit the tweet generated by the IMDb application. If someone includes a personal mention in a tweet, it has higher probability to receive engagement in the future.

In case of the gradient boosted tree algoritm (GCGT) we set the maximum depth of the trees 4, and enabled maximum 18 iterations. Note that this algorithm performed the best on the test set even with and without using the retweet features.

By using support vector machine (SVM) we were able to achieve our second highest nDCG@10. We could observe that normalization, parameters and kernel selection are a very important step. Because of the different scale of the features, we scaled them linearly between zero and one except for the rating and retweet features. Our main reason was to gain advantage of the known relevance of these features.

4.3 Matrix factorization

We used two different matrix factorization techniques that are both implemented in GraphLab Create. The first one is a traditional matrix factorization method [15], while the second one is Steffen Rendle's LibFM algorithm [19]. Both techniques use stochastic gradient descent to optimize for mean-square error on the training set. The difference between the methods is that LibFM uses the side information of the users and items more sophisticatedly. Therefore we used the original matrix factorization technique without any side information, and used LibFM for user and item feature included collaborative filtering prediction. Note that in both cases we used movies instead of tweets as items, as each tweet (excluding the few retweets) is unique in the data.

4.4 Ranking based approach

NDCGboost [24] (NB) is a decision tree boosting algorithm that optimizes the expectation of NDCG over all possible permutations of items. We usend NDCFboost as one algorithm in our ensemble. The models included twenty trees with 8 leaves each.

4.5 Blending

We combined our methods linearly to yield the final prediction. Our methods reach close to the best possible combination weights that we obtained by grid search over the testing set as seen in Table 6. In the simplest method, we average the well performing methods. We also learn the weights by linear regression. Here we used the implementation of scikit-learn[3]

Conclusions

The RecSys Challenge 2014 task for predicting engagement of movie rating tweets has turned out to be a mix of Twitter activity prediction and user-movie top-K recommendation. For the activity prediction component of the task, classification and regression methods have worked well. And for top-k recommendation, we have used dyadic classifiers, variants of recommender methods that may use side information. As different classes of methods model different views

[3]http://scikit-learn.org/

of the data, they have combined well in our final ensemble. Due to the variety of user, movie and tweet side information, data collection and cleansing issues have played an important role. We have collected IMDb movie metadata, removed Twitter spam, and noticed an information leak for retweet information that was probably unintentionally collected for the challenge data set.

Acknowledgments

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).

5. REFERENCES

[1] A. Al-Maskari, M. Sanderson, and P. Clough. 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*, pages 773–774. ACM, 2007.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999.

[3] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[4] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46, 2005.

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

[6] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19:193, 2007.

[7] R. Caruana, A. Munson, and A. Niculescu-Mizil. Getting the most out of ensemble selection. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 828–833, Washington, DC, USA, 2006. IEEE Computer Society.

[8] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 18, New York, NY, USA, 2004. ACM.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

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

[11] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

[12] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[13] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 57–58, New York, NY, USA, 2011. ACM.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[16] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[17] A. K. Menon and C. Elkan. A log-linear model with latent features for dyadic prediction. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 364–373. IEEE, 2010.

[18] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In *ICWSM*, 2011.

[19] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[20] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[21] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. *Journal of the American Statistical Association*, 101:1730–1730, December 2006.

[22] G. Takács, I. Pilászy, B. Németh, and D. Tikk. 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*, pages 1–8. ACM, 2008.

[23] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996.

[24] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS) 22*, pages 1883–1891, 2009.

[25] M. Weimer, A. Karatzoglou, and A. Smola. Adaptive collaborative filtering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 275–282. ACM New York, NY, USA, 2008.

[26] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.

Peer-to-peer Online Collaborative Filtering

Andrea N. Bán
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
ban.andrea@sztaki.mta.hu

Levente Kocsis
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
kocsis@sztaki.mta.hu

Róbert Pálovics
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
palovics.robert@sztaki.mta.hu

ABSTRACT

Recommender systems often deal with a large amount of sequential data. For these scenarios, online matrix factorization techniques based on online prediction and incremental updates are often the most promising approaches. Decentralizing the system and keeping the user data on their devices is an important step in the direction of preserving user privacy. In this paper we propose a peer-to-peer online matrix factorization algorithm that stores the ratings of a user and her private data local. Additionally, the users have a local copy of the common part of the factor model and communicate with other users to advance towards a consensus on it. The algorithm is proven to converge to a set of local optima in the stationary case, while we show empirically that the algorithm performs well in the non-stationary case, both in terms of ranking performance and privacy preservation.

Categories and Subject Descriptors

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

Keywords

Recommender systems, Collaborative Filtering, Online learning, Peer-to-peer networks

1. INTRODUCTION

Recommender systems are now ubiquitous in most online applications. Approaches based on collaborative filtering have enjoyed significant success in several competitions, and became an area of extensive research. In the competitions it is natural to separate the collected data into a training set and a test set. However, in most real applications the data arrives continuously, and the systems need to recommend some items on the spot. While several authors argued for an online testing scenario [21, 11], it has received less attention. Incremental processing also becomes a necessity due to the size of the data. In this article we consider the scenario where users connect to the system sequentially, they request a recommendation, and the system is updated depending on the preference of the user.

Most recommender systems have a centralized structure, negating privacy requirements of the users. It is argued in [24] that it is paramount for privacy that users keep their data (e.g. ratings and user specific models) locally. Instead of storing such data on cloud servers, it is a natural idea to keep it only on personal devices. Consequently, the processing is fully distributed. This would increase the level of privacy and remove the dependence on a central infrastructure. An early approach in this direction is [20] that adapts a neighborhood algorithm to a fully distributed architecture. More recently, [28] used contextual bandit framework for social recommender systems. While these are fairly interesting approaches, we prefer matrix factorization models that have been proven perhaps the most successful approaches to collaborative filtering [18].

In this paper we consider an approach to online matrix factorization, where the user preferences and user latent vectors are kept locally, along with an instance of the item matrix. Consensus on the item matrix is reached by exchanging parts of the item matrix with other users. This way the most sensitive data stays local. We assume that revealing parts of the local copies of the item matrix will not reveal too much about the users preference. While we believe that our approach is a right step in improving the privacy of the users, and we provide an empirical evaluation against a deanonymization algorithm, the main focus of the paper is on the prediction quality of the peer-to-peer algorithm.

For non-stationary stochastic optimization, regret bounds on the online performance are provided mostly when the objective function is convex (see e.g., [6]). For batch settings, [8, 7] show that peer-to-peer stochastic approximation algorithms converge to some local optima when the complete model is sent. In a stationary online collaborative filtering framework, we extend their work on two aspects: (1) we assume that the model has a private component, and this component is not sent between peers; and (2) only parts of the common component of the model is sent at a certain moment. The first aspect is natural from privacy point of view, while the second may be beneficial for reducing the necessary communication.

The rest of the paper is organized as follows. In Section 3 we provide a skeleton for peer-to-peer online prediction algorithms. Further, we prove that the algorithm converges to a set of local optima. In this section we consider a more

general framework than matrix factorization: a model that has a private and a common component. Centralized online matrix factorization algorithms are described in Section 4 for rating prediction and for top-K recommendation. The peer-to-peer version of these matrix factorization algorithms are described in Section 5. Section 6 discusses the privacy preserving properties of these algorithms, and describes a standard de-anonymization algorithm adapted against the online peer-to-peer algorithms. The empirical performance of these algorithms in non-stationary setting is evaluated in Section 7. Conclusions and future work are provided in Section 8.

2. RELATED RESEARCH

There have been several techniques to parallelize matrix factorization algorithms, including [3, 13]. However, the cited techniques require a central component that has access to the complete model, and all the user preferences (ratings). This drawback is circumvented to some respect in [16] by assigning computational nodes to both users and items, with the items having access to the private data of users that interact with. In their setting the space of items overlaps with that of users, and if items are trusted users, this approach is sufficient. However, in most cases items are handled by one or a few service providers, and the privacy would be compromised using this technique. An approach very similar to ours was proposed in [15] for singular value decomposition (SVD). Each row of the matrix is assigned to a user, along with the corresponding user vectors, and the item matrices are communicated between the items in order to achieve consensus. Our approach is different in that we focus on online ranking prediction (as opposed to SVD on a fixed data set), which enables a more selective broadcasting of item vectors. A theoretical analysis of such selective communication strategies is also provided.

As mentioned before, we are interested in (non-stationary) online prediction with matrix factorization, which has non-convex objective functions. There have been very few papers with theoretical guarantees for non-stationary online optimization for non-convex functions. The few exceptions, such as [22], use some form of a global search, which do not scale very well with the dimensionality of the problem (factor models have relatively high dimensions). For non-stationary stochastic optimization, there exists regret bounds on the online performance when the objective function is convex (see e.g., [6]). Regret bounds were proven for distributed optimization of stationary convex functions as well [9, 12, 29]. Most of these approaches have a periodical consensus step to ensure that the local models are identical. Of these, [9] is perhaps the most interesting on using variance based dynamic communication. Unfortunately, the objective functions are not convex in the parameters of a matrix factorization model, and therefore, these results can not be applied. There have been several papers on optimization of non-convex functions in batch settings, including peer-to-peer stochastic approximation algorithms. The papers that have the mildest assumptions on the communication protocol include [7, 8], which are also the closest to our analysis. They show that the approximation algorithms converge to some local optima when the complete model is sent. We extend their work by facilitating a private part of the model (the user vectors for matrix factorization), and allowing the algorithms to send only a part of the common model (e.g., only the vector corresponding to the rated item).

One of the aims for peer-to-peer online collaborative filtering is improving the privacy of the users. The vulnerability of centrally collected data was also shown by [23]. There have been several attempts to improve privacy of nearest neighborhood (e.g., [20, 10]) or contextual bandit approaches [28], but we choose to focus on matrix factorization. [24] suggests extending the server with a crypto-service provider, which imposes restrictions on the collaborative filtering algorithm, and still requires some trust in the server. An algorithm designed to improve privacy of collaborative filtering data against attacks described in [23] is the k-CoRating algorithm [31]. It extends the rating matrix such that each user has at least $k-1$ peers that rated exactly the same items. The cost of the improved privacy can be a deterioration of the collaborative filtering algorithm that uses that extended data (as shown in our experiments). The peer-to-peer computational architecture adopted by our approach improves privacy without harming prediction performance as shown [30] for convex optimization, and as illustrated by our experiments described in Section 7.3. However, further techniques can be applied to harden the privacy requirements.

3. CONVERGENCE

In this section we show that peer-to-peer online prediction algorithms converge to the same set of local optima as their corresponding centralized algorithms. We consider a more general framework than matrix factorization: a model that has a private and a common component.

Assume that we have N users. At every time step n, user u connects to its local recommender system (RS), and requests some prediction that can be either a rating of an item or a list of top items. The local RS observes some context that includes the item itself in case of rating prediction, as well as any other relevant information. After receiving the prediction the user reveals its preference (i.e. an item and/or the rating of an item). The triplet of user activity, context and user preference will be identified by the random variables X_n, which are supposed to be independent and identically distributed (i.i.d.). X_n may represent the activity of more than one user, if several users are active at the same time.

We consider RS's represented by a model consisting of a user specific vector $\varphi^u \in \mathbb{R}^{d_2}$, and a common part $\theta \in \mathbb{R}^{d_1}$. For instance, in the case of matrix factorization, the former is represented by the user vectors, and the latter by the item vectors. In a peer-to-peer variant of the RS, the local RS will store the vector specific to its user φ^u and a local instance, θ^u, of the common vector.

The performance of the RS is evaluated by a loss function that can be decomposed into some local functions $f^u(\theta^u, \varphi^u, X)$. The loss functions of inactive users are assumed 0. The function depends only on the local component of X (the local context, activity and user preference). The aim is to minimize the global loss function

$$f^*(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}\boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X),$$

where $\boldsymbol{\theta} = (\theta^{1T}, \ldots, \theta^{NT})^T$, $\boldsymbol{\varphi} = (\varphi^{1T}, \ldots, \varphi^{NT})^T$ and $\boldsymbol{f}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) = \sum_{u=1}^{N} f^u(\theta^u, \varphi^u, X)$.

In order to minimize the loss function, after observing the user preference, for each user the local RS will update its parameters in the direction of the negative gradient, as

follows:

$$\begin{aligned}\tilde{\theta}_n^u &= \theta_{n-1}^u - \gamma_n \nabla_\theta f^u(\theta_{n-1}^u, \varphi_{n-1}^u, X_n) \\ \varphi_n^u &= \varphi_{n-1}^u - \gamma_n \nabla_\varphi f^u(\theta_{n-1}^u, \varphi_{n-1}^u, X_n).\end{aligned} \quad (1)$$

Note that for inactive users the local gradients are zero.

To obtain generalization, the local gradient steps are followed by a communication step aimed at improving the consensus on the common part of the model. The communication among users can be represented by a sequence of i.i.d. random matrices $W_n \in \mathbb{R}^{Nd_1 Nd_1}$, and using the notation $\tilde{\boldsymbol{\theta}}_n = (\tilde{\theta}_n^{1T}, \dots, \tilde{\theta}_n^{NT})^T$ we update $\boldsymbol{\theta}_n$ as follows:

$$\boldsymbol{\theta}_n = W_n \tilde{\boldsymbol{\theta}}_n. \quad (2)$$

In the following, first we prove that every θ^u converges to the same limit θ^*. Then we prove that the pair consisting of θ^* and the limit of $\boldsymbol{\varphi}_n$ is a local minimum of $f^*(\theta, \boldsymbol{\varphi})$. Let us introduce some further notations: $\mathbf{1} = (1, 1, \dots, 1)^T \in \mathbb{R}^N$. For $\boldsymbol{x} \in \mathbb{R}^{Nd}$ we write $\langle \boldsymbol{x} \rangle = 1/N(x^1 + \dots + x^N)$ so that

$$\langle \boldsymbol{x} \rangle = \frac{1}{N}(\mathbf{1} \otimes I_d)\boldsymbol{x}, \quad (3)$$

where $\otimes$ denotes the Kronecker product. Further, J stands for the projection onto the 'consensus space':

$$J := (\mathbf{1}\mathbf{1}^T/N) \otimes I_d,$$

whence $J\boldsymbol{x} = \mathbf{1} \otimes \langle \boldsymbol{x} \rangle$. Let us denote by $J_\perp := I_{dN} - J$ the matrix of the orthogonal projection to the consensus space (here $I_{d'}$ denotes the identity matrix of dimension d'). Finally $\boldsymbol{\theta}_{\perp,n} := J_\perp \boldsymbol{\theta}_n$.

Here, we make three assumptions: (1) the communication matrix is such that ensures an averaging process for $\boldsymbol{\theta}_n$, (2) standard conditions on γ_n that enable the stochastic approximation to converge, and (3) the objective function is 'sufficiently nice' for the stochastic approximation.

ASSUMPTION 1. *Let W_n be a sequence of i.i.d. random matrices of size $Nd_1 \times Nd_1$ with non-negative elements. The following conditions hold true:*

- *W_n is row-stochastic for every n: $W_n\mathbf{1} = \mathbf{1}$,*
- *$\mathbb{E}(W_n)$ is column-stochastic for every n: $\mathbf{1}^T\mathbb{E}(W_n) = \mathbf{1}^T$,*
- *the spectral norm ρ of $\mathbb{E}((W_n)^T(I-J)W_n)$ is less than one.*

ASSUMPTION 2. *Let γ_n be a monotone non-increasing sequence of positive numbers that satisfies the following conditions:*

- *$\sum_n \gamma_n^2 < \infty$,*
- *$\sum_n \gamma_n = \infty$, and*
- *$\gamma_n/\gamma_{n+1} \to 1$.*

ASSUMPTION 3. *For any $u = 1, \dots, N$, $f^u(\theta, \varphi, X)$ satisfies the next assumptions:*

- *$f^u(\theta, \varphi, X)$ is continuously differentiable with respect to θ and φ,*
- *$f^u(\theta, \varphi, X)$ is bounded from below and*
- *for all realizations of θ, φ and X the gradient $\nabla_\theta f^u(\theta, \varphi, X)$ satisfies*

$$\sup_\theta \mathbb{E}[|\nabla_\theta f^u(\theta, \varphi, X)|^2] < \infty.$$

THEOREM 1. *If the matrices W_n satisfy Assumption 1, the sequence γ_n satisfies Assumption 2 and the function $f^u(\theta, \varphi, X)$ satisfies Assumption 3, then the variables $\boldsymbol{\theta}_n$ in (2) achieve consensus, that is $\sum_n \mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] < \infty$ and $\boldsymbol{\theta}_{\perp,n} \to 0$ almost surely as $n \to \infty$.*

PROOF. The proof is based on the proof of Lemma 1 in [8]. To shorten the notation we use $\mathbf{Y}_n = -\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_n, \boldsymbol{\varphi}_n, X_n)$, so we have $\boldsymbol{\theta}_n = W_n(\boldsymbol{\theta}_{n-1} + \gamma_n \mathbf{Y}_n)$. Using the row-stochasticity of W_n it is easy to check that $J_\perp W_n J_\perp = J_\perp W_n$. Thus (2) can be written as $\boldsymbol{\theta}_{\perp,n} = J_\perp W_n(\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n)$. Whence we conclude the estimation

$$\begin{aligned}|\boldsymbol{\theta}_{\perp,n}|^2 &= (\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n)^T W_n^T J_\perp W_n(\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n) \\ &\le \lambda_1(W_n^T J_\perp W_n)|\boldsymbol{\theta}_{\perp,n-1} + \gamma_n \mathbf{Y}_n|^2, \end{aligned} \quad (4)$$

where $\lambda_1(M)$ is the largest eigenvalue of the matrix M. Taking the expectation of (4) and using the Cauchy-Schwarz inequality we obtain

$$\begin{aligned}\mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] &\le \rho(W_n^T J_\perp W_n)\mathbb{E}[|\boldsymbol{\theta}_{\perp,n-1}|^2] \\ &+ 2\gamma_n \mathbb{E}[|\mathbf{Y}_n|]\sqrt{\mathbb{E}[|\boldsymbol{\theta}_{\perp,n-1}|^2]} + \gamma_n^2 \mathbb{E}[\mathbf{Y}_n^2].\end{aligned}$$

In order to conclude $\sum_n \mathbb{E}[|\boldsymbol{\theta}_{\perp,n}|^2] < \infty$, we need an additional lemma.

LEMMA 1. *Assume that for the sequence $\gamma_n > 0$, $\gamma_n/\gamma_{n+1} \to 1$, $\sum_n \gamma_n = \infty$, $\sum_n \gamma_n^2 < \infty$ and $0 < \rho < 1$. Let $v_n > 0$ be such that $v_n \le \rho v_{n-1} + \sqrt{v_{n-1}}\gamma_n + \gamma_n^2$. Then $\sum_{n=0}^\infty v_n < \infty$.*

A statement similar to Lemma 1 (with slightly different assumptions) is proved in [8], and we omit the proof for space limitations. The assertion of Theorem 1 then follows from the Borel-Cantelli lemma. □

In Theorem 2 we prove that the algorithm converges to a local minimum. The proof of the convergence is based on the proof of Theorem 1 in [7]. We will need some additional assumptions and a lemma.

ASSUMPTION 4. *There are two finite constants C_1 and C_2 such that for all realizations of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2$, $\boldsymbol{\varphi}$ and X the next assumptions are satisfied:*

- *$|\langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}, X) \rangle - \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_2, \boldsymbol{\varphi}, X) \rangle| < C_1|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2|$,*
- *$|\nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_1, \boldsymbol{\varphi}, X) - \nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_2, \boldsymbol{\varphi}, X)| < C_2|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2|$.*

LEMMA 2. *If the Assumptions 1, 2, 3 and 4 are satisfied then there exist some random vectors $\zeta_{n,\theta} \in \mathbb{R}^{d_1}$ and $\zeta_{n,\varphi} \in \mathbb{R}^{Nd_2}$ which satisfy*

$$\limsup_{k\to\infty} \sup_{l \ge k} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\theta}| = 0 \text{ and} \quad (5)$$

$$\limsup_{k\to\infty} \sup_{l \ge k} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\varphi}| = 0, \quad (6)$$

almost surely, such that

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle + \gamma_n \zeta_{n,\theta}, \quad (7)$$

$$\boldsymbol{\varphi}_n = \boldsymbol{\varphi}_{n-1} - \gamma_n \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n \zeta_{n,\varphi}. \quad (8)$$

PROOF. Rearranging terms, we obtain $\zeta_{n,\varphi} = \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) - \nabla_\varphi \boldsymbol{f}(\boldsymbol{\theta}_n, \boldsymbol{\varphi}_{n-1}, X_n)$. Using Assumption 4 and the inequality $2ab \leq a^2 + b^2$, we get

$$\begin{aligned} |\sum_{i=k}^{l} \gamma_i \zeta_{i,\varphi}| &\leq \frac{1}{2}\sum_{i=k}^{l} |\gamma_i|^2 + \frac{1}{2}\sum_{i=k}^{l} |\zeta_{i,\varphi}|^2 \\ &\leq \frac{1}{2}\sum_{i=k}^{l} |\gamma_i|^2 + \frac{C_2^2}{2}\sum_{i=k}^{l} |\boldsymbol{\theta}_{\perp,i}|^2. \end{aligned}$$

Now (6) follows from Assumption 2 and Theorem 1.
The argument proving (5) is similar to the one in Theorem 1 of [7]. The row-stochastic property of W_n implies $W_n J = J$. Using (1), (2) and (3) we obtain

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \langle Z_n \rangle$$

where $Z_n = W_n \left(\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n^{-1} \boldsymbol{\theta}_{\perp,n-1} \right)$. Rearranging terms, we get

$$\begin{aligned} \langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle &- \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle \\ &+ \gamma_n e_{n,\theta} + \gamma_n \xi_{n,\theta} \end{aligned}$$

where $e_{n,\theta}$ and $\xi_{n,\theta}$ are defined as

$$\begin{aligned} e_{n,\theta} &= \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) \rangle \\ &\quad - \langle W_n(-\nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n^{-1} \boldsymbol{\theta}_{\perp,n-1}) \rangle \\ \xi_{n,\theta} &= \langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \rangle \\ &\quad - \langle \nabla_\theta \boldsymbol{f}(\boldsymbol{\theta}_{n-1}, \boldsymbol{\varphi}_{n-1}, X_n) \rangle. \end{aligned}$$

We can prove $\limsup_{k\to\infty} \sup_{l \geq k} |\sum_{i=k}^{l} \gamma_i \xi_{i,\theta}| = 0$ in the same way as we proved the corresponding statement for $\zeta_{i,\varphi}$. On the other hand, a direct computation shows that $\gamma_i e_{i,\theta}$ is a martingale difference sequence. Furthermore, the corresponding martingale is bounded in L^2, hence converges with probability probability 1 (see e.g. Corollary 2.2 in [14]). That is, $\sum_{k=1}^{\infty} \gamma_k e_{k,\theta}$ exists and is finite almost surely. The lemma follows. □

Let us define

$$\mathcal{L} = \{(\theta, \boldsymbol{\varphi}) : \nabla f^*(\theta, \boldsymbol{\varphi}) = 0\}.$$

In order to guarantee the convergence (i.e. Theorem 2), we need some further criteria:

ASSUMPTION 5.
- *There exists $M_0 > 0$ such that $\mathcal{L} \subset \{(\theta, \boldsymbol{\varphi}) : f^*(\theta, \boldsymbol{\varphi}) < M_0\}$.*
- *There exists $M_1 \in (M_0, \infty)$ such that $\{(\theta, \boldsymbol{\varphi}) : f^*(\theta, \boldsymbol{\varphi}) < M_1\}$ is a compact set.*
- *The interior of the set $f^*(\mathcal{L})$ is empty.*

THEOREM 2. *If the Assumptions 1, 2, 3, 4, 5 are satisfied and θ_0 and $\boldsymbol{\varphi}_0$ are such that $f^*(\theta_0, \boldsymbol{\varphi}_0) < M_0$ then $\boldsymbol{\theta}_n$ and $\boldsymbol{\varphi}_n$ converge with probability 1, moreover*

$$\lim_{n\to\infty} \inf\{|(\langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_n) - (\theta, \boldsymbol{\varphi})|, (\theta, \boldsymbol{\varphi}) \in \mathcal{L}\} = 0.$$

PROOF. Note that $\boldsymbol{\theta}_n = \mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle + \boldsymbol{\theta}_{\perp,n}$. Since in Theorem 1 we proved that $\boldsymbol{\theta}_{\perp,n} \to 0$ almost surely as $n \to \infty$, we only need to examine the convergence of $\langle \boldsymbol{\theta}_n \rangle$. Lemma 2 says

$$\langle \boldsymbol{\theta}_n \rangle = \langle \boldsymbol{\theta}_{n-1} \rangle - \gamma_n \left\langle \nabla_\theta \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_{n-1} \rangle, \boldsymbol{\varphi}_{n-1}, X_n) \right\rangle + \gamma_n \zeta_{n,\theta}$$

$$\boldsymbol{\varphi}_n = \boldsymbol{\varphi}_{n-1} - \gamma_n \nabla_\varphi \boldsymbol{f}(\mathbf{1} \otimes \langle \boldsymbol{\theta}_n \rangle, \boldsymbol{\varphi}_{n-1}, X_n) + \gamma_n \zeta_{n,\varphi}.$$

So we can use Theorem 2.2 and 2.3 from [4] for both $\langle \boldsymbol{\theta}_n \rangle$ and $\boldsymbol{\varphi}_n$ (note that we need Lemma 2 to verify certain assumptions of the cited theorems). The statement of the Theorem 2 follows. □

In Section 5, we will instantiate the peer-to-peer algorithm discussed above to matrix factorization. In the following, we will consider whether the assumption hold for this instance. An example of a communication protocol when the complete model (θ) is sent and satisfies Assumption 1 is provided in [5]. For a matrix factorization model it may be preferable not to send the latent vectors corresponding to all items but just a small subset of the items. The strategies to select the items is discussed in Section 5. The conditions on the learning rate (Assumption 2) are fairly standard for a stochastic optimization algorithm in stationary setting. If the environment is non-stationary (as it is in the experiments), it is standard to switch to a constant learning rate. It is clear that if the loss function is the mean square error, then the function is continuously differentiable with respect to the parameters of the matrix factorization model and is bounded from below. The boundedness of the gradient in Assumption 3 and Assumption 4 hold if the parameters of the model stay bounded in the trajectories starting from an appropriately chosen starting point. This is not necessarily the case for any dataset, but in our experience it seems that the parameters stay bounded unless the initial learning rate is too high. It is clear that if the parameters diverge, they are likely to do so for the centralized algorithm as well, and the parameters needs to be projected into a compact set after the gradient step. In this case, proving Theorem 2 takes a different course, one that mirrors the proof of Theorem 1 of [8]. Assumption 5 also holds, for instance, the condition on the empty interior is satisfied by Sard's theorem if the loss function is Nd_1d_2-times differentiable, which is the case for a matrix factorization model.

4. ONLINE MATRIX FACTORIZATION

In this section we consider the online algorithms for rating prediction and top-K recommendation that form the base of the peer-to-peer algorithms presented in Section 5.

Matrix factorization algorithms constitute the most successful approaches to collaborative filtering [18, 19]. In these algorithms, users and items are mapped into a joint latent factor space of dimensionality d. Accordingly, each user u is associated with a d-dimensional vector U_u, and each item i with a d-dimensional vector V_i. The preference of the user for the specific item is given by the inner product of the two vectors. Additionally, it is usual to have biases specific to users b_u and items b_i. The predicted preference of a user u for item i, $\hat{r}_{ui}$ is given by the following formula,

$$\hat{r}_{ui} = b_u + b_i + U_u^T V_i. \tag{9}$$

There are several ways to tune factor models. In an online scenario, the most natural is the stochastic gradient descent (SGD) [1]. SGD have been a popular choice for matrix factorization algorithms and in online prediction. The online rating prediction algorithm for matrix factorization is presented in Algorithm 1.

At each time step a user connects to the recommender system and selects an item (line 2). The system predicts the rating of the item (line 3), after which the user reveals the 'true' rating and the recommender system suffers a loss

Algorithm 1 Online rating prediction

1: **for** $n = 1$ to T **do**
2: user u connects to RS and selects item i
3: RS predicts rating $\hat{r}_{ui}$
4: user reveals r_{ui} and RS suffers loss $f(r_{ui}, \hat{r}_u i)$
5: $U_u \leftarrow U_u + \gamma(r_{ui} - \hat{r}_{ui})V_i$
6: $b_u \leftarrow b_u + \gamma(r_{ui} - \hat{r}_{ui})$
7: $V_i \leftarrow V_i + \gamma(r_{ui} - \hat{r}_{ui})U_u$
8: $b_i \leftarrow b_i + \gamma(r_{ui} - \hat{r}_{ui})$

Algorithm 2 Online ranking prediction

1: **for** $n = 1$ to T **do**
2: user u connects to RS
3: RS recommends top items $\mathcal{R}$
4: user selects item i (and additionally rating r_{ui})
5: RS suffers loss $f'(i, \mathcal{R})$
6: select negative sample set $\mathcal{N}$
7: $U_u \leftarrow U_u + \gamma(r_{ui} - \hat{r}_{ui})V_i - \gamma \sum_{j \in \mathcal{N}} \hat{r}_{uj} V_j$
8: $b_u \leftarrow b_u + \gamma_t(r_{ui} - \hat{r}_{ui}) - \gamma \sum_{j \in \mathcal{N}} \hat{r}_{uj}$
9: $V_i \leftarrow V_i + \gamma(r_{ui} - \hat{r}_{ui})U_u$
10: $b_i \leftarrow b_i + \gamma(r_{ui} - \hat{r}_{ui})$
11: **for** j in $\mathcal{N}$ **do**
12: $V_j \leftarrow V_j - \gamma \hat{r}_{uj} U_u$
13: $b_j \leftarrow b_j - \gamma \hat{r}_{uj}$

(line 4). The loss function, in our case, is the mean square error between the predicted and true rating. The model parameters are then updated in the direction of the negative gradient with a constant learning rate (line 5–8). In Section 3 we assumed that the learning rate follows a decaying schedule, however, for non-stationary environments, a constant rate is more appropriate.

While rating prediction has a larger literature due to some of the competitions, recommending a list of top items is a more natural task in real applications. Continuing with the SGD set-up, we use the algorithm suggested in [25] that uses the visited item as positive training instance and some randomly sampled unvisited items as negative instances. As shown in Algorithm 2, at each time step, when a user connects to the recommender system, the system ranks the items, and presents the user the top of this list. The items are ranked according to the predicted rating, using equation (9). The system then suffers a loss. There are several choices of ranking measures, a popular one that is used in the experiments is NDCG@K [17]. In our case, there is only one item with non-zero label, and the NDCG@K of a permutation π of the items reduces to

$$\text{NDCG@K}(\pi) = \begin{cases} 1/\log_2(rank_\pi(i) + 1) & \text{if } rank_\pi(i) \leq K \\ 0 & \text{otherwise} \end{cases},$$

where i denotes the visited item, and $rank_\pi(i)$ is the position of the item in the permutation π. In the experiments, for the selection of the negative training instances (line 6 of Algorithm 2), we also consider a mechanism suggested in [32], namely sample randomly a number of unvisited items, and then select only a few items from the top of the sampled list (ranked by the model). The update of the model parameters in Algorithm 2 is shown in lines 7–13.

5. ONLINE PEER-TO-PEER PREDICTION

In Section 3, we presented the framework for a peer-to-peer online prediction algorithm. For the special case of a recommender system using matrix factorization, each user has at hand a local copy of the system, the local model including the latent vector of the user, U_u, the user bias, b_u, a local instance of the item matrix, V^u, and a local instance of the item biases, $b_\cdot^u$. Thus, the user latent vectors and the user biases from Section 4 correspond to φ from Section 3, and the item vectors and biases correspond to θ. The peer-to-peer online recommender is shown in Algorithm 3. At each time step, when a user connects to its local system (line 7), the system makes a prediction. The prediction in the case of rating prediction is estimating the rating of an item, while in the case of recommendation, a list of items. After the prediction, the user reveals its preference, and the system suffers a loss. The preference of the user is the true rating in the case of rating prediction, and a selected item for top-K recommendation. For the latter, the item may or may not be in the recommended list. These steps, shown succinctly in line 8, are identical to the corresponding steps of Algorithm 1 or Algorithm 2. The gradient descent update of the local model (line 9) is also identical to the corresponding centralized variants (including the sampling of negative instances for top-K recommendation). In line 10 the algorithm selects a set of target users and a set of items. Subsequently, the local instance of the latent vectors and biases corresponding to the selected items are sent to the selected set of target users.[1] In a simple variant, a fixed number of users are selected randomly, but more intricate strategies are also possible. A more elaborated method for setting the number of target users is described in Section 7, while users with similar taste could also be given preference. When a social network of users is available, selecting friends as target users seems reasonable for privacy reasons. It is also an easy way to identify users with similar taste. In the case of rating prediction, the natural choice for the item set to be sent is the rated item. While for the top-K recommendation task, the item set should include the item visited by the user and some subsample of the negative set of instances. When the target users receive some vectors, they combine the vectors with their local copies, using a mixing coefficient β (line 11–13).

The conditions for the peer-to-peer algorithm to mirror the centralized algorithm are as follows: (1) the set of target users should include all users, (2) the set of items sent consists of the rated one and, in the case of top-K recommendation, all negative samples, (3) $\beta = 0$, and (4) appropriately chosen initialization for the latent vectors. The last condition requires that the same latent user or item vectors are generated when first encountered in the centralized or peer-to-peer algorithms. If the first three conditions are met, then the item vectors are the same at each user, disregarding items that have not been updated by any user.

So far, we assumed that all users are present in the system from the start of the protocol until the end of it. A more realistic setting is when new users are coming continually in the system. For this setting, Algorithm 3 includes a set-up phase (line 2–6): the new user requests a copy of item vectors

[1]When it is clear from the context, for brevity, we will say sending items instead of sending the corresponding item vectors and biases.

Algorithm 3 Online peer-to-peer prediction

for $n = 1$ to T **do**
  for each new user u' **do**
    select sources $\mathcal{U}'$
    for v in $\mathcal{U}'$ **do**
      $V^{(v)} \leftarrow \frac{1}{|\mathcal{U}'|} \sum_{v \in \mathcal{U}'} V^{(v)}$
      $b^{(v)} \leftarrow \frac{1}{|\mathcal{U}'|} \sum_{v \in \mathcal{U}'} b^{(v)}$
  user u connects to local RS
  local RS predicts rating and suffers loss
  update U, b_u, $V^{(u)}$, $b_i^{(u)}$
  select target user set $\mathcal{U}$ and item set $\mathcal{I}$
  for v in $\mathcal{U}$ and j in $\mathcal{I}$ **do**
    $V_j^{(v)} \leftarrow \beta V_j^{(v)} + (1-\beta) V_j^{(u)}$
    $b_j^{(v)} \leftarrow \beta b_j^{(v)} + (1-\beta) b_j^{(u)}$

and biases from a random set of users already present in the system, and then, averages these copies. When the local copies of item vectors and biases are identical, it is enough to request from only one user. When, due to a limit on the amount of communication, this is not the case, it may be beneficial to request from more users and average their instances. This issue will be revisited in the experimental section.

6. PRIVACY

One of the advantages of the peer-to-peer architecture for recommender systems is that user data is kept locally, preventing an unsolicited party to access it. While the user data cannot be accessed externally, the communication may still reveal some private information.

A similar peer-to-peer algorithm was considered by [30] for online convex optimization, and the authors have shown that such an algorithm has intrinsic privacy-preserving properties, i.e. the gradients of the local function cannot be reconstructed for various topologies of user communication graphs. While in their case, the full model was transmitted, for matrix factorization the user vector are kept locally, and only part of the item vectors are sent at any time. It is easy to see that the inaccessibility of the user vectors and the non-convexity of the function to be optimized makes it even more difficult to reconstruct the gradients.

While the user data is kept locally, and it is sufficiently difficult to reconstruct the gradients, the communication may still reveal information about which items are rated by the user. This vulnerability comes in because the users are sending the vectors corresponding to a restricted subset of the items that contains the rated items. A natural choice for 'attacking' the data made available to a malicious party is the ScoreBoard algorithm of [23]. The algorithm can handle well noise both in the prior information, and the information received from the communication. It assumes that the malicious party gained some auxiliary information about the preferences of a particular person (such as ratings available on IMDB, or preferences stated in a discussion), and attempts to identify the person in the network from the received communications. Subsequently, the algorithm deciphers additional information in the form of preferences for various items.

The ScoreBoard algorithm maintains a score s_u for each user, reflecting how likely is that the information about the particular user matches the auxiliary information. The auxiliary information is represented by the set of items Aux. The algorithm relies on a similarity measure at attribute (item) level. The similarities of the items in the auxiliary set, weighted with a function that depends on their respective frequencies, are combined in the scoring function:

$$s_u = \sum_{i \in Aux} \frac{1}{\log |n_i + 1|} \mathbb{I}_{(r_{ui} > \delta)},$$

where n_u if number of ratings for item i, r_{ui} is the ratio of item i received in the messages received from user u, and δ is appropriate threshold.

7. EXPERIMENTS

In this section we will test empirically the performance of the peer-to-peer online prediction algorithms using two standard benchmarks: the 1M and the 10M MovieLens datasets.[2] The datasets include movie ratings with a timestamp recorded in seconds. We use the timestamps to establish a sequential order of the ratings. Ratings with identical timestamps are considered in random order. The datasets stretch over several years thus there is a fair amount of non-stationarity. The learning rate for all experiments is kept constant 0.05, which was found to perform well for the centralized prediction variants, and kept the same for the distributed ones. The dimensionality of the latent vectors is set to 10.

Since we consider the top-K recommendation task the more interesting one, we focus most of our attention on this task. There is, however, a feature that favours rating prediction: in this task the loss function and the gradient step are closely tied. In contrast, in top-K recommendation the gradient is computed for a loss function that is only related to the true objective measure. Since the convergence results of Section 3 refer to the loss function of which gradient is taken, the first set of experiments (described in Section 7.1) can be thought as a bridge between the theoretical results and the experiments on the top-K recommendation task (Section 7.2). Finally, we describe experiments on privacy in Section 7.3.

7.1 Rating prediction

We consider a 'fixed' communication protocol where the number of target users is kept the same at each time step. Recall from Section 5 that in the case of rating prediction, only the vector and bias corresponding to the rated item are sent in each iteration. The average mean square error (MSE) for the two datasets are shown in Figure 1 and Figure 2. Each data point corresponds to a run where the number of selected targets is kept at a certain value. In both figures the data points with most messages correspond to the situation when the vectors are sent to all users and the loss obtained is the same as the one resulting from a centralized algorithm. We observe that the loss increases logarithmically with decreasing the number of messages sent (target users selected). In the figures two simple dynamic schedules that depend on the suffered loss (MSE) are included as well: namely the number of target users selected equals $(N-1)/(1+c/f_n)$, where f_n is the loss at time step n, and c is constant that is varied in the figure. One can notice that for both datasets a slight improvement can be obtained with the dynamic communication schedule.

[2]http://grouplens.org/datasets/movielens/

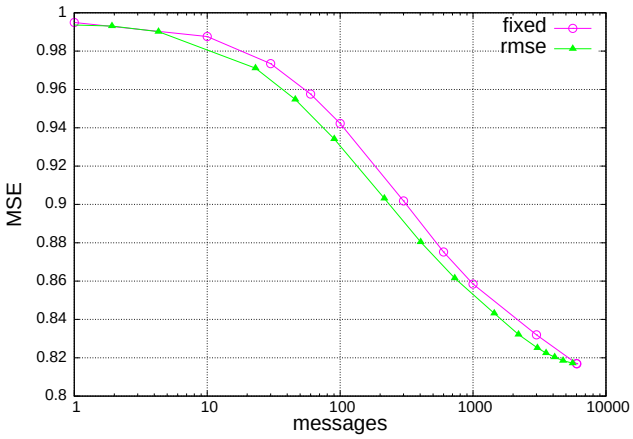


Figure 1: Peer-to-peer rating prediction on the 1M MovieLens dataset. The x-axis shows the average number of item vectors sent per time step.

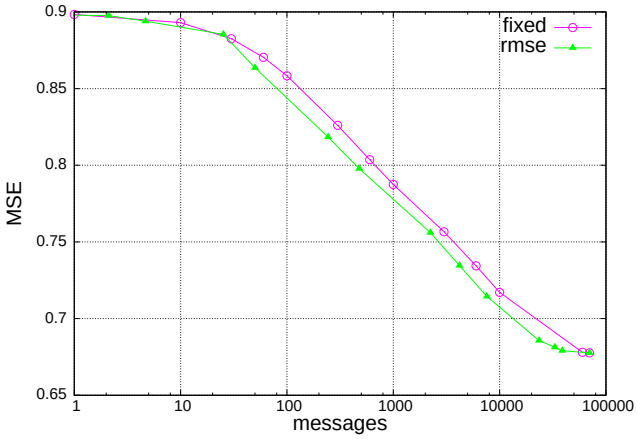


Figure 2: Peer-to-peer rating prediction on the 10M MovieLens dataset.

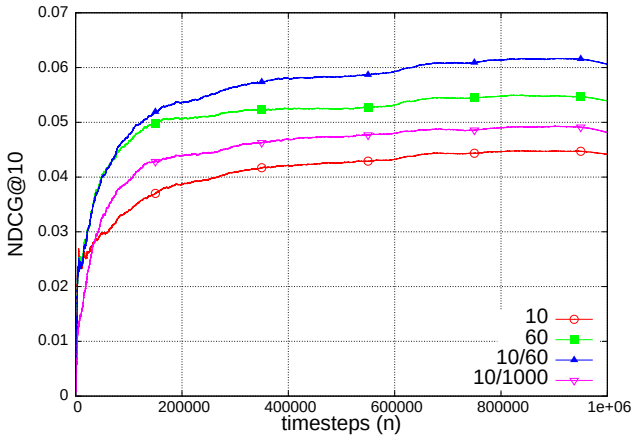


Figure 3: Centralized top-K recommendation on the 1M MovieLens dataset with different strategies to select negative training instances.

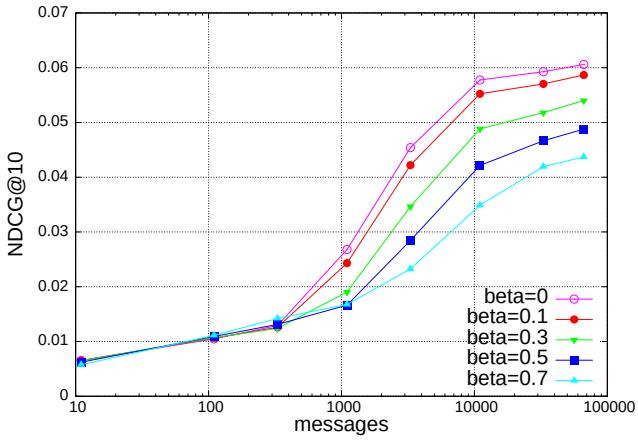


Figure 4: Peer-to-peer top-K recommendation on the 1M MovieLens dataset with varying mixing rate.

7.2 Top-K recommendation

For the top-K recommendation, first, we choose the strategy of selecting negative samples (line 6 in Algorithm 2). In Figure 3 we plot the performance of four strategies on the 1M dataset: (1) select randomly 10 unvisited items, (2) select randomly 60 items, (3) select randomly 60 items and use only the top ten of these, and (4) select to top ten of 1000 random ones. The performance is measured as average cumulative NDCG@10. More precisely, at each time step n, we sum the instantaneous NDCG@10 up to that point, and divide it with n. While in Algorithm 2 we refer to the measure as loss function, for NDCG the higher is the better. The performance seems very sensitive to the chosen negative sampling. The best performing strategy is to select randomly 60 item, and use the top 10 for the update step. We will use this strategy for the remaining set of experiments.

Turning to the peer-to-peer recommendation, the first question is how to mix the received vectors with the local ones (β in Algorithm 3). In this experiment the vectors of the positive instance and all negative instances are sent to the target users. The number of target users is fixed (in a similar way as in the fixed protocol above) to a particular value that is varied. Figure 4 shows the performance on the 1M dataset with different values for β. The average cumulative NDCG@10 is shown in this figure and the subsequent ones for the whole dataset. Thus, the data point on these figures correspond to the last measurement in Figure 5 (i.e. for the data point with largest number of time steps). The performance of the centralized algorithm is indeed identical to the performance of the peer-to-peer algorithm for the data point that satisfies the conditions enumerated in Section 5. Recall that the conditions include sending all negative items to all users and having $\beta = 0$. It is clear from Figure 4 that the best result is obtained for $\beta = 0$, in which case the received latent vector and bias replaces the local instance. In the following we will use this setting.

The next experiment investigates if all (10) negative instances should be sent, or just a random subset of these, along with the positive one. Therefore, in Figure 5 we plot the performance if 3, 5, 7, or 10 (all) negative instances are sent. Note that if we select the same number of target users and send only vectors corresponding to five negative instances the amount of communication is reduced (almost halved). The results for the four values are rather interest-

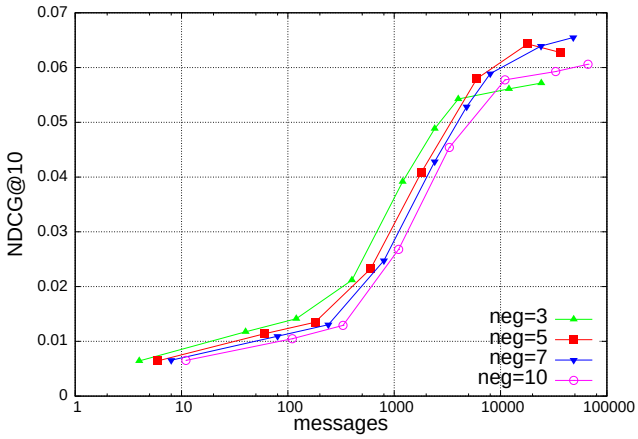


Figure 5: Peer-to-peer top-K recommendation on the 1M MovieLens dataset with varying item selection.

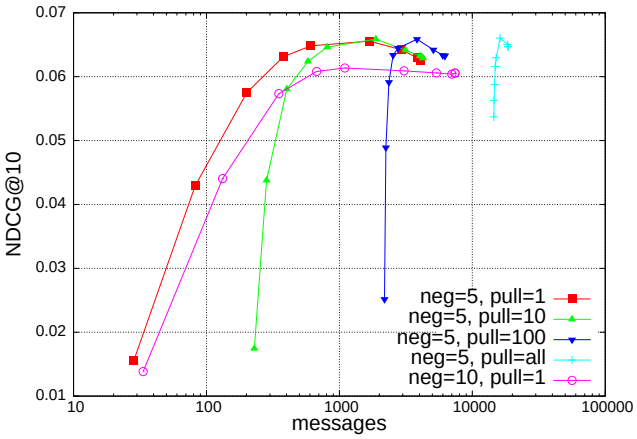


Figure 6: Peer-to-peer top-K recommendation on the 1M MovieLens dataset in the case of dynamic user availability.

ing. First, it is clear that if the number of messages are kept small, then it is better to select less items and use the reduction in the message to send to more users. Surprisingly, when all (or almost all) users are targets, performance gain can be obtained by not sending all negative items. In fact, this results in a performance gain over the centralized algorithm. We do not have a clear explanation for the improvement, it could be due to some additional diversity among the copies of the models, but this is purely a conjecture.

Up to this point, we assumed that users were in the system during the whole episode. In the final experiment we consider the situation when this is not the case. It is difficult to know from the data when a particular user was active, and we set this time frame as ranging from the first rating of the user until the last one (naturally the user could have registered much earlier, and could have been using the system for much later without performing any rating). In this scenario lines 2–6 of Algorithm 3 become relevant, and we have to decide from how many users to 'pull' the model. The results are shown in Figure 6 with different number of source users. The performances are shown for the case when five negative instances are sent after a gradient update (a combination of all negatives sent and one source user polled is also shown). It seems obvious from the figure that the communication efficient strategy is to request the model from only one user. If we send to the same number of target users (after gradient update) pulling models from more users increases the performance, but it comes at a cost of increased communication that can be better used after the gradient update.

For top-K recommendation, the results were presented for the 1M dataset. We observed a similar pattern of results for the 10M dataset (not shown). Finally, we would like to draw some comparison between how the performance degrades with decreasing communication for the two types of tasks, i.e. rating prediction and top-K recommendation. If we compare Figures 1 and 2 with Figures 4 and 5, the curves seems fairly similar taken into account that for the first task we minimize the measure, while for the second, we maximize. Figure 6 shows a different pattern. We observe a plateau at the end of each curve, clearest being for the curve corresponding to sending all negative samples and requesting from only one user (neg=10, pull=1). This suggests that if at each time step we are only concerned to send items that are active, then we do not lose in performance even if we the amount of communication drops by an order of magnitude.

7.3 Privacy

The privacy preserving properties of the peer-to-peer recommendation system is tested with the ScoreBoard algorithm described in Section 6. We consider the top-K recommendation task on the 1M MovieLens dataset with varying item selection and with continuous presence of all users (thus the same set-up as for Figure 5). We consider two scenarios: when the adversary knows 10 percent of the items rated by a particular user and when all ratings of the user are known. The probability of correctly identifying the user from the messages received is averaged over all users. If the adversary has access to the whole data set the detection probability is 0.48 and 0.96, respectively. The detection probability against the peer-to-peer algorithm is plotted in Figure 7 and Figure 8. In both figures the threshold δ was set to 0.5 which was found to result the highest probability of identifying correctly the users. The value plotted was chosen as the highest value at any time for a given set-up. In both scenarios (10 percent and all ratings) the detection probability is hugely decreased compared to the above mentioned baselines. Sending more negative items increases privacy (i.e. decreases detection probability) as well as sending to items to less peers.

Next, we compare the ranking performance/privacy tradeoff for the peer-to-peer algorithm and the centralized one using the k-CoRating algorithm to improve privacy. k-CoRating extends the rating matrix in a heuristic way such that for any user there are at least $k-1$ other users that rated exactly the same items. In an online scenario this is fairly difficult to achieve, but we assume that there is an oracle that knows in advance all the ratings and extends the rating matrix with ratings such that the above property will hold at the end. The artificial ratings are spread uniformly over the episode, and set to an average value.[3] The training for the centralized algorithm is performed on the extended

[3]The rating value is not critical since the ranking performance is not sensitive to the actual rating (cf. the equation of NDCG@K in Section 4).

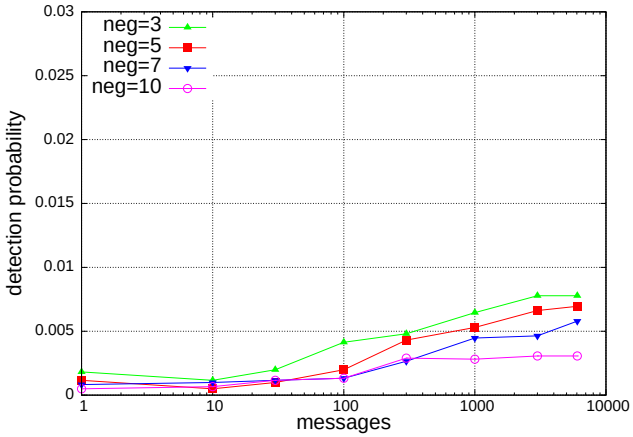


Figure 7: Detection probability on the 1M MovieLens dataset when 10% of the ratings are known for a particular user.

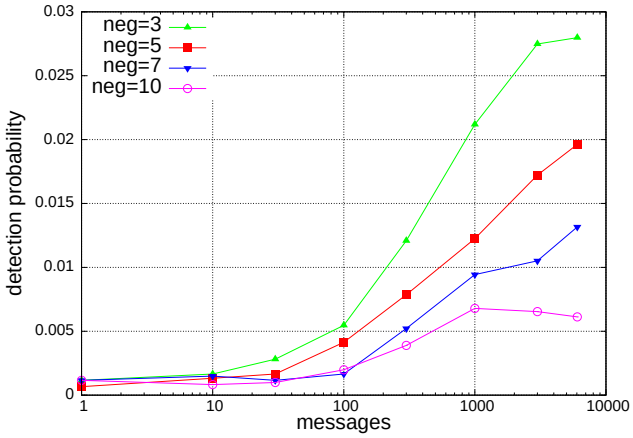


Figure 8: Detection probability on the 1M MovieLens dataset when all ratings are known.

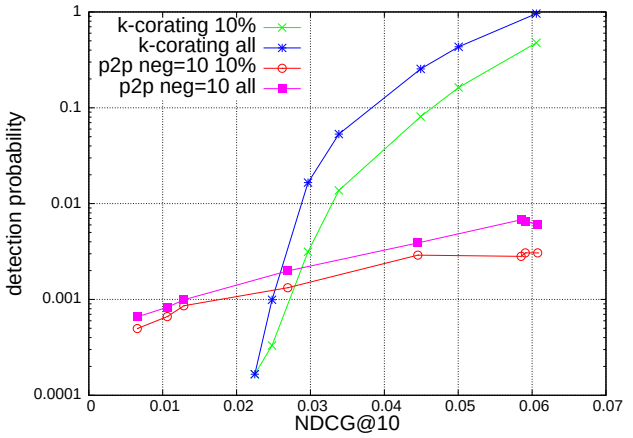


Figure 9: Detection probability vs. ranking performance.

rating sequence, the ranking performance is measured on the original sequence, and the detection probability (with the ScoreBoard algorithm) on the extended sequence.

The detection probability corresponding to various levels of ranking performance is plotted in Figure 9. For the peer-to-peer algorithm, we included the results with all negative items sent, with the number of target users varied (as in Figure 7 and 8, while for the centralized algorithm with k-CoRating the value k is varied. Again the adversary knows all ratings of a user or ten percent of it. We observe that to obtain a reasonable ranking performance k-CoRating is unable to ensure high level of privacy (values of k close to 1), while the peer-to-peer algorithm can achieve a good ranking performance and in the same time offer considerable privacy. It is interesting that k-CoRating can achieve perfect privacy and still provide a meaningful ranking, while the peer-to-peer algorithm is unable to achieve this in the considered version. It is expected that if more items are sent (other than the positive and negative samples) the privacy can be further improved without deteriorating the ranking performance. This comes however with the expense of increased communication costs.

8. CONCLUSIONS

In this paper we proposed an online peer-to-peer collaborative filtering algorithm that stores the ratings of a user and the private data local. Additionally, the users have a local copy of the common part of the factor model and communicate with other users to advance towards a consensus on it. A general form of the algorithm is proven to converge to a set of local optima in the stationary case. In a more specific form, we provided peer-to-peer matrix factorization algorithms for rating prediction and top-K recommendation.

We note that the general form is straightforward to instantiate to more intricate models such as factorization machines [26], or extend with context-aware features [2]. In the same way, it is easy to replace the negative sampling and gradient update step in the top-K recommendation algorithm with other choices employed in a centralized algorithm, e.g., [27] or [32]. Essentially, most algorithms where the private data can be separated, and use some form of stochastic gradient descent can be distributed in the same way as we did with our matrix factorization model in Section 5.

The online peer-to-peer matrix factorization algorithms were evaluated on the two larger MovieLens datasets that seem to us fairly non-stationary. We observed that the mean square error increases logarithmically with decreasing communication, and we suggested a simple way to control the amount of communication more efficiently. We expect that targeting users with similar tastes could make the communication more efficient, but we leave this issue for future research. For top-K recommendation we compared a few negative sampling strategies, and we showed how to handle new users. Crucially, we observed that sending only a subset of the negative samples could not only make the communication more efficient, but it could improve on the performance of the centralized algorithm. Understanding why the improvement is possible could help to improve centralized recommendation algorithms as well.

The peer-to-peer recommendation algorithm was shown to offer improved privacy without deteriorating the recommendation performance, which was not the case for the baseline algorithm tested.

9. REFERENCES

[1] J. Abernethy, K. Canini, J. Langford, and A. Simma. Online collaborative filtering. Technical report, UC Berkeley, Tech. Rep, 2011.

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

[3] M. Ali, C. C. Johnson, and A. K. Tang. Parallel collaborative filtering for streaming data. *University of Texas Austin, Tech. Rep*, 2011.

[4] C. Andrieu, É. Moulines, and P. Priouret. Stability of stochastic approximation under verifiable conditions. *SIAM Journal on control and optimization*, 44(1):283–312, 2005.

[5] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, 2009.

[6] O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *arXiv preprint arXiv:1307.5449*, 2013.

[7] P. Bianchi, G. Fort, and W. Hachem. Performance of a distributed stochastic approximation algorithm. *IEEE Trans. on Information Theory*, 59:7405–7418, 2013.

[8] P. Bianchi and J. Jakubowicz. Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization. *IEEE Transactions on Automatic Control*, 58(2):391–405, 2013.

[9] M. Boley, M. Kamp, D. Keren, A. Schuster, and I. Sharfman. Communication-efficient distributed online prediction using dynamic model synchronizations. In *First International Workshop on Big Dynamic Distributed Data (BD3@VLDB)*, pages 13–18, 2013.

[10] A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. Privacy-preserving distributed collaborative filtering. In *Networked Systems*, pages 169–184. Springer, 2014.

[11] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, pages 1–53, 2013.

[12] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research*, 13:165–202, 2012.

[13] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–77. ACM, 2011.

[14] P. Hall and C. C. Heyde. *Martingale Limit Theory and Its Application*. Academic Press, 1980.

[15] I. Hegedus, M. Jelasity, L. Kocsis, and A. A. Benczúr. Fully distributed robust singular value decomposition. In *14-th IEEE International Conference on Peer-to-Peer Computing*, pages 1–9. IEEE, 2014.

[16] S. Isaacman, S. Ioannidis, A. Chaintreau, and M. Martonosi. Distributed rating prediction in user generated content streams. In *Proc. Fifth ACM Conf. on Rec. Sys.*, pages 69–76. ACM, 2011.

[17] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48, 2000.

[18] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[19] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[20] N. Lathia, S. Hailes, and L. Capra. Private distributed collaborative filtering using estimated concordance measures. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 1–8. ACM, 2007.

[21] N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 796–797. ACM, 2009.

[22] O.-A. Maillard and R. Munos. Online learning in adversarial lipschitz environments. In *Machine Learning and Knowledge Discovery in Databases*, pages 305–320. Springer, 2010.

[23] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy, 2008.*, pages 111–125. IEEE, 2008.

[24] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *ACM SIGSAC Conf. on Comp. and Comm. Security*, pages 801–812. ACM, 2013.

[25] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Eighth IEEE International Conference on Data Mining (ICDM'08)*, pages 502–511. IEEE, 2008.

[26] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[27] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[28] C. Tekin, S. Zhang, and M. van der Schaar. Distributed online learning in social recommender systems. *arXiv preprint arXiv:1309.6707*, 2013.

[29] K. I. Tsianos and M. G. Rabbat. Consensus-based distributed online prediction and optimization. In *IEEE GlobalSIP Network Theory Symposium*, 2013.

[30] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi. Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2483–2493, 2013.

[31] F. Zhang, V. E. Lee, and R. Jin. k-corating: Filling up data to obtain privacy and utility. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[32] W. Zhang, T. Chen, J. Wang, and Y. Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 785–788, 2013.

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

Statistical analysis of NOMAO customer votes for spots of France

Róbert Pálovics[1,2], Bálint Daróczy[1,2], András Benczúr[1,3], Julia Pap[1], Leonardo Ermann[4], Samuel Phan[5], Alexei D. Chepelianskii[6], and Dima L. Shepelyansky[7]

[1] Informatics Laboratory, Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Pf. 63, H-1518 Budapest, Hungary
[2] Technical University Budapest, Hungary
[3] Eötvös University Budapest, Hungary
[4] Departamento de Física Teórica, GIyA, CNEA, Av. Libertador 8250, (C1429BNP) Buenos Aires, Argentina.
[5] NOMAO.COM, 1 av Jean Rieux, 31500 Toulouse, France
[6] LPS, Université Paris-Sud, CNRS, UMR 8502, F-91405, Orsay, France
[7] Laboratoire de Physique Théorique du CNRS, IRSAMC, Université de Toulouse, UPS, F-31062 Toulouse, France

Dated: April 30, 2015

Abstract. We investigate the statistical properties of votes of customers for spots of France collected by the startup company NOMAO. The frequencies of votes per spot and per customer are characterized by a power law distributions which remain stable on a time scale of a decade when the number of votes is varied by almost two orders of magnitude. Using the computer science methods we explore the spectrum and the eigenvalues of a matrix containing user ratings to geolocalized items. Eigenvalues nicely map to large towns and regions but show certain level of instability as we modify the interpretation of the underlying matrix. We evaluate imputation strategies that provide improved prediction performance by reaching geographically smooth eigenvectors. We point on possible links between distribution of votes and the phenomenon of self-organized criticality.

PACS. 89.75.Fb Structures and organization in complex systems – 89.75.Hc Networks and genealogical trees – 89.20.Hh World Wide Web, Internet

1 Introduction

The young startup company NOMAO [20] collected a large database about customer (or users) votes for spots (or Points of Interest POIs or items) in France. The spots represent mainly restaurants and hotels with known geolocation coordinates. In this paper we investigate the statistical properties of these NOMAO votes and ratings of geolocalized items in a mix of geographic information and recommendations systems. The geographical distributions of votes are shown in Fig. 1 for the whole France and more specifically for Paris. The frequency distributions of votes per spot and votes per user are shown in Fig. 2 for France at different time intervals. It shows that these frequency distributions are stabilized in time and thus we are dealing with an unusual statistical system been at a certain steady-state. We note that at present a variety of real systems and networks are found to possess power law distribution (see e.g. [10]) and thus here we investigate a new type of such a case with algebraic statistical properties.

To analyze the statistical properties of this real system we use the methods of recommender systems [29] which gained a broad recognition in computer science after the Netflix Prize competition [28]. In our research, distance, region and location become a side information over a multi-objective classification or regression problem. We concentrate on predicting user preferences by a spectral analysis based collaborative filtering that uses geolocation in addition to the ratings matrix.

We investigate how user taste, as described by latent factors, is reflected in the geographic information system. We compare the latent factors obtained by a full spectral analysis and by the stochastic gradient method, the standard recommendation technique applicable for matrices with a very large fraction of values missing.

The key difficulty in the spectral analysis lies in the abundance of missing values in the rating matrix: our matrix consists of 99.5% missing values while the Netflix matrix for example is 99% unknown. Several early results describe expectation maximization based singular value decomposition (SVD) algorithms dating back to the seventies [13] and [6,23,34] describe the method for a recommender application.

A successful implementation of spectral analysis in recommender matrices with only a few known elements is described by Simon Funk in [12]. His method is a variant of

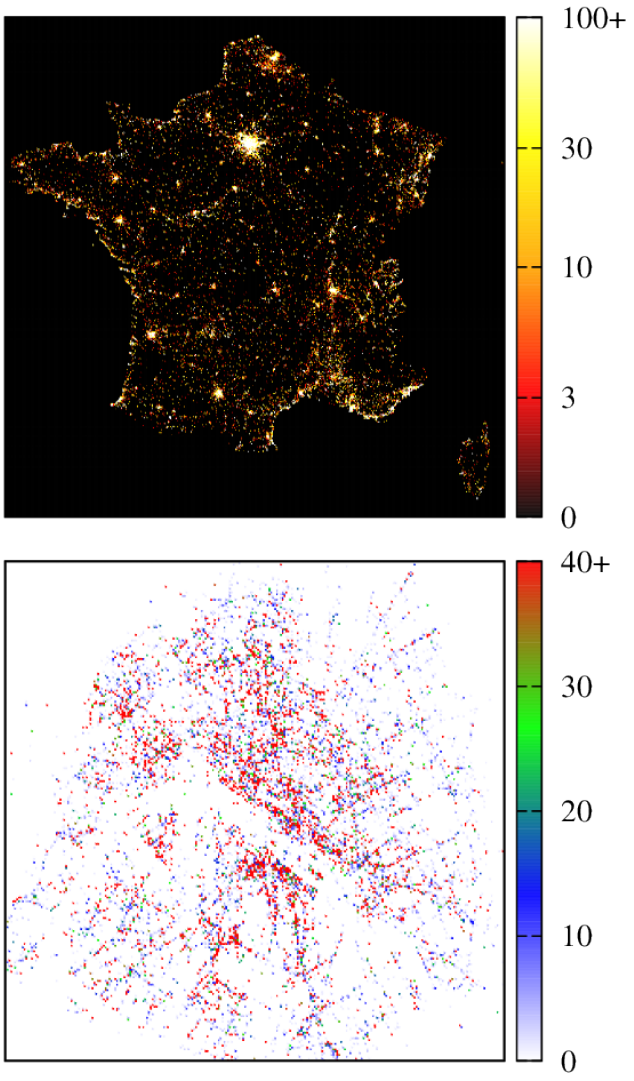


Fig. 1: Geographical distribution of votes for spots (POIs) in the original datasets. Top panel: case of France; (each square pixel represents $7.8km^2$); bottom panel: case of Paris (each square pixel represents $1370m^2$); color bars give a number of votes per pixel (cell), a limitation in number of votes is introduced for a better color representation.

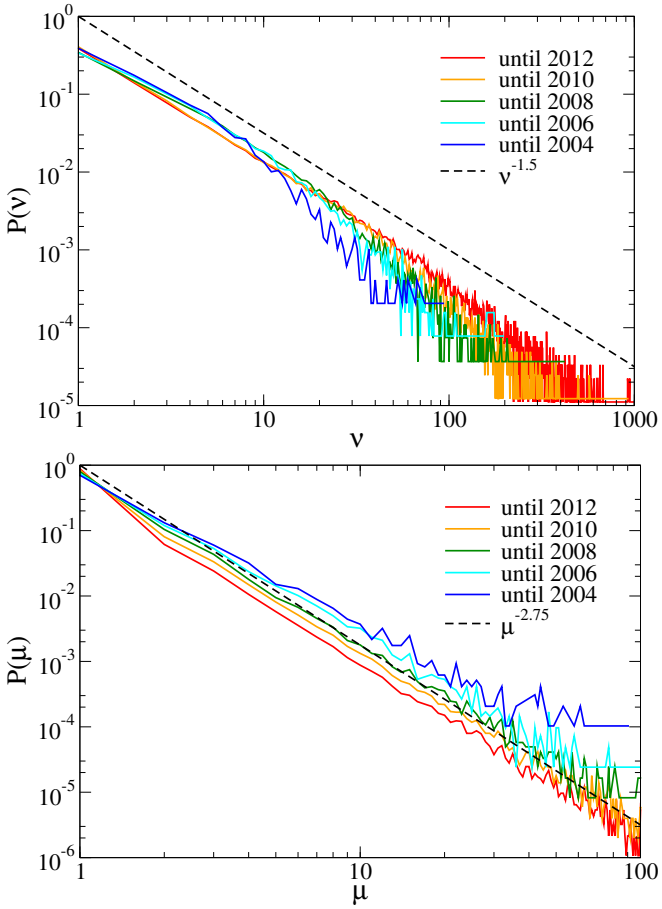


Fig. 2: Differential frequency distributions of votes for the case of France for different time intervals until year 2004, 2006, 2008, 2010, 2012. Top panel: differential probability $P(\nu)$ to have ν votes for spots; bottom panel: differential probability $P(\mu)$ to have μ votes per user (customer). Here the dashed lines show an average algebraic decay with exponents -1.5 (top), -2.75 (bottom).

Stochastic Gradient Descent (SGD) reminiscent of gradient boosting [11]. SGD computes no eigenvalues and does not guarantee the orthogonality of the matrix factors. On the other hand, regularization is easily incorporated in SGD, which enables a better handling of the very large amount of missing values in the matrix and in particular, prevents overfit to training elements and provide better quality predictions of the unknown ratings.

In this paper, after describing methods and related results (Section 2) and the NOMAO data sets (Section 3), we compare and visualize the geo-localization of the matrix factors defined by SVD and SGD under various parameter settings in Section 4. We show that by imputing ratings to nearby locations we may form factors that yield a better description of the ratings matrix in Section 5.

2 Methods and related results

Recommenders based on the rank k approximation of the rating matrix based on the first k singular vectors are probably first described in [5,21,15,22] and many others near year 2000.

The Singular Value Decomposition (SVD) of a rank ρ matrix R is given by $R = U^T \Sigma V$ with U an $m \times \rho$, Σ a $\rho \times \rho$ and V an $n \times \rho$ matrix such that U and V are orthogonal. By the Eckart-Young theorem [14] the best rank-k approximation of R with respect to the Frobenius norm is

$$||R - U_k^T \Sigma_k V_k||_F^2 = \sum_{ij} (r_{ij} - \sum_k \sigma_k u_{ki} v_{kj})^2, \quad (1)$$

where U_k is an $m \times k$ and V_k is an $n \times k$ matrix containing the first k columns of U and V and the diagonal Σ_k containing first k entries of Σ.

The RMSE differs from the above equation only in that summation is over known ratings

$$\text{RMSE}^2 = \sum_{ij \in \text{known}} \text{err}_{ij}^2 \text{ where } \text{err}_{ij} = r_{ij} - \sum_k \sigma_k u_{ki} v_{kj}. \quad (2)$$

Early works [22] used SVD for recommenders by defining various strategies for handling the missing values in the rating matrix R [18]. The most natural idea is to impute the missing elements by zeroes, averages, or even repeatedly re-fill by predictions. It has turned out that all above missing value imputation methods overfit to the imputed values [18]. More recent results emphasize the importance of regularization to avoid overfitting [3,25]. For this reason, the recommender systems community turned away from SVD and use other optimization methods for rating matrices with missing values, most notably stochastic gradient descent [26] and alternating least squares [16].

In our problem, locality is an additional information that can be exploited for analyzing the recommender matrix. Surveys on recommendations in location-based social networks [2,24] combine spatial ratings for non-spatial items, nonspatial ratings for spatial items, and spatial ratings for spatial items [19]. Flickr geotags are used for travel route recommendation, concentrating on routes and not individual places in [17]. User similarity based methods may combine friendship information with the distance of the user home locations [31,32].

Most similar to our method is the Probabilistic Matrix Factorization approach that fuses geographic information [7] and observes that "users tend to check in around several centers, where the check-in locations follow a Gaussian distribution at each center [. . . and] the probability of visiting a place is inversely proportional to the distance from its nearest center; if a place is too far away from the location a user lives, although he/she may like that place, he/she would probably not go there."

3 The Nomao Datasets

Nomao is a startup company located in France [20]. It performs the analysis of point of interest (POI) rating and reservation services and collects POI information including user ratings from France with a special accent on Paris and Toulouse regions where the company headquarters are located. The Nomao dataset used in our experiments contain user-POI ratings, and GPS information of the rated POIs. We investigate two separated datasets. The first one contains information on POIs in France, while the second has ratings only on POIs located in Paris. We analyze the datasets collected during the time period up to year 2012.

Table 1 (**top**) shows the basic attributes of the original datasets. The average number of ratings per item is relatively large, the average number of ratings per user is very low. Moreover, only a very few percent of all user-item scores is known.

Table 1: Attributes of the original (**top**), and cleaned (**bottom**) datasets.

original	Paris	France
Number of ratings	1,539,964	1,432,601
Number of users	998,127	1,077,568
Number of items	20,576	99,976
Average ratings per user	1.543	1.329
Average ratings per item	74.84	14.32
Ratio of known ratings	0.0075%	0.0013%
cleaned	Paris	France
Number of ratings	114,352	97,452
Number of users	5,756	9,471
Number of items	2,952	7,605
Average ratings per user	19.87	10.29
Average ratings per item	38.74	12.81
Ratio of known ratings	0.672%	0.135%
Average rating	3.714	3.747

The distribution $P(\nu)$ of frequency of votes per spot ν (or item i) is shown in top panel of Fig. 2. This distribution is stable in time and is well described by the power law $P(\nu) \propto 1/\nu^a$ with $a \approx 1.5$. Also, the distribution $P(\mu)$ of frequency of votes per customer μ (or user u) remains stable in time with the power law dependence $P(\mu) \propto 1/\mu^b$ with $b \approx 2.75$. It is important to note that this distributions remain stable from year 2004 up to year 2012 even if the number of votes increases almost by two orders of magnitude during this period. At the moment we cannot prodive theoretical reasons for the values of these exponents.

We call user activity how many times a user scored different items. We define item activity similarly. Fig. 3 shows the probability density function (PDF), an the cumulative density function (CDF) of user activities. Fig. 4 shows the same distributions for items. Both user and item activities follow power-law distributions with the exponent values being very similar for Farnce and Paris datasets. As in Fig. 2 we find that the exponent for probability of votes for POIs is $a \approx 1.5$ while the exponent for the exponent for probability of votes of users is $b \approx 2.75$.

To handle the extreme sparsity of the user-item matrices, we selected a smaller subset of the user-item rating datasets by the following selection criteria:

- We only used ratings between 0-5. Part of the ratings, probably originating from a different system, were out of this range.
- We filtered out users and items that have less than A ratings. In other words, we selected the subgraph of the user-item rating bipartite graph with users and items that have degree at least A. For Paris we set $A = 10$, for France we set $A = 5$.

Table 1 (**bottom**) shows the attributes of the selected subsets. In what follows we use these datasets in our experiments.

In Fig. 5 we show the score distributions: the top (bottom) panel shows the distributions for the original (cleaned) datasets. We see that the original and cleaned datasets have similar distributions of scores.

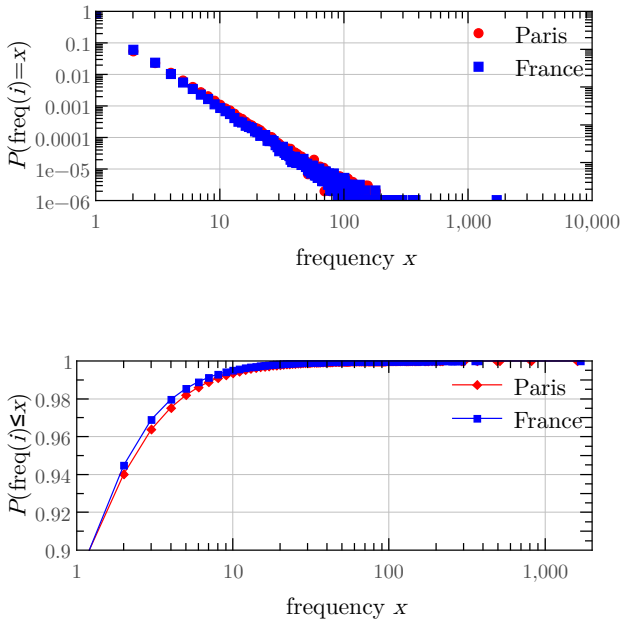


Fig. 3: Probability density function (**top**), and cumulative density function (**bottom**) of user activities in the original datasets.

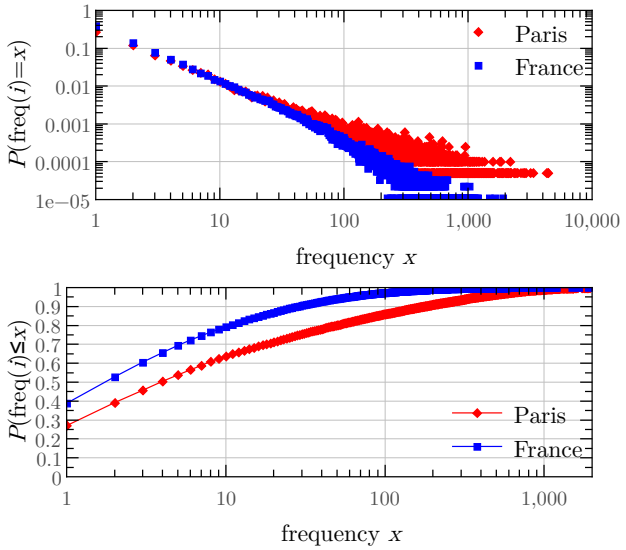


Fig. 4: Probability density function (**top**), and cumulative density function (**bottom**) of item activities in the original datasets.

Fig. 6 shows the geographical density of POIs in the Paris (top) and in the France (bottom) for the original datasets. The geolocation data of POIs are used in the following Sections for spectral analysis.

In the following we perform all computations with the cleaned datasets since the analysis of multiple votes of the same user provides more reliable statistical data.

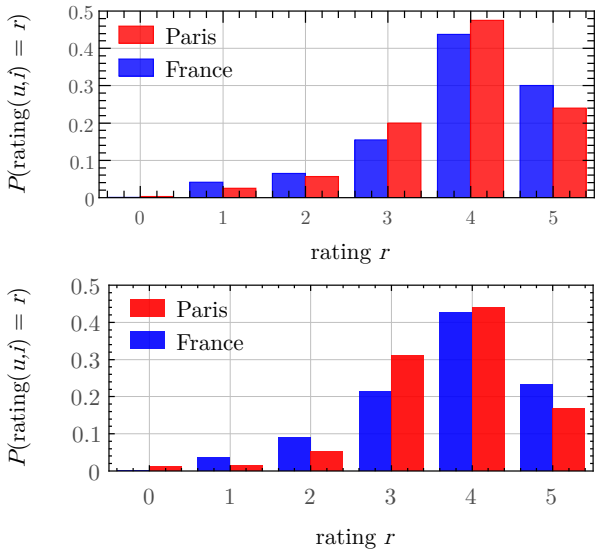


Fig. 5: Score distributions with integer binning. **Top**: original dataset, **Bottom**: cleansed dataset.

4 Spectra of recommender matrices

4.1 Singular value decomposition

The recommender matrix R consists of the preference values $r(u,i)$ of users u for items i. The values may denote *explicit* rating values, e.g. 1-5 stars for Netflix movies [3]. We may also consider the so-called *implicit* ratings problem, where the value is 1 if the user visited POI i and 0 otherwise. The value of the explicit matrix is missing whenever the user gave no rating yet. In most of the cases, this matrix is very sparse with only 1% or less known values. The implicit matrix is always a full 0–1 matrix, however the 0 values are uncertain: the user may not know about the item or had no time yet to visit it.

The so-called *Latent Factor Model* is an approximation $\hat{R}$ of the original rating matrix R,

$$\hat{r}(u,a) = \sum_{f=1}^{k} p_{uf} q_{af}, \tag{3}$$

where $P = [p_{uf}]$ and $Q = [p_{uf}]$ are the user and item factor models, respectively.

For a fixed number of factors k, $\hat{r}$ approximates r with the smallest root mean squared error if it is defined by the singular vectors corresponding to the k largest singular values,

$$\hat{r}(u,a) = \sum_{f=1}^{k} p_{uf} q_{af}, \tag{4}$$

where the singular value decomposition (SVD) of R is $U\Sigma V^T$.

Since

$$RR^T = U\Sigma^2 U^T \text{ and } R^T R = V\Sigma^2 V^T, \tag{5}$$

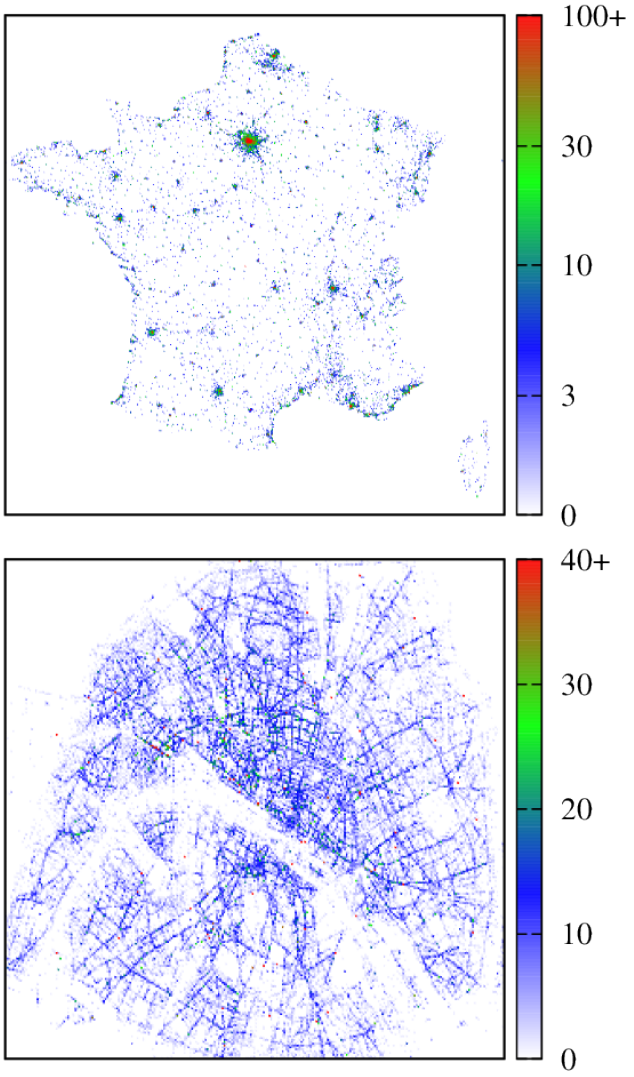


Fig. 6: Geographical distribution of POIs in the original datasets. Top panel: case of France (each square pixel represents $7.8km^2$); bottom panel: case of Paris (each square pixel represents $1370m^2$); color bars give number of POIs per pixel (cell); a limitation in number of POIs is introduced for a better color representation.

the spectrum of the recommender matrix R is defined identically by the square root of the eigenvalues of RR^T or R^TR. These latter matrices are symmetric positive semidefinite, the spectrum is non-negative real.

If R contains missing values such as in the case of an explicit rating matrix, SVD is undefined. We may still define the best root mean square approximation by summing the error for the known ratings only as in equation (1).

4.2 Stochastic gradient descent: Latent factor modeling with missing values

We use the regularized matrix factorization method of [25]. and optimize the minimum squared error of the k-factor model

$$\hat{r}(u,i) = \sum_{l=1}^{k} p_{ul} q_{il}, \tag{6}$$

where p and q contain the user and item models, respectively. By adding regularization with weight λ, we optimize the quantity

$$\sum_{u,i} \left(r(u,i) - \sum_{l=1}^{k} p_{ul} q_{il} \right)^2 + \lambda \sum_u \sum_{l=1}^{k} p_{ul}^2 + \lambda \sum_i \sum_{l=1}^{k} q_{il}^2. \tag{7}$$

For a single event (u,i) we optimize the coefficients p_{ul} and q_{il} for $l = 1, \ldots, k$ by gradient descent with learning rate lrate as

$$p_{ul} \leftarrow p_{ul} + \text{lrate} \cdot \left(r(u,i) - \sum_{l=1}^{k} p_{ul} q_{il} \right) q_{il} - \text{lrate} \cdot \lambda p_{ul}; \tag{8}$$

$$q_{il} \leftarrow q_{il} + \text{lrate} \cdot \left(r(u,i) - \sum_{l=1}^{k} p_{ul} q_{il} \right) p_{ul} - \text{lrate} \cdot \lambda q_{il}. \tag{9}$$

Unlike SVD where eigenvalues are sorted, the SGD factors are not ordered by the above equations. In order to produce the eigenvector maps, we built ranked factors by an iterative SGD that optimize only on a single factor at a time [12].

4.3 Mapping SVD and SGD latent factors

First we set each unknown value of R to zero and computed the SVD decomposition. The first, second, and fourth singular vectors are plotted over the map of France (Fig. 7, left) by assigning the value in the vector to the location of the POI. More specifically, we averaged these values on a grid to create the final heatmaps. The smoothing algorithm weighted the value of each POI to the closest grid point inversely proportional to their euclidean distance.

The heatmaps in Fig. 7, left, indicate that the singular vectors are strongly geolocation related. The first few dimensions correspond to the largest cities in France.

Similarly, we investigated the latent vectors of R computed with the SGD algorithm. The first, second and fourth latent vectors are plotted over the map of France in Fig. 7, right, similar to the SVD eigenvectors. While the SVD singular vectors were centralized one-by-one on a large city, the SGD latent factors are the linear combination of them. The latent factors are also geolocation related, but not separated among the main cities like the SVD singular vectors.

In 8 we mapped the first three singular vectors of the Paris dataset. The different vectors may focus on different districts. However, they are not as clearly separated as the singular vectors of the France dataset.

In Section 5 we use these key observations to improve the recommendation quality of the SGD.

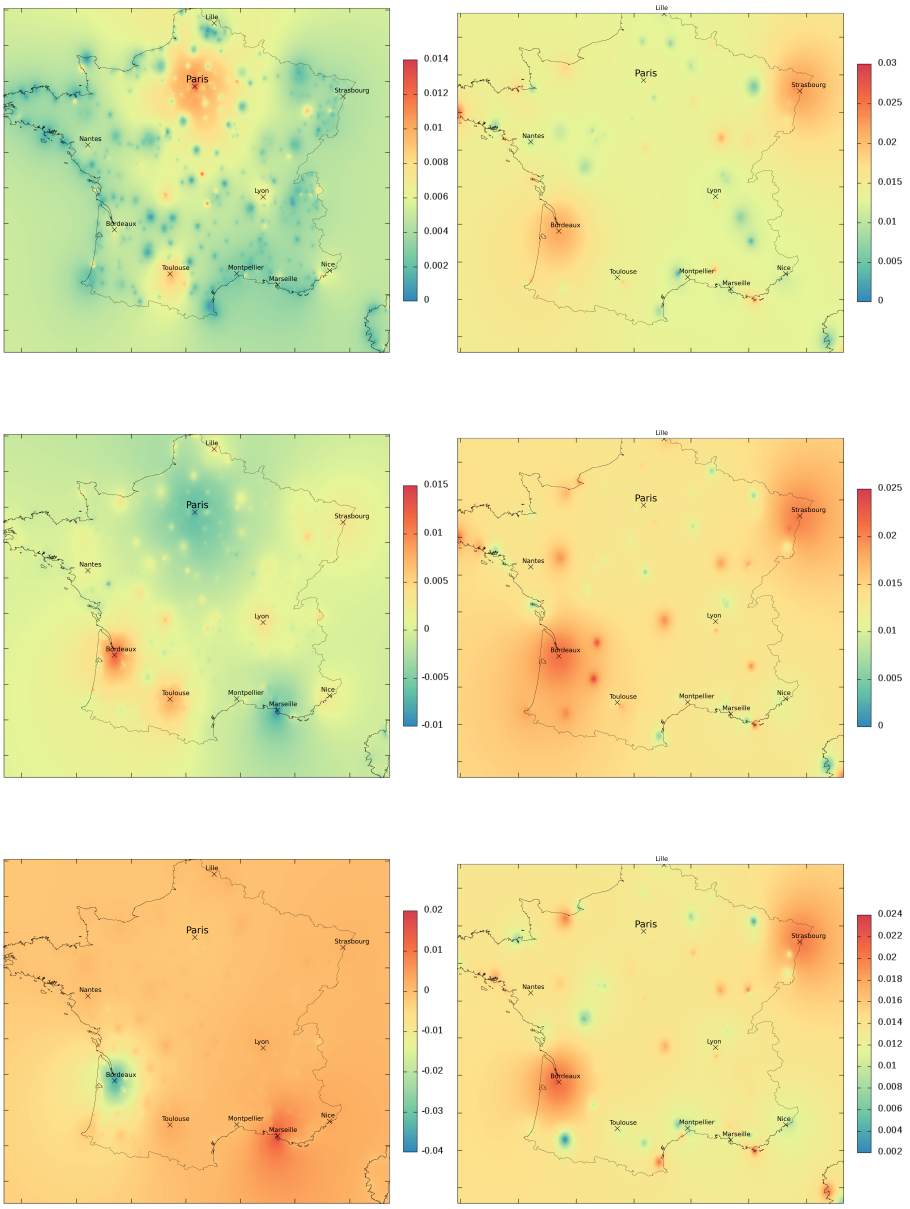


Fig. 7: The first, second and fourth singular vectors of the Nomao France rating matrix by SVD (**left**) and SGD (**right**). Here, the SVD eigenvectors correspond to Paris and Toulouse; Bordeaux, Toulouse and Marseille; Bordeaux and Marseille respectively, while the SGD plots for respective vectors are scattered around several cities.

5 Prediction for ratings and visits to locations

5.1 Recommender evaluation

Recommender systems serve to find *new* products for the users that are relevant for them. More specifically, for a given user u an item i a recommender system may retrieve the predicted relevancy $\hat{r}_{ui}$. This is called the rating prediction task. The Netflix Prize competition [4] was a challenge in rating prediction. While in the Netflix Prize competition, contestants were optimizing to predict all ratings to the users, a recommender system in practice selects the top rated items for a given user. In this top-K prediction task [9,8,33], a recommender system should retrieve for a given user a top list of items with length K. The top list should contain the most relevant items for the given user. This problem is more application related than the rating prediction task. In our experiments we examine both problems on the NOMAO datasets.

In addition to RMSE defined by equations (1) for full and (2) for partial matrices, we use two measures that evaluate the accuracy of the top-K recommendation task.

Recall at k is defined as the number of relevant POIs among the highest k values of row u in the matrix approximation,

$$\text{Recall}_u(k) = \frac{1}{|R_u|} \sum_{i=1}^{k} \text{rel}_{u,i}, \tag{10}$$

where $\text{rel}_{u,i}$ is the actual relevance of POI i for user u in the evaluation data, and R_u is the number of relevant items for user u in the dataset. We may average for all users to obtain

$$\text{Recall}(k) = \frac{1}{|U|} \sum_{u} \text{Recall}_u(k). \tag{11}$$

Normalized Discounted Cumulative Gain at k weights the relevance by the order of the predicted values as

$$\text{NDCG}_u(k) = \frac{\text{DCG}_u(k)}{\text{iDCG}_u(k)}, \tag{12}$$

where

$$\text{DCG}_u(k) = \sum_{i=2}^{k} \frac{\text{rel}_{u,i}}{\log_2(i+1)} \tag{13}$$

and

$$\text{NDCG}(k) = \frac{1}{|U|} \sum_{u} \text{NDCG}_u(k). \tag{14}$$

In our experiments, we randomly cut the data to training and test sets. We only use records in the training set to set the parameters of our model. The lower MSE, and the higher NDCG and recall we measure on the test set, the better is our model.

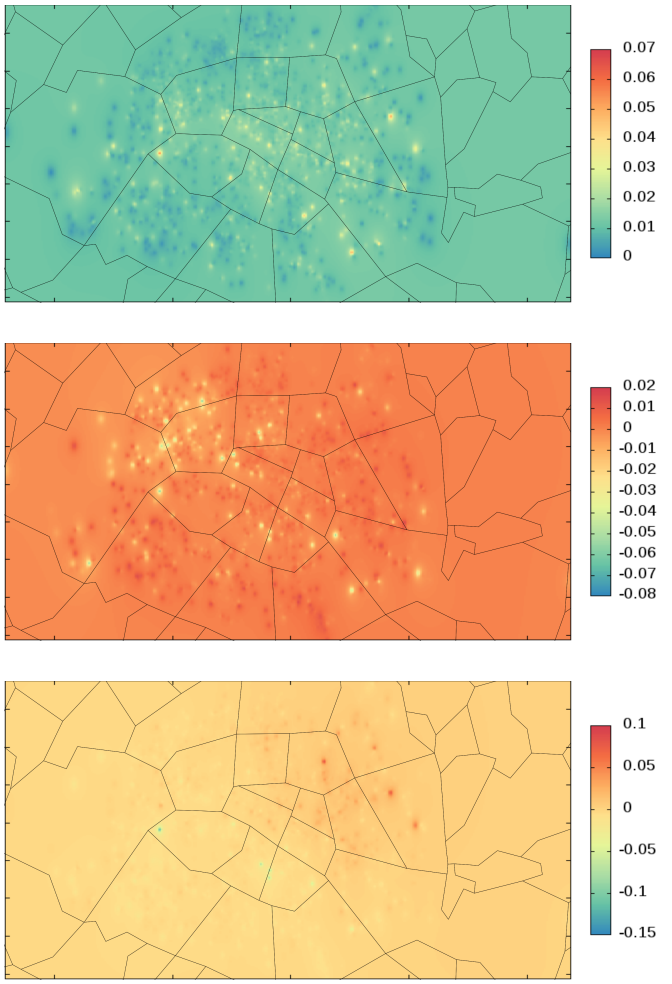


Fig. 8: The first, third and fourth singular vectors of the Nomao Paris rating matrix obtained by SVD.

5.2 The rating prediction task

As indicated in Table 1, bottom, and in Fig. 5, the scores have a peaked distribution. This indicates first that the rating prediction task makes less sense with these datasets. We trained up an SGD recommender by using 50% of the datasets and computed NDCG(k) for $K = 1 \ldots 20$. To understand the performance of the model, we also measured the performance of a random recommender that predicts ratings uniform randomly. We repeated our experiments 10 times with 10 different random training and test sets. Fig. 9 shows the computed ten performance curves for the SGD and the baseline random recommendation. Both for SGD and the random prediction the ten curves are similar. This indicates the stability of our algorithms and evaluation metrics. We achieved significantly better result with the SGD recommender. However for the random algorithm, the baseline NDCG is around 0.85. This is due to the fact that most of the ratings are around the mean as the score distribution is peaked.

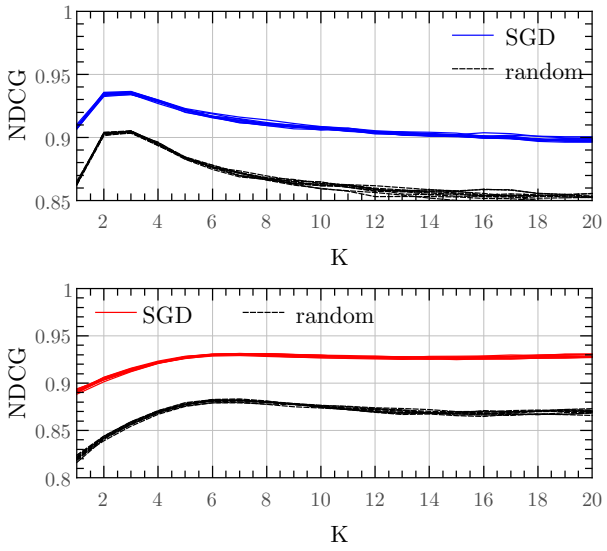


Fig. 9: Performance of score prediction on the France (**top**) and Paris (**bottom**) datasets.

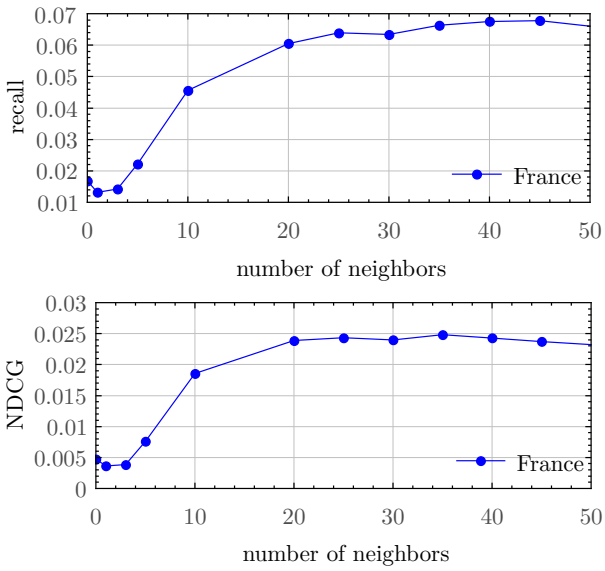


Fig. 10: Recall@100 and NDCG@100 for expansion with the original ratings.

5.3 Improving top recommendation with rating imputation

Instead of simply recommending locations near to already visited places, we expand the training set by relying on the locality of the ratings. We compare our results by using SVD or SGD both for the rating matrix and for simply predicting the visits, i.e. the existence of a rating regardless of its value. When considering locality, we may identify the nearest neighbors by taking the absolute distance and possibly correcting by density: in an area densely served by POIs, customers may reach more locations, on the other hand, the speed of travel is likely lower than in rural areas.

For our imputation methods, let E be the set of known ratings and N_j the neighbors of location j. We modify the training set as follows. For all (u, i),

$$\hat{r}_{u,i} = \begin{cases} r_{u,i} & \text{if } (u,i) \in E \\ f(R_u, N_{u,i}) & \text{if } (u,i) \notin E \text{ and for some } j,\ (u,j) \in E \text{ and } i \in N_j \\ \text{missing} & \text{otherwise,} \end{cases} \quad (15)$$

where f is function of R_u, the set of known ratings by user u, and $N_{u,i}$, the set locations visited by u in the neighborhood of i.

In our model, we expand the list of locations per user with the neighbors of visited places by the two strategies:

Constant:

$$f(R_u, N_{u,i}) = c \quad (16)$$

Ratings Average:

$$f(R_u, N_{u,i}) = \frac{1}{|N_{u,i}|} \sum_{j \in N_{u,i}} r_{u,j} \quad (17)$$

The performance for expansion with the original ratings (see (17)) on the France dataset is seen in Fig. 10 where we observe that expansion by the 30-40 nearest POIs improves significantly the matrix approximation by the first few eigenvectors.

We may also consider the task of predicting which POIs the user will visit, regardless of the actual rating given by the user. In this so-called implicit recommendation task, we consider a 0–1 matrix. Although the matrix is fully known, the meaning of a "1" is certain while a "0" may simply mean that the user has not yet had a chance to visit the POI or does not even know about it. Based on (16), the performance of the implicit task with expansion for the France dataset is seen in Fig. 11 showing an improvement compared to Fig. 10. However, for the Paris dataset, both in case of the ratings and implicit expansion experiments, we could not improve further the original SVD. This can be due to the fact that the Paris dataset is more dense geographically.

5.4 Improving recommendation with fixed factors

Results of Fig. 7 indicate that while SGD finds the most important cities in France, it can not separate them precisely. Furthermore, not recommending to a user POIs, that he/she have not visited, can be easily implemented without using SGD. Indeed, SGD should learn the taste of the different users like in case of the movie prediction task of Netflix. To fix this issue in the France dataset, we selected the top t cities in France. For a given item, we fixed the first ith factor to 1, if the item is located in the ith city, and 0 otherwise. We set the user factors similarly according to the places visited by the user in the test set. We then trained a k dimensional latent factor model where we updated only the remaining $k-t$ dimensions. We compared this recommender with a traditional k dimensional SGD recommender.

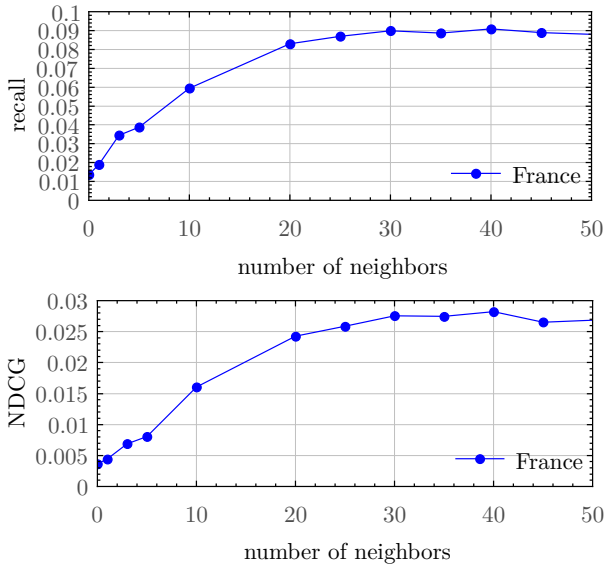


Fig. 11: Recall@100 and NDCG@100 for expansion nearby visited locations.

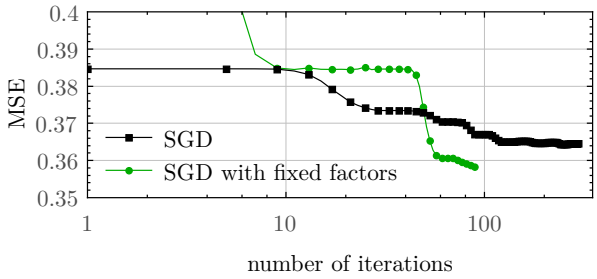


Fig. 12: Improved MSE results with fixed factors on the France dataset.

In our experiments we included the top $t = 10$ cities, in order Paris, Marseille, Lyon, Toulouse, Nice, Nantes, Strasbourg, Montpellier, Bordeaux, and Lille. Fig. 12 shows the MSE on the test set as the function of the number of iterations on the training set. With the fixed factor model we can achieve significantly better MSE. Furthermore, our best result is achieved with half less iterations compared to the number needed to train the original latent factor model.

6 Discussion

Our statistical analysis of NOMAO votes of customers for spots of France shows that it is described by a power law frequency distributions with exponents $a \approx 1.5$ (for spots) and $b \approx 2.75$ (for customers) which remain stable in time even is the number of votes is increased almost by two orders of magnitude during this time period. Further studies are required to establish the physical origins of such laws and to clarify for universal they are. It is possible that the physical reasons for emergence of such type distributions have certain similarities with the phenomenon of self-orgnanized criticality broadly discussed in physical systems (see e.g. [1,27,30]). It is interesting to note that the exponent of cluster distribution in self-critical models in 3D has an exponent close to 1.4 [1] being not so far from the exponet $a = 1.5$ we find for spots.

We explored the spectrum and the singular vectors of a POI ratings matrix of customer votes for spots of France. The fact that the matrix consists of 99.5% missing values makes the spectrum highly dependent on how we handle the missing values. We computed the SVD of the full 0–1 "implicit" matrix of the visits without considering the rating. For the ratings matrix, we used SGD, a popular approach that uses only the known values to compute the factors. We observed that SGD and SVD factors are similar but SVD has stronger geo-localization. SVD singular vectors with highest eigenvalues are mostly correlated with a particular place. As key practical observations, we found that imputing the missing ratings for the neighbors of visited places could increase the performance, and that defining fixed Geographic factors could improve SGD recommendation quality.

We expect that a broader analysis of a larger number of similar type datasets of votes will allow to gain better understanding of underlying physical process and provide better recommendations for specific customers and spots.

7 Acknowledgments

We thank the representatives of NOMAO [20] and especially Estelle Delpech (NOMAO) for providing us with the friendly access to the NOMAO datasets. This research is supported in part by the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE *No* 288956).

References

1. P. Bak, C. Tang and K. Wiesenfeld, *Self-organized criticality: an explanation of 1/f noise*, Phys. Rev. Lett. **59(4)**, 381 (1987)
2. J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel, *A survey on recommendations in location-based social networks.* ACM Transaction on Intelligent Systems and Technology (to be published) (2013)
3. R. M. Bell and Y. Koren, *Lessons from the Netflix prize challenge.* ACM SIGKDD Explorations Newsletter **9(2)**, 75 (2007)
4. J. Bennett and S. Lanning, *The Netflix prize*, KDD Cup and Workshop in conjunction with KDD 2007, (2007).
5. D. Billsus and M. J. Pazzani, *Learning collaborative information filters.* ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning, pages 46–54, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1998)
6. J. Canny, *Collaborative filtering with privacy via factor analysis*, SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, p.238, New York, NY, USA, ACM Press (2002)

7. C. Cheng, H. Yang, I. King, and M. R. Lyu, *Fused matrix factorization with geographical and social influence in location-based social networks*, AAAI **12**, 1 (2012).
8. P. Cremonesi, Y. Koren, and R. Turrin, *Performance of recommender algorithms on top-n recommendation tasks*, Proceedings of the fourth ACM conference on Recommender systems, ACM p.39 (2010).
9. M. Deshpande and G. Karypis, *Item-based top-n recommendation algorithms*, ACM Transactions on Information Systems (TOIS) **22(1)**, 143 (2004).
10. S. Dorogovtsev, *Lectures on complex networks*, Oxford University Press, Oxford (2010)
11. J. H. Friedman, *Greedy function approximation: A gradient boosting machine*, The Annals of Statistics **29(5)**, 1189 (2001)
12. S. Funk, *Netflix update: Try this at home*, http://sifter.org/˜ simon/journal/20061211.html, 2006.
13. K. R. Gabriel and S. Zamir, *Lower rank approximation of matrices by least squares with any choice of weights*, Technometrics **21**, 489 (1979)
14. G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, Baltimore (1983)
15. D. Gupta, M. Digiovanni, H. Narita, and K. Goldberg, *Jester 2.0 (poster abstract): evaluation of an new linear time collaborative filtering algorithm*, SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, p.291, ACM Press, New York, NY, USA (1999)
16. Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems*, Computer **42(8)**, 30 (2009)
17. T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, *Travel route recommendation using geotags in photo sharing sites*, Proceedings of the 19th ACM international conference on Information and knowledge management, ACM p.579 (2010)
18. M. Kurucz, A. A. Benczúr, K. Csalogány, and L. Lukács, *Spectral clustering in telephone call graphs* WebKDD/SNAKDD Workshop 2007 in conjunction with KDD 2007, (2007)
19. J. J. Levandoski, M. Sarwat, A. Eldawy, and M. F. Mokbel, *Lars: A location-aware recommender system*, Data Engineering (ICDE), 2012 IEEE 28th International Conference on, IEEE p.450 (2012)
20. NOMAO company, official web site http://fr.nomao.com/ . Web. 26 Apr. 2015
21. M. H. Pryor, *The effects of singular value decomposition on collaborative filtering*, Technical report, Dartmouth College, Hanover, NH, USA (1998)
22. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Application of dimensionality reduction in recommender systems–a case study*, ACM WebKDD Workshop (2000)
23. N. Srebro and T. Jaakkola, *Weighted low-rank approximations*, T. Fawcett and N. Mishra, editors, ICML, AAAI Press p.720 (2003)
24. P. Symeonidis, D. Ntempos, and Y. Manolopoulos, *Location-based social networks*, Recommender Systems for Location-based Social Networks, Springer, Berlin, p.35 (2014)
25. G. Takács, I. Pilászy, B. Németh, and D. Tikk, *Investigation of various matrix factorization methods for large recommender systems*. Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, ACM p.1 (2008)
26. G. Takács, I. Pilászy, B. Németh, and D. Tikk, *A unified approach of factor models and neighbor based methods for large recommender systems*, Applications of Digital Information and Web Technologies, 2008, ICADIWT 2008, First International Conference IEEE p.186 (2008)
27. Wikipedia contributors, *Abelian sandpile model*, Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 8 Apr. 2015, Web. 26 Apr. (2015)
28. Wikipedia contributors, *Netflix Prize*, Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 11 Feb. 2015, Web. 26 Apr. (2015)
29. Wikipedia contributors, *Recommender system*, Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 1 Apr. 2015, Web. 26 Apr. (2015)
30. Wikipedia contributors, *Self-organized criticality* , Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 13 Apr. 2015, Web. 28 Apr. (2015)
31. M. Ye, P. Yin, and W.-C. Lee, *Location recommendation for location-based social networks*, Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM p.458 (2010)
32. M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, *Exploiting geographical influence for collaborative point-of-interest recommendation*, Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, ACM p.325 (2011)
33. Q. Yuan, L. Chen, and S. Zhao, *Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation*, Proceedings of the fifth ACM conference on Recommender systems, ACM p.245 (2011)
34. S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman, *Using singular value decomposition approximation for collaborative filtering*, CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), Washington, DC, USA, IEEE Computer Society p.257 (2005)

Centrality Prediction in Temporally Evolving Networks*

Róbert Pálovics Ferenc Béres Nelly Litvak Frederick Ayala-Gómez András A. Benczúr

1 Abstract

In networks with very fast dynamics, such as Twitter mentions and retweets, predicting links and the emerging centrality of nodes is a challenging task. In contrast to existing methods that either consider static networks or a sequence of static snapshots, in this paper we give predictive models and centrality measures, both of which group can be dynamically updated after the addition of each new edge.

We propose a variant of matrix factorization for link prediction and compare the results with the online version of matrix factorization. To analyze the centrality measures, we propose online evaluation of Harmonic Centrality, PageRank, and Katz. To demonstrate our results, we use collections of topic specific Tweets.

2 Introduction

The research of complex networks and large graphs generated a wide variety of stochastic graph models that try to capture the properties of these complex systems [7, 11, 16, 25, 24]. Most of the well-known models can describe a static graph extracted from a real-world dataset. They are capable of generating an ensemble of graphs, in which all graph instances are similar in terms of specific statistics to the original one. For example, models that capture the power-law degree distribution of real-world networks such as the Albert-Barabasi one are dynamic but do not attempt to model the actual temporal evolution of large graphs. Our goal is to give temporal stochastic graph model for the temporal dynamics of these complex systems.

Our models address the link prediction problem introduced by Liben-Nowell and Kleinberg [29], in a *temporal* setting. More specifically, we try to predict accurately each new link in the graph at the time when it is created in the network. This experimental setting is similar to our method introduced for recommender systems [32]. In Section 6.1 we explain this setup in case of dynamic graphs. For baseline algorithm, we apply online matrix factorization [23, 34, 35] on temporal network data (see Section 4).

Various node centrality measures capture the "importance" of a node by using the structural properties of the graph [10]. While these metrics are widely investigated, few is known about the evolution of graph centrality in temporal graphs. In our work, we investigate the applicability of node centrality metrics in temporal graphs by examining their temporal behavior and computational complexity. We also use these metrics as side features in our matrix factorization models.

In our experiments we use the data set of [2] that consists of the messages and the corresponding user network of the Occupy movement.

As our main result, we demonstrate that methods of matrix factorization by online learning are capable of improving predictions for the future centrality of nodes. Surprisingly, we find no direct relation between the quality of predicting the links and the derived quality of predicting centrality. As a byproduct, we define time aware variants of certain centrality measures, however our main goal is the prediction and not the definition of centrality metrics as those in e.g. [28, 17]. For centrality metrics, we use those in [29, 10] with appropriate time aware modifications.

*The publication was supported in part by the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956).

2.1 Related results

Social influence in Web based networks is investigated in several results: Bakshy et al. [5] model social contagion in the Second Life virtual world. Ghosh and Lerman [18] compares network measures for predicting the number of votes for Digg posts, who even give an empirical comparison of information contagion on Digg vs. Twitter [27]. In [19, 20], long discussion based cascades built from comments are investigated in four social networks, Slashdot (technology news), Barrapunto (Spanish Slashdot), Meneame (Spanish Digg) and Wikipedia. They propose models for cascade growth and estimate model parameters but give no size predictions.

A number of related studies have largely descriptive focus, unlike our quantitative prediction goals. In [12] high correlation is observed between indegree, retweet and mention influence, while outdegree (the number of tweets sent by the user) is found to be heavily spammed. [26] reports similar findings on the relation among follower, mention and retweet influence. Several more results describe the specific means of information spread on Facebook [6, 3, 8]. In the first paper on the data set that we also use for our experiments [2], the authors investigate how emotions appear in Twitter.

Myers and Leskovec [30] showed that the Twitter network is highly dynamic with about 9% of all connections changing in a month. Thus, in order to infer central nodes, the factors driving the dynamics of this social network must be considered. They focused on local bursts in the user-follower network to identify key events or bursts in the information flow. They consider both follow and unfollow bursts.

Chierichetti et al. [14] propose a robust model for the real-time identification of key events. They examined tweet and retweet production/consumption patterns around these incidents. The experiments showed that there is a heartbeat phenomenon in the balance of primary and secondary information spreading. When and important event unfolds, the users are busy with tweeting about it, as they try to report everything. Thus, nobody has time to retweet these messages. Whereas after the event, there is a huge amount of tweets to be retweeted. So in this case, the secondary information spreading dominates the network. The authors used this phenomenon to obtain a simple classifier which, by only evaluating the tweet/retweet volume could detect these events.

The results of Bakshy et al. [4] attempt to predict the influential users of a Twitter user-follower graph by generating diffusion cascades. At first, they extract influential vertices with regression merely based on network features. Their main problem is the minority of cascades with significant size. Although they improved their results with content information about the cascades, the problem remained open.

Cheng et al. [13, 21] predict retweet count based on network features. Petrovic et al. [33] introduce time sensitive modeling by using the PA algorithm of [15], which is an online solution to the linear regression problem. They only predict if a tweet will be retweeted at all.

Rodriguez et al. [36] gave an algorithm for inferring the structure of temporal diffusion networks. They examined an interesting aspect of centrality for many real-time events.

The direct starting point of our work is the first comprehensive overview of methods for time agnostic link prediction is given in [29]. Most of the methods used in [29] are listed in Section 5.

As one of the first time aware link prediction methods, Tylenda et al. [38] propose a maximum entropy model with weights inversely proportional to the age of the edges, however their method is trained on a single, though timestamped, snapshot and evaluated on the future in a batch. Similar to our evaluation methodology described in Section 6.1, they use DCG, however they do not consider a time aware DCG evaluation as first proposed in our work [32].

Closest to our work, [28, 17] defines a new time aware centrality measure, which they evaluate only on yearly snapshots of scientific citation networks. Our main contribution is the use of online learning methods for fine granularity evaluation of centrality measures similar to those in time aware centrality research results.

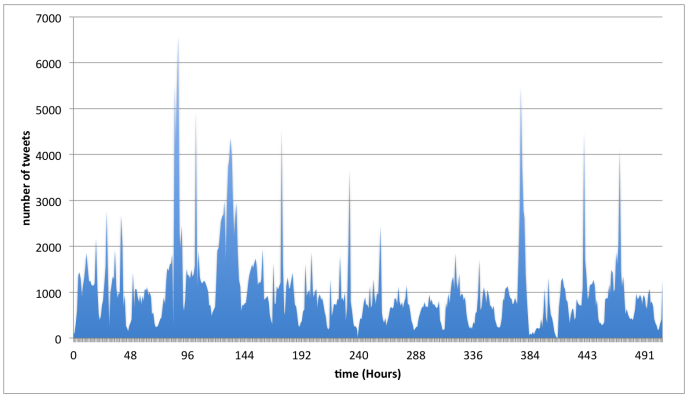


Figure 1: Temporal density of tweeting activity.

Table 1: Size of the tweet time series.

Number of users	371,401
Number of tweets	1,947,234
Number of retweets	1,272,443

3 Datasets

The dataset was collected by Aragón et al. [2] using the Twitter API that we extended by a crawl of the user network. Our data set hence consists of two parts:

- *Tweet dataset:* tweet text and user metadata on the Occupy Wall Street movement[1].
- *Follower network:* The list of followers of users who posted at least one message in the tweet dataset.

Table 1 shows the number of users and tweets in the dataset. One can see that a large part of the collected tweets are retweets. Table 2 contains the size of the crawled social networks. Note that the average in- and outdegree is relatively high. Fig. 1 shows the temporal density of tweeting activity.

For each tweet, our data contains

- tweet and user ID,
- timestamp of creation,
- hashtags used in the tweet, and
- the tweet text content.

In case of a retweet, we have all these information not only on the actual tweet, but also on the original *root tweet* that had been retweeted. We define the root tweet as the first occurrence of a given tweet.

[1] http://en.wikipedia.org/wiki/Occupy_Wall_Street

Table 2: Size of the follower network.

Number of users	330,677
Number of edges	16,585,837
Average in/out degree	37

4 Dynamic adjacency matrix factorization

Batch modeling algorithms may iterate several times over the graph until convergence. In our temporal setting, 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 factorization method for the graph adjacency matrix. Matrix factorization yields a low-rank approximation of the adjacency matrix with entries for non-edges filled with values that we consider an indication for the edge to appear. Links for a given node are predicted by taking the largest values in the corresponding row or column. 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 each record in the dataset as a positive training instance and generate negative training instances by selecting random items for each positive record. We use the regularized matrix factorization method of [37], and use the k-factor model for prediction.

Temporal modeling methods 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 that we empirically verify in our research.

5 Centrality measures

5.1 Negative β-measure

Let $d^+(v)$ denote the outdegree of vertex v. Negative β-measure is defined by

$$\sum_{y \to x} \frac{1}{d^+(y)}.$$

and it can be considered as *Markovian indegree*.

5.2 Closeness

The closeness of node x is defined by

$$\frac{1}{\sum_y d(y,x)},$$

where $d(y,x)$ denotes the distance of x from y in the directed network. While indegree and negative β-measure were relying on the local graph structure, closeness is defined by the global graph structure. Thus, it is more costly to compute.

It is important to remark that the graph must be strongly connected. Without this condition the result will be a null score for all x node, that cannot coreach the whole graph. Nevertheless there have been many propositions on how to mend this troublesome quality of closeness. But the most straightforward idea is to exclude infinite distances

$$\frac{1}{\sum_{d(y,x)<\infty} d(y,x)}.$$

5.3 Lin's index

One of the ideas that tried to repair the definition of closeness for graphs with infinite distances was Nan Lin's. Lin's index defines the score of node x with a nonempty coreachable set as

$$\frac{|\{y|d(y,x) < \infty\}|^2}{\sum_{d(y,x)<\infty} d(y,x)}.$$

Nodes with an empty coreachable set have centrality 1 by definition.

This change in the definition means that closeness is not the inverse of a sum of distances, but rather the inverse of the average distance. One of the results of this modification is that closeness is normalized across the graph.

5.4 Harmonic centrality

Paolo Boldi and Sebastiano Vigna in [10] gave another solution on how to eliminate the problem of non-finite distances between nodes. The main idea is to use harmonic mean instead of arithmetic averaging. The reason why harmonic mean is involved is that it conveniently deal with ∞ distances, as $\frac{1}{\infty} = 0$. The definition for the harmonic centrality of node x is

$$\sum_{x \neq y} \frac{1}{d(y,x)} = \sum_{d(y,x)<\infty, x \neq y} \frac{1}{d(y,x)}, \tag{1}$$

which is the reciprocal of the denormalized harmonic mean of distances. In [10] the authors found that harmonic centrality is strongly correlated to closeness in simple networks. Moreover, this definition also accounts for nodes y that cannot reach x. Thus, this measure can also be used in cases when the given graph is not strongly connected.

5.5 Katz index

Katz defined his index through summation of all paths coming into a node x. In order to get a finite score, he introduced an *attenuation factor* β with which a weight could be calculated for the paths. The Katz index can be expressed as

$$k = \mathbf{1} \cdot \sum_{i=0}^{\infty} \beta^i A^i, \tag{2}$$

which is equivalent to

$$k = \mathbf{1} \cdot (1 - \beta A)^{-1}, \tag{3}$$

where $\mathbf{1}$ is the vector with uniformly 1 coordinates. Furthermore, by Brauer's theorem on the displacement of eigenvalues, the Katz index is the left dominant eigenvector of a perturbed matrix

$$\beta\lambda \cdot A + (1 - \beta\lambda) \cdot e^T \cdot \mathbf{1},$$

where e is a right dominant eigenvector of A such that $\mathbf{1}e^T = \lambda$. Hubbell [22] proposed a generalization for the Katz index, in which some preference vector v is used instead of $\mathbf{1}$. In other words, the paths can be weighted individually depending on their starting node. The normalized limit of the Katz index when $\beta \to \frac{1}{\lambda}$ is the dominant eigenvector.

5.6 PageRank

Recently, PageRank is one of the most frequently discussed and cited spectral measure in use, mainly because of its alleged use in the Google ranking algorithm. PageRank [31] is defined by the unique vector p satisfying equation

$$p = \alpha \cdot p\bar{A} + (1 - \alpha)v, \tag{4}$$

where $\bar{A}$ is derived from the adjacency matrix A with the same l_1-normalization, that was used in the formulation of Seeley's index and the negative β-measure. PageRank has two additional parameter. A *damping factor* $\alpha \in [0, 1)$, and a *preference vector* v. The only constraint for v is that it must be a distribution.

However, it is important to note that p is not necessarily a probability distribution if A has null rows. There has been several propositions on how to make $\bar{A}$ stochastic. A common solution is to replace every

null row with the preference vector v. Another popular idea is to add loop arcs to all nodes with zero outdegree (dangling nodes).

Equation 4 is solvable even without any patching, as after reorganizing the formula we get

$$p = (1-\alpha)v(1-\alpha\bar{A})^{-1}. \tag{5}$$

Moreover, another equation can be formulated for PageRank

$$p = (1-\alpha)v\sum_{i=0}^{\infty}\alpha^i\bar{A}^i, \tag{6}$$

which shows that the Katz index and PageRank differ only by a constant factor and by the l_1 normalization applied to the adjacency matrix. If A has no null rows, or $\bar{A}$ has been patched to be stochastic, PageRank can be equivalently defined as the stationary distribution of the Markov chain whose transition matrix is

$$\alpha\bar{A} + (1-\alpha)\mathbf{1}^T v.$$

5.7 Betweeness

Let σ_{yz} denote the number of shortest paths going from y to z. A subset of these paths also passes through node x, and suppose their number is $\sigma_{yz}(x)$. The betweenness measure of node x is defined by

$$\sum_{y,z\neq x,\sigma_{yz}\neq 0}\frac{\sigma_{yz}(x)}{\sigma_{yz}}$$

The definition tries to capture the intuition that if a significantly large fraction of shortest paths passes through x, then x is an important junction point of the graph. Moreover, Boldi et al. in [9] showed that removing nodes with high betweenness score results in an instant network disruption.

6 Experiments

6.1 Experimental setting and evaluation metrics

In the dynamic link prediction task, we have to rank the best K links for the given node at the given time instance. Our dataset contains records $< u, v, t >$ of links between users u and v that appear at time t. Our goal is to recommend new links for user u at time t with the constraint that there is only a single link that appears at the given time t. This means that we have to maximize the rank of the given link in the actual predicted list of links. A time sensitive or online link prediction system should retrain its model after each and every training record $< u, v, t >$. We have to generate new top-K recommendation list for *every* single record. The online top-K task is hence different from the standard recommender evaluation settings, since there is always a single neighbor only in the ground truth and the goal is to aggregate the rank of these single neighbors over the entire testing period. For our task, we need carefully selected quality metrics that we describe next. We use our full dataset both for training and testing. We iterate on the records one by one in temporal order. For a given record $< u, v, t >$, we allow the recommender algorithm to use full of the data *before* t in question for training and require a ranked top list of possible neighbors as output. We evaluate the given single actual neighbor v in question against the recommended top list of length K.

For measuring the accuracy of predicting a new link, we face the difficulty that only a single correct answer exists at the given time and the next edge arrives to be tested against an updated model. We propose DCG [38, 32], a modified version of NDCG, the preferred model for batch top-K recommendation [1]. DCG is a slowly decreasing function of the rank and hence measures how close the actual new link appears in the top list.

To sum up, in our experiments we use this experimental setting and evaluation. We iterate over the edge list of a given graph in temporal order. One record in our dataset is a timestamped edge between two users in the graph, $< u, v, t >$. Instead of items, we recommend for users new neighbors. For each $< u, v, t >$, we evaluate our top-K recommendation by using DCG as evaluation metric. Finally, we compute temporal averages of the DCG scores.

6.2 Accuracy of link prediction

In Fig. 2, we show daily average link prediction quality by using online matrix factorization defined in Section 4. We give results for different learning rates and conclude that there is a large variance but in general, very low learning rates around 0.05 perform the best.

6.3 Accuracy of centrality prediction

In Fig. 3, we show daily average centrality prediction quality by computing various centrality measures over the graph augmented by the edges predicted by online matrix factorization as in Section 4. We give results for different learning rates. Unlike for link prediction, we observe stable performance across different metrics improving up to a learning rate of 0.08 and declining beyond.

Also note that for in-degree, Beta and PageRank we are able to improve over the prediction given by the previous state of the graph as baseline. We plan to evaluate different weighted combinations of centrality values on past and predicted future graphs.

7 Conclusion and Further Work

In this paper, we analyze the dynamic network data as a stream of nodes and edges. To predict link formation, the regularized matrix factorization model is proposed. Different centrality measures are used with online computation over the graph stream to identify the evolution of centrality. As the main lesson learned, we show how recent results in recommender systems can be deployed for the analysis of complex networks.

Matrix factorization algorithm may use so-called side information associated with the rows and columns of the matrix. We plan to use centrality measures as side information associated with the nodes. We may use directed centrality with different values for rows and columns of the same node. We plan to compare the following metrics in the temporal setting of dynamic networks based on [10]: Harmonic Centrality, PageRank, HITS and SALSA.

In addition, we would like to test our methods on a variety of other data sets from Twitter, Last.fm, scientific citation networks and more.

References

[1] A. Al-Maskari, M. Sanderson, and P. Clough. 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*, pages 773–774. ACM, 2007.

[2] P. Aragón, K. E. Kappler, A. Kaltenbrunner, D. Laniado, and Y. Volkovich. Communication dynamics in twitter during political campaigns: The case of the 2011 spanish national election. *Policy & Internet*, 5(2):183–206, 2013.

[3] E. Bakshy, D. Eckles, R. Yan, and I. Rosenn. Social influence in social advertising: evidence from field experiments. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 146–161. ACM, 2012.

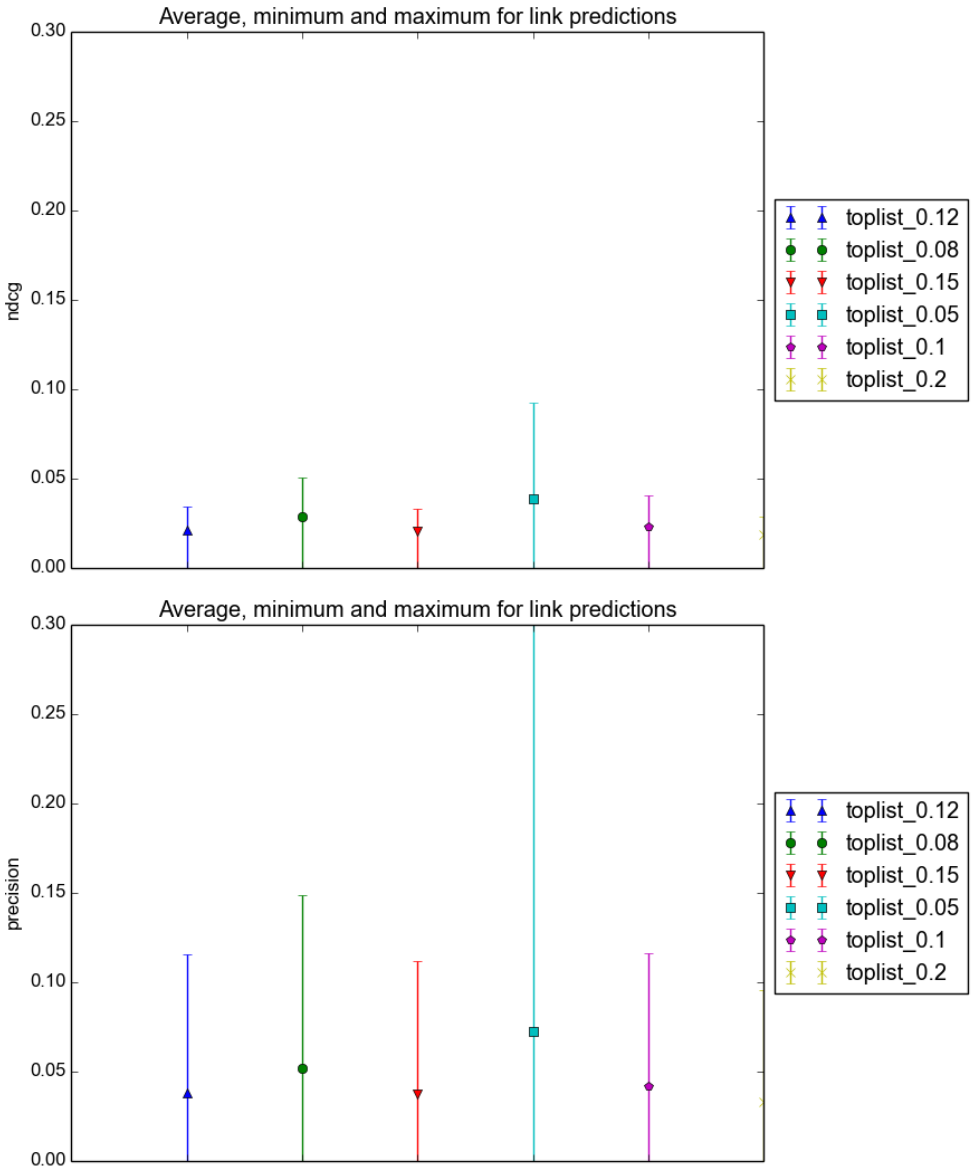


Figure 2: Quality of link prediction, NDCG (**top**) and precision (**bottom**).

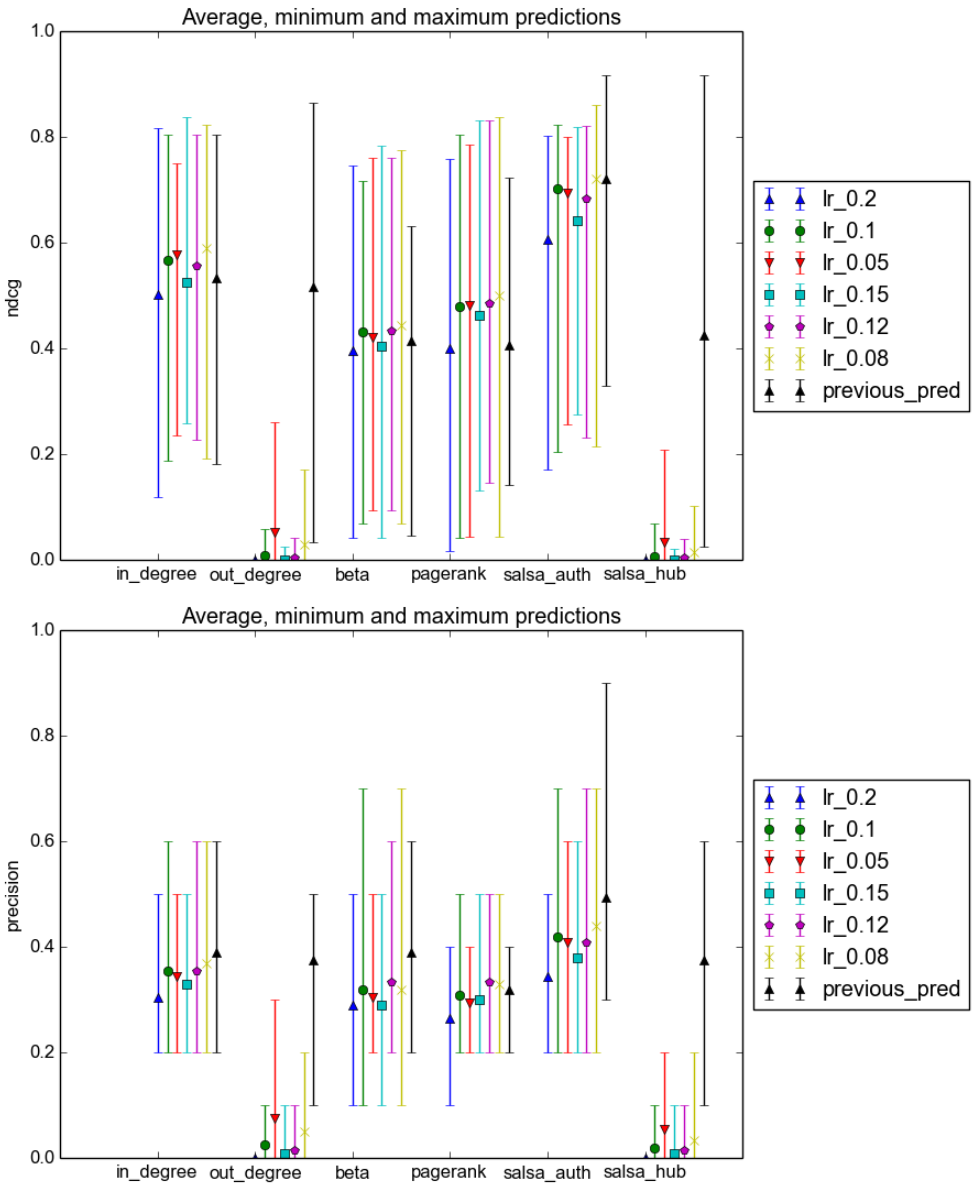


Figure 3: Quality of centrality prediction, NDCG (**top**) and precision (**bottom**).

[4] E. Bakshy, J. M. H., W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 65–74. ACM, 2011.

[5] E. Bakshy, B. Karrer, and L. A. Adamic. Social influence and the diffusion of user-created content. In *Proceedings of the 10th ACM conference on Electronic commerce*, pages 325–334. ACM, 2009.

[6] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.

[7] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] M. S. Bernstein, E. Bakshy, M. Burke, and B. Karrer. Quantifying the invisible audience in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 21–30. ACM, 2013.

[9] P. Boldi, M. Rosa, and S. Vigna. Robustness of social networks: comparative results based on distance distributions. In *Proceedings of the Third international conference on Social informatics*, pages 8–21. Springer-Verlag, 2011.

[10] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[11] B. Bollobás, O. Riordan, J. Spencer, and G. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001.

[12] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2010.

[13] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936. International World Wide Web Conferences Steering Committee, 2014.

[14] F. Chierichetti, J. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. Event detection via communication pattern analysis. In *Eighth International AAAI Conference on Weblogs and Social Media*, 2014.

[15] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[16] V. Csiszár, P. Hussami, J. Komlós, T. F. Móri, L. Rejtő, and G. Tusnády. When the degree sequence is a sufficient statistic. *Acta Mathematica Hungarica*, 134(1-2):45–53, 2012.

[17] R. Ghosh, T.-T. Kuo, C.-N. Hsu, S.-D. Lin, and K. Lerman. Time-aware ranking in dynamic citation networks. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 373–380. IEEE, 2011.

[18] R. Ghosh and K. Lerman. Predicting influential users in online social networks. *arXiv preprint arXiv:1005.4882*, 2010.

[19] V. Gómez, H. J. Kappen, and A. Kaltenbrunner. Modeling the structure and evolution of discussion cascades. In *Proceedings of the 22nd ACM conference on Hypertext and hypermedia*, pages 181–190. ACM, 2011.

[20] V. Gómez, H. J. Kappen, N. Litvak, and A. Kaltenbrunner. A likelihood-based framework for the analysis of discussion threads. *World Wide Web*, pages 1–31, 2012.

[21] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 57–58, New York, NY, USA, 2011. ACM.

[22] C. H. Hubbell. An input-output approach to clique identification. *Sociometry*, pages 377–399, 1965.

[23] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[24] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–10, 2000.

[25] M. Kurucz and A. Benczúr. Geographically organized small communities and the hardness of clustering social networks. *Data Mining for Social Network Data*, pages 177–199, 2010.

[26] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010.

[27] K. Lerman and R. Ghosh. Information contagion: An empirical study of the spread of news on digg and twitter social networks. In *Proceedings of 4th International Conference on Weblogs and Social Media (ICWSM)*, 2010.

[28] K. Lerman, R. Ghosh, and J. H. Kang. Centrality metric for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 70–77. ACM, 2010.

[29] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the 12th Conference on Information and Knowledge Management (CIKM)*, pages 556–559, 2003.

[30] S. A. Myers and J. Leskovec. The bursty dynamics of the twitter information network. In *Proceedings of the 23rd international conference on World wide web*, pages 913–924. International World Wide Web Conferences Steering Committee, 2014.

[31] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford University, 1998.

[32] R. Pálovics, A. A. Benczúr, L. Kocsis, T. Kiss, and E. Frigó. Exploiting temporal influence in online recommendation. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 273–280. ACM, 2014.

[33] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In *ICWSM*, 2011.

[34] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[35] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[36] M. G. RODRIGUEZ, J. LESKOVEC, D. BALDUZZI, and B. SCHÖLKOPF. Uncovering the structure and temporal dynamics of information propagation. *Network Science*, 2(01):26–65, 2014.

[37] G. Takács, I. Pilászy, B. Németh, and D. Tikk. 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*, pages 1–8. ACM, 2008.

[38] T. Tylenda, R. Angelova, and S. Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis*, page 9. ACM, 2009.

Liquid FM: Recommending Music through Viscous Democracy*

Paolo Boldi, Corrado Monti, Massimo Santini, and Sebastiano Vigna

Dipartimento di Informatica, Università degli Studi di Milano, Italy
corrado.monti@unimi.it

Abstract. Most modern recommendation systems use the approach of *collaborative filtering*: users that are believed to behave alike are used to produce recommendations. In this work we describe an application (Liquid FM) taking a completely different approach. Liquid FM is a music recommendation system that makes the user responsible for the recommended items. Suggestions are the result of a voting scheme, employing the idea of *viscous democracy* [3]. Liquid FM can also be thought of as the first testbed for this voting system. In this paper we outline the design and architecture of the application, both from the theoretical and from the implementation viewpoints.

1 Introduction

Most modern recommendation systems use the approach of *collaborative filtering* [8,2]: users that are believed to behave alike are used to produce recommendations. The idea behind Liquid FM is to tip over this approach by making the user responsible for these matches: deciding who they want to resemble becomes a choice of the user, instead of being inferred algorithmically. This scenario can be cast as a voting scheme: each user has to select another one that is believed to be a good recommender. This idea allows us to use this task as a testbed for *viscous democracy* [3].

Viscous democracy is a kind of liquid democracy [6]. In liquid (or *delegative*) democracy, each member can take an active role—by participating directly and exercising their decision power—or a passive role—by *delegating* to other members their share of responsibility. It can be seen as a compromise between representative democracy, where voters are usually neglected any decision making and can only delegate others to do so, and direct democracy, where every voter is called to an active role, regardless of what their inclinations are.

In this sense, liquid voting systems try to take the best from both worlds. Every member's opinion, in a direct democracy, is directly relevant to a final decision, but the vote of each one can be (knowingly!) uninformed; instead, in a classical representative system, elected representatives are encouraged to be informed on the specific decision they are making, but on the other hand the majority of people feel that their opinion on that matter is basically irrelevant.

* The authors were supported by the EU-FET grant NADINE (GA 288956)

Liquid democracy permits members to choose among expressing their opinion directly if they feel entitled to do so, or delegating their voting power if they believe others are more capable. Note that these two options are not necessarily exclusive—in our case, in fact, users will be able to do both, if they want to.

Viscous democracy was proposed by Boldi *et al.* in [3] as a particular way to compute the outcome of a liquid democracy voting scheme. It takes advantage of known techniques for measuring centrality in social networks, and in particular it resembles Katz's index [5]. It stems from the assumption that the delegating mechanism should transfer *a fraction* of the user voting power. I.e., if A delegates B and B delegates C, the trust that A puts in C should be less than if A voted C directly. This principle will be further detailed in the next section.

This framework can be used in a variety of settings. In our application, we show how it can be easily adapted to music recommendation. For a certain music genre, we ask users to express a short list of their favorite songs, or to delegate one of their Facebook friends they consider to be an expert on that genre. This builds a graph of delegations for each music genre. We wish to employ this data to create recommendations for each user.

We will detail how we extract information from this graph in Section 2; then, in Section 3, we will describe how we have developed the system: how its algorithms were implemented, the architecture of its components, and the external resources we used; finally, in Section 4 we will sum up our work and present possible directions for future research.

2 Viscous democracy and recommender systems

From now on, we will denote with $d_G(x, y)$ the distance from node x to node y in the graph G, and with $o_G(x)$ the outdegree of node x in G; we may omit reference to G if it is obvious from the context.

Let us define U as the set of users and S as the set of songs[1]. $D = (U, A_D)$, with $A_D \subseteq U \times U$, is the directed graph of delegations; an arc from user u to u' means the former delegates the latter as an expert on the topic. $V = (U, S, A_V)$, with $A_V \subseteq U \times S$, is the bipartite graph of votes, where an arc from $u \in U$ to $s \in S$ means that the user u recommends song s.

We are going to put some restrictions on these graphs: first of all, we are going to assume that there is an underlying, undirected friendship graph $F = (U, E_F)$, with $E_F \subseteq U \times U$, where an edge $(u, u') \in E_F$ expresses a personal acquaintance of u and u'. We impose that $A_D \subseteq E_F$: this permits us to ensure that the trust expressed through a delegation is a result of personal knowledge, as suggested in [3].

Further, we are going to impose that $\forall u \in U$, we have $0 \leq o_D(u) \leq 1$, meaning that a user can delegate only one person, and $0 \leq o_V(u) \leq 3$, meaning that every user can vote up to 3 songs. We are going to consider only nodes

[1] As we will explain in Section 3, we are going to consider different sets of songs and votes, one for each music genre treated. For the rest of this section, we are going to consider the music genre as fixed.

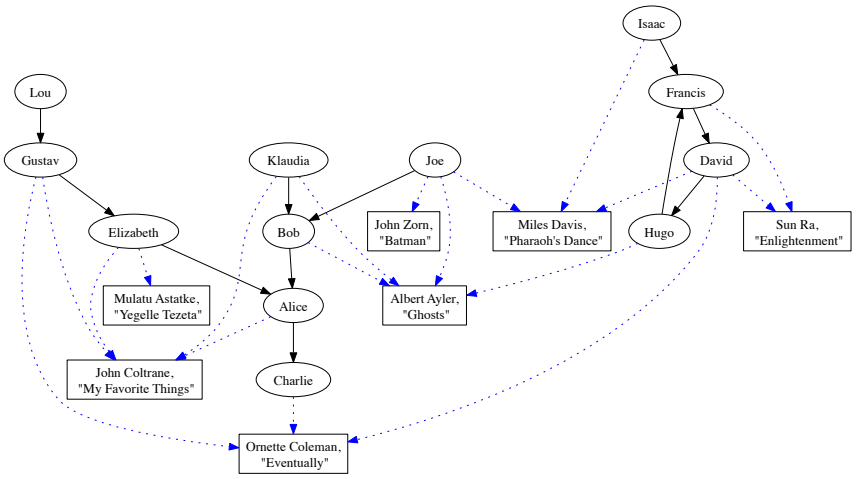


Fig. 1. An example of delegation and voting graphs. Users $u \in U$ are represented with a circle; songs $s \in S$ with a box; the delegation graph D is drawn with black solid arrows, while the bipartite voting graph V with blue dotted arrows.

$u \in U$ having $o_D(u) > 0 \vee o_V(u) > 0$. An example of such a setting is pictured in Figure 1.

2.1 Liquid voting

A voting system is a function $v_D : U \to \mathbb{R}$ assigning a score to each user, depending on the delegation graph. Such a function will be the basic building block of our recommendations.

Usually, in liquid vote this function is just the size of the tree with root in $u \in U$:

$$l_D(u) = \left| \{u' \in U | d_D(u', u) < \infty\} \right|$$

This function is used, e.g., by the well-known `LiquidFeedback`[2] platform. Nonetheless, it assumes that "trust" transferred from a to b is the same whether a delegated b directly, or whether they are connected by a long chain of delegations—and they may not even know each other.

Let us assume that we wish, instead, that the amount of trust passed on from a to b is greater if $(a, b) \in A_D$, and lesser if there are many steps connecting them. To do so, we introduce a *damping factor* $\alpha \in (0, 1]$, defining how much of the voting power of a is transferred to b when a delegates b. Therefore, the

[2] `http://liquidfeedback.org/`

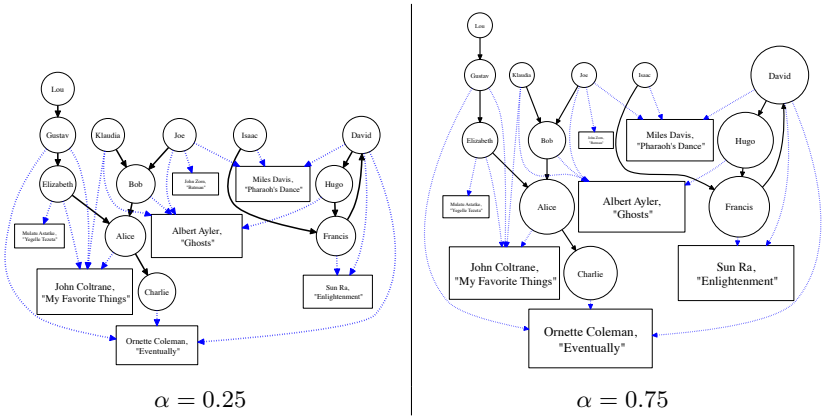


Fig. 2. The same graphs pictured in Figure 1 are here displayed with node size proportional to their viscous score, with two different values for α. Note how a higher α gives higher importance to users delegated by important users. Lowering its value get us closer to a simple vote count. For example, Ornette Coleman's song is ranked higher than Coltrane's only for higher α: this is because it is voted by fewer users, but those are recognized by the community as experts.

scoring function characterizing *viscous* democracy will be:

$$v_D(u) = \sum_{u' \in U} \alpha^{d(u',u)} \tag{1}$$

Authors [3] have noted how, depending on the value of α, the behavior of the voting function greatly differs. For higher values of α, the fraction of trust "lost" in each delegation step becomes smaller and smaller; in fact, for $\alpha \to 1$, we have that $v_D \to l_D$: all the nodes in the tree of u contribute with all their voting power to u, exactly as in pure liquid democracy. Note that if we allow $\alpha = 1$, we must explicitly avoid cycles in D—exactly as with pure liquid democracy; this constraint is not needed with viscous democracy with $\alpha \in (0, 1)$.

With α approaching 0, instead, the voting power becomes nontransferable: all users become equal, regardless of the delegations they received; in other words, the model becomes a direct democracy, without any proxy vote. These differences are presented graphically in Figure 2, making use of the song-scoring function we will show in the next section.

2.2 Global recommendations

Having a score for each user, we can easily score each song $s \in S$. Indeed, we can define a function $r : S \to \mathbb{R}$ as

$$r(s) = \sum_{u \in U | (u,s) \in A_V} v_D(u) \tag{2}$$

This function will get us a score for a song proportional to the importance of who voted it, according to v_D. The score is completely defined by the graphs V and D. We can then proceed to rank each song with r, and present them to the users accordingly. As in many standard information retrieval tasks, a user looking for results (about a certain music genre, as we will explain in Section 3) will be presented with all possible items—all songs in S—ranked from higher to lower r. Users will be therefore more likely to listen to songs ranked higher in this list.

Let us call the *influence* of $u \in U$ the difference the votes of user u make in the final rankings—that is, $\sum_{s \in S} r(s) - r_{V \setminus \{u\}}(s)$. Please note that, since we have not normalized r, users giving more votes have a larger influence in the final rankings, serving the purpose of encouraging them to give more recommendations. However, it also explains why we had to put a limit on $o_V(u)$: if we had not, a single user u could have an arbitrary influence on the score r, resulting in the possibility of spam.

In the end, the influence of a user on song scores is determined by the number of recommendations they give—limited, but under their control—and by the delegations they received—unlimited, but not under their direct control.

As mentioned before, an example of how r behaves is pictured in Figure 2.

2.3 Personalized recommendations

The song-scoring function we presented gives the same ranks to whoever is their observer. This behavior is unusual in recommender systems, where the goal is to give the right recommendation to the right person. In our case, a user may be more interested in listening to what their delegate suggested, rather than other—possibly more popular—items. Looking at our example in Figure 2, Francis may be more interested in listening to "*Pharoah's Dance*", even if it is not globally highly-ranked, because it is the recommendation of his delegate David. Similarly, Hugo may be interested in it, because he has, in turn, delegated Francis.

This goal can be easily expressed as a personalized song-scoring function. Let us define a function $p : S, U \to \mathbb{R}$ as

$$p(s,u) = \sum_{u' \in U | (u',s) \in A_V} \alpha^{d(u,u')} \tag{3}$$

Such a function permits the user u to get a positive score only for the songs recommended by users belonging to the chain of delegations starting in u. For the purpose of maintaining this intention, but at the same time avoiding to completely discard all the songs highly ranked by the original r, we can define a linear combination of the two functions, normalized to 1:

$$c(s,u) = \delta \frac{p(s,u)}{\max_{s' \in S} p(s',u)} + (1-\delta) \frac{r(s)}{\max_{s' \in S} r(s')} \tag{4}$$

where $\delta \in [0,1]$ regulates the amount of personalization of c.

An example is pictured in Figure 3.

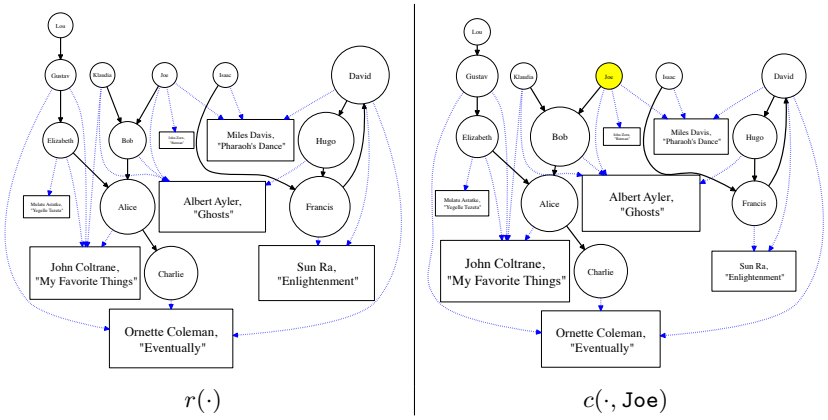


Fig. 3. The same graphs pictured in Figure 2 are here displayed with global song-scoring function r on the left and, on the right, with the personalized function c from the view point of user Joe (in yellow) and $\delta = 0.9$. In the latter, recommendations suggested by the delegate of Joe acquire more importance; those suggested by indirect delegates (namely, Alice and Charlie) increase as well, but by a minor amount.

2.4 Insights for users

In addition to the presented ways to compute recommendations, the setting here described also permits to compute other information that may be of interest to the users. Particularly, it allows them to know how authoritative (i.e., trustable) their taste is in a particular music genre. The function v_D, in fact, can be normalized into a percentile-based scoring, obtaining an easy-to-read assessment in the form "u is better than $\widehat{v_D}(u)$ people out of 100" (for a specific genre), with $\widehat{v_D}(u) = 100\frac{|\{u' \in U | v_D(u') < v_D(u)\}|}{|U|}$. It can then be used to provide useful information from two different perspectives:

1. Showing to the user a fair evaluation about which music genres they are believed to be more expert about.
2. Presenting to a user interested in learning more about a specific genre which of their friends is considered an expert—making use of the direct knowledge graph defined on page 2.

3 Development

We will now discuss how the presented techniques have been implemented in practice. The final result is Liquid FM: a Facebook application that enable its users to vote one of their friends as an expert on a music genre, and (by means of the described formulas) recommends them some piece of music to listen to, by identifying the best experts.

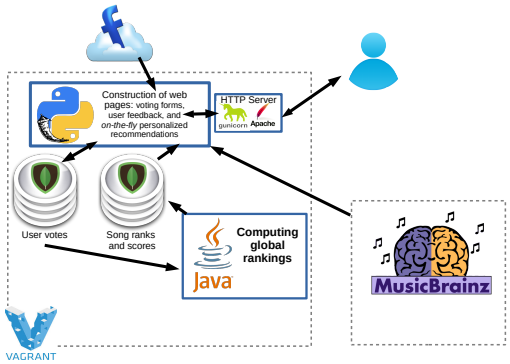


Fig. 4. A schematic representation of the architecture of Liquid FM. An arrow going from A to B indicates data flowing from A to B.

Firstly, we will present a general overview of the architecture of Liquid FM, explaining the role of its main components; then, we will give a more detailed look at the implementations of the formulas presented in Section 2; finally, we will discuss the external components we employed.

Categories As anticipated in the previous section, we applied our scoring algorithms to 9 music genres, called *categories* from now on, and their set will be denoted as C. In this way, we will have different votes and different recommendations for each category. Such a behavior is closer to reality: an expert in HipHop is not assumed to be qualified to give, say, classical music suggestions. However, it also permits to have different graphs for the same users—an interesting fact for future analysis.

The selected categories were Classical, Electronic, Folk, HipHop, Indie, Jazz, Metal, Pop, Rock. They were chosen by inspecting LastFM top 20 tags[3] and discarding those not expressing a musical genre (such as "seen live") and subgenres (having included Indie and Rock, we discarded "Indie Rock"). We decided to add Classical (only ranked 36th on Last Fm), since it is a different and interesting community, under-represented in services such as the one we referred to.

3.1 Overview

As pictured in Figure 4, Liquid FM features two main components:

– a Java part, with the role of analyzing the whole graphs and computing global scores through v_D and r (equations 1 and 2): it is meant to be fast, and executed periodically;

[3] `http://www.last.fm/charts/toptags`

– a Python part, with the role of glueing the different parts together and providing all the other functions: from the construction of web pages to the implementation of personal scores (functions p and c, equations 3 and 4).

These two parts interact with each other through a shared database, that persistently stores every information. We chose MongoDB, an open-source document-oriented NoSQL database, for various reasons:

– We want fast access in reading and writing data (especially very small chunks, as in delegations and votes) in order to be able to support a large amount of users, like in modern recommendation systems. Moreover, we would like our system to be scalable.
– We want flexibility: since this application is also a proof-of-concept, we need to be able to modify data schemas, totally or partially, without much concern.
– Finally, we do not often need complex operations, involving more than one collection. When it occurs, we would like to control what is happening at application-level, permitting fine-grained handling.

On this database, we have two main collections gathering user-submitted data: following the notation introduced in Section 2, the first stores the graph D and the second the graph V. These collections are both represented in Figure 4 as "User votes". A document in the collection for D looks like this[4]:

{ category : c, from : u, to : u' }

While a document in the collection for V has this structure:

{ category : c, user : u, advice : s }

The advice s is a dictionary containing author and title of the song, as well as a YouTube video id. In fact, we associate with each song selected by a user a YouTube video, in order to be able to play it as a recommendation. YouTube is in fact one of the largest and most used music streaming platforms, and it can be included in third-party services (with small limitations). A screenshot of the voting phase in displayed in Figure 5.

Please note that the structure of an advice, as well as the category c, is well-incapsulated: therefore, the schemas of these collections can be easily extended in the future in order to support different (i.e., not music-related) scopes.

3.2 Recommendations

The division of global and personalized recommendations into two separate components originates from efficiency reasons. Having to compute and store all the personalized scores for each user would be impracticable, as they would

[4] Whenever the id of a document is not explicitly expressed, it is automatically generated by MongoDB. This is done efficiently; furthermore, such an id stores the timestamp of creation of the document.

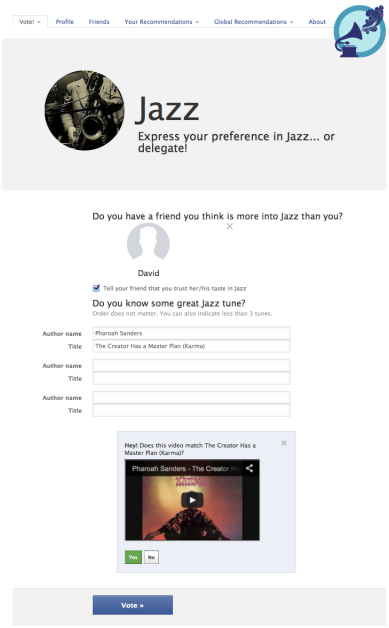


Fig. 5. A screenshot of the voting phase of a user.

be $|C| \cdot |S| \cdot |U|$ scores. Therefore, they are computed with a lazy approach: when a user u asks for her personal recommendations, we compute all of them *on-the-fly* and cache them. Global recommendations, instead, are the main result of the system, and every user depends on them—even to see the personalized scores, since we use the function c (equation 4). For this reason, we compute them periodically with a fast Java component, and save them to a dedicated MongoDB collection.

Global recommendations in Java The global recommendation component was carried out in Java, since for this task it is faster than Python and since efficient open-source libraries to deal with graphs are available; in particular, we employed extensively the WebGraph framework [4] and the `fastutil` library.

This component is run periodically. It takes as input the graphs D and V, memorized in their MongoDB collections, and it results in a new collection for each category, composed of documents of this form:

$$\{ \quad \texttt{advice} : s, \ \texttt{rank} : r(s) \ \}$$

and in another collection ranking users, where each document has this form:

$$\begin{aligned} &\{ \quad \texttt{\_id}: u, \\ &\quad \mathit{category}_1 \texttt{:} \ \{ \ \texttt{score} : v_D(u), \ \texttt{perc} : \widehat{v_D}(u) \ \}, \ \ldots \\ &\} \end{aligned}$$

We can schematize the process, for each category c, in these steps:

1. Read the graph D from MongoDB and convert it to WebGraph format.
2. Use a parallel implementation of the Gauss-Seidel method (from WebGraph) to compute v_D for each user u. We decided to choose 0.75 as the value of α.
3. Compute the percentile-based normalization $\widehat{v_D}$, and save the user-ranking collection to MongoDB.
4. Read the graph V from MongoDB, identifying the set S of songs. In this step, we also find which YouTube video is the most frequently associated with a certain song, using author and title as identifiers. While doing this, we compute $r(s)$ for each song s.
5. Save $r(s)$ in their collection, indexing documents by decreasing scores. Also save $\max_{s \in S} r(s)$, for normalization purposes.

Personalized on-the-fly recommendations As discussed above, personalized recommendations are computed on-the-fly by a Python component. Python was in fact chosen as the main language of the application, due to its versatility and its fast production times; also, we decided to use `Flask`[5], an open-source web development micro-framework particularly suited for our task.

Personalized recommendations are computed only when users ask for them, since they require to see only a very small part of the graphs, and because storing all of them would be unfeasible. The score we will use to rank personalized recommendations is the function $c(s, u)$ (eq. 4); in order to compute it we must, in the first place, compute p (eq. 3).

To compute $p(s, u)$ for all songs $s \in S$ and a fixed user u we walk through the chain of delegations on graph D, starting from u. Since $\forall u\ o_D(u) \leq 1$, this path on D is unique (although it may end in a cycle). Therefore, we simply proceed as follows (for a suitable stopping threshold ϵ):

---

1. Let $\mathtt{p}$ be a map with 0 as default value for missing keys.
2. **while** $t > \epsilon$ **and** $\exists u'$ s.t. $(u, u') \in A_D$
   (a) $u \leftarrow u'$ s.t. $(u, u') \in A_D$
   (b) For each s s.t. $(u, s) \in A_V$:
   $\mathtt{p}[s] \leftarrow \mathtt{p}[s] + t$
   (c) $t \leftarrow t \cdot \alpha$

Now we have all the ingredients for function c, and we can proceed to compute the ranking order according to it.

First of all, consider that the ranking order induced by $c(s)$ is equivalent to

$$\bar{c}(s) = k \cdot p(s, u) + r(s) \qquad \text{where} \qquad k = \frac{\delta \cdot \max_{s' \in S} r(s')}{(1 - \delta) \cdot \max_{s' \in S} p(s', u)}$$

[5] `http://flask.pocoo.org/`

Therefore, for each element s of the map `p`, we multiply its value by k and add the value of $r(s)$. Then, we retrieve all the other items s s.t. $r(s) \geq \min_{s'} \mathtt{p}[s']$, and insert them in the map `p`. Finally, we can build the iterator of personal recommendations by chaining two iterators:

1. the iterator of all elements in `p`, sorted by their values;
2. the iterator of all other elements $s \in S$ having $r(s) < \min_{s'} \mathtt{p}[s']$, sorted by their values; remember that they are already indexed in this order in the database.

The final iterator can be implemented in a lazy fashion, allowing us to retrieve the elements of the second iterator only when necessary. The first iterator, instead, will be computed eagerly upon its request, and then cached. To cache these (and other) values, we employed `redis`[6], an open-source in-memory key-value cache.

3.3 External services

To conclude this section, we would like to briefly describe the main external software components we used in developing Liquid FM.

Facebook As mentioned earlier, Liquid FM is a Facebook application. The reason for it is that we used the Facebook friendship graph as the graph F defined on page 2. In fact, Facebook is at the moment the largest existing social network (with 1.4 billion users), and it has been previously used as a good approximation of an acquaintance graph [1]. Therefore, we require users to have a Facebook account, in order to limit their choice of delegate to their acquaintances. Accordingly, in the collections described earlier, we used a Facebook-provided id[7] to identify a user u.

Musicbrainz To ensure the validity of the set S of songs chosen by users, we check them against the Musicbrainz database. Musicbrainz is an open music database that anyone can edit. At the time of writing, it contained information about more than 900 000 artists and 14 000 000 recorded songs. Since it follows the open-content paradigm, a user who does not find its favorite song in the database is in principle free to add it; however, the community-review process acts as a filter. Furthermore, Musicbrainz provides a disk image to set up a virtual machine with a fully-functioning Musicbrainz server; we used this approach to be able to access the database fast, without network delays and minimizing the impact on their hosts. Moreover, the database of this virtual machine has been set up to self-update itself periodically, in order to adopt every new edit accepted by Musicbrainz.

[6] `http://redis.io/`

[7] To protect users' privacy, this id is valid only within our app, and cannot be used outside of it.

4 Discussion, conclusions and future work

In this work, we presented a Facebook application aimed at putting the viscous democracy framework [3] to the test. This is at the same time a proof-of-concept of how that voting system can be practically implemented in a real-world social network, and a way to collect data corroborating (or disproving) the supposed advantages of viscous democracy when compared to other, more standard, ways of performing elections in a social setting. An interesting point, here, is that the usage of viscous democracy for recommendation seems to avoid the filter bubble [7], at least in its more algorithmic sense, because this kind of recommendation does not rely on collaborative filtering but is based on a conscious choice. Whether this choice (delegation) can itself induce a similar kind of bubble will be subject of future analysis.

The discussed application is currently active on `http://bit.ly/liquidfm`, and we have so far collected some small datasets; currently, the delegation graphs consist of few tens of delegations, so it is impossible to draw any conclusion from them. In order to be able to collect larger amount of information it is crucial that we find a way to make the application *viral*: this is a matter of social engineering that needs to be taken into careful consideration.

References

1. Backstrom, L., Boldi, P., Rosa, M., Ugander, J., Vigna, S.: Four degrees of separation. In: Proceedings of the 4th Annual ACM Web Science Conference. pp. 33–42. ACM (2012)
2. Bernhardsson, E.: Collaborative filtering at spotify. New York Machine Learning meet-up (jan 2013), `http://www.slideshare.net/erikbern/collaborative-filtering-at-spotify-16182818`
3. Boldi, P., Bonchi, F., Castillo, C., Vigna, S.: Viscous democracy for social networks. Communications of the ACM 54(6), 129–137 (2011)
4. Boldi, P., Vigna, S.: The webgraph framework i: compression techniques. In: Proceedings of the 13th international conference on World Wide Web. pp. 595–602. ACM (2004)
5. Katz, L.: A new status index derived from sociometric analysis. Psychometrika 18(1), 39–43 (1953)
6. O'Donell, G.A.: Delegative democracy. Journal of democracy 5(1), 55–69 (1994)
7. Pariser, E.: The filter bubble: What the Internet is hiding from you. Penguin UK (2011)
8. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)

LlamaFur: Learning Latent Category Matrix to Find Unexpected Relations in Wikipedia

Paolo Boldi*
Dipartimento di Informatica
Università degli Studi di Milano
Italy
paolo.boldi@unimi.it

Corrado Monti
Dipartimento di Informatica
Università degli Studi di Milano
Italy
corrado.monti@unimi.it

ABSTRACT

Besides finding trends and unveiling typical patterns, modern information retrieval is increasingly more interested in the discovery of surprising information in textual datasets. In this work we focus on finding *unexpected links* in hyperlinked document corpora when documents are assigned to categories; our approach is based on the determination of a latent category matrix that explains common links; the matrix is built using a perceptron-like technique. We show that our method provides better accuracy than most existing text-based techniques, with higher efficiency and relying on a much smaller amount of information. It also provides higher precision than standard link prediction, especially at low recall levels; the two methods are in fact shown to be orthogonal and can therefore be fruifully combined.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Data Mining; H.3.3 [**Information Storage and Retrieval**]: Information Retrieval; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods

General Terms

Networks; Categorization; Outliers

1. INTRODUCTION

In general, data mining (text mining, if the data involved take the form of textual documents) aims at extracting potentially useful information from some (typically, unstructured, or poorly structured) dataset. The basic and foremost aim of data mining is discovering frequent patterns, and this problem attracted and still attracts a large part of the research efforts in this field. Nonetheless a quite important and somehow dual problem is that of finding unexpected (unusual, new, unforeseen...) information; it is surprising to note that this line of investigation did not receive the same amount of attention.

Albeit there is some research on the determination of surprising information in textual corpora (most often based on the determination of outliers in the distribution of terms or n-grams) there is essentially no work dealing with *unexpected links*. Even if some of the previous proposals exploiting text features can be adapted to this case, a simpler (and, as we here show, more effective) way to approach this problem is by using link prediction algorithms [15], stipulating that a link that is difficult to predict is unexpected.

In this paper, we prove that the availability of some form of categorization of documents can significantly improve the techniques described, leading to algorithms that are extremely efficient, use much less information than text-based methods, and offer better precision/recall trade-offs. Compared to link prediction, our technique also provides higher precision at low recall levels; moreover, the two methods have orthogonal outputs, and therefore their combination improves over both.

Our idea is that if the documents within a linked corpora are tagged with categorical information, one can learn which category/category pairs are more likely to appear, and as a consequence determine which links are unusual (in the sense that they are not "typical"). For example, documents of the category "Actor" often contain links to the category "Movie" (simply because almost all actor pages contain links to the movies they acted in). The fact that George Clooney used to own Max, a 300-pound pig, for 18 years presents itself as a link from an "Actor" page to a page belonging to the category "Pigs"/"Coprophagous animals", which is atypical in the sense above.

Our basic algorithm – henceforth called *LlamaFur*, "Learning LAtent MAtrix to Find Unexpected Relations" – tries to learn a category/category matrix describing the latent relations between categories: to this aim we apply a Passive-Aggressive learning method. Then we reconstruct which part of the graph is more explainable according to the matrix, and which links cannot be justified by the categories alone. Not only LlamaFur is also more efficient than both link prediction and the previous techniques based on the analysis of the textual content of the page, but it also improves the accuracy of link prediction algorithms in identifying unexpected links, if the two are combined.

*Partially supported by the EU-FET grant NADINE (GA 288956).

It is worth noting that the discovery of unexpected links offers a chance to find unknown information: given a certain document, we can highlight text snippets containing unexpected links. Meaningful text is often characterized, in web documents, by the presence of links that enrich its semantic; this is especially true in the case of Wikipedia, often used as a knowledge base for ontologies. Its link structure has proven to be a powerful resource for many tasks [20, 19]. For this reason, finding unexpected links seems a valuable way to detect meaningful text with information unknown to the reader.

Nevertheless, our results could be in principle applied to a plethora of different kinds of objects. The only assumption on the input is a (directed) graph, and a meaningful categorization of its nodes; categories can be overlapping as well, so in fact they may just be some observed features of each object. These assumptions are quite general, and could be applied to many use cases, from the detection of unexpected collaborations between grouped individuals to finding surprising travel habits from geotagged data.

The paper is organized as follows. In Section 2 we will review other works dealing with mining of unexpected information. Our technique will be presented through Section 3, where we explain how to estimate a latent category matrix with online learning; Section 4, where we describe a simpler, naive way to compute it; and Section 5, where we show how to use such a matrix to measure the unexpectedness of a link. In Section 6 we exhibit experimental evidence for the efficacy of our methods, by comparing them with different approaches derived from literature. Finally, in Section 7 we will sum up our work and suggest possible directions for future research.

2. RELATED WORK

One of the first papers trying to consider the problem in the context of text mining was [14]. In that work, two supposedly similar web sites are compared (ideally, two web sites of two competitors). The authors first try to find a match between the pages of the two web sites, and then propose a measure of unexpectedness of a term when comparing two otherwise similar pages. All measures are based on term (or document) frequencies; unexpected links are also dealt with but in a quite simplistic manner (a link in one of the two web sites is considered "unexpected" if it is not contained in the other).

This unexpectedness measure is taken up in [11], where the aim is that of finding documents that are similar to a given set of samples and ordering the results based on their unexpectedness, using also the document structure to enhance the measures defined in [14]. Finding outliers in web collections is also considered in [3], where again dissimilarity scores are computed based on word and n-gram frequency.

Some authors approach the strictly related problem of determining lacking content (called *content hole* in [18]) rather than unexpected information, using Wikipedia as knowledge base. A similar task is undertaken by [9], this time assuming the dual approach of finding content holes in Wikipedia using the web as a source of information.

More recently, [21] considers the problem of finding unexpected related terms using Wikipedia as source, and taking into account at the same time the relation between terms and their centrality.

An alternative way to approach the problem of finding unexpected links is by using *link prediction* [15]: the expectedness of a link e in a network G is the likelihood of the creation of e in $G - \{e\}$. In fact, we will later show that state-of-the-art link prediction algorithms like [1] are very good at evaluating the (un)expectedness of links. Nonetheless, it turns out that the signal obtained from the latent categoy matrix is even better and partly orthogonal to the one that comes from the graph alone, and combining the two techniques greatly improve the accuracy of both.

3. LEARNING THE CATEGORY MATRIX

Consider a directed graph $G = (D, L)$ (the "document graph"), whose nodes $d \in D$ represent *documents* and whose arcs $(d, d') \in L$ represent *(hypertextual) links* between documents. Further assume that we have a set C of *categories* and that each document $d \in D$ is assigned a set of categories $C_d \subseteq C$.

Our first goal is to reconstruct the most plausible latent "category matrix" that explains the observed document graph; more precisely, we wish to find a $C \times C$ real-valued matrix W such that

$$\sum_{c \in C_d} \sum_{c' \in C_{d'}} w_{c,c'} \tag{1}$$

is positive iff $(d, d') \in L$.

We are going to assume that in most cases a relation is *unexpected* – that is, surprising to the reader – if it is poorly explained by a plausible category matrix. We will put this assumption under test in the experimental section.

To find such a matrix W, we recast our goal in the framework of online binary classification. Binary classification is a well-known problem in supervised machine learning. Suppose to have a traning set of *examples*, each one associated with a binary label $\widehat{y}_i \in \{-1, 1\}$; based on these data, the problem is to build a classifier able to label correctly unknown data. *Online* classification simplifies this problem by assuming each example is presented in a sequential fashion; the classifier (1) observes an example; (2) tries to predict its label; (3) receives the true label, and consequentially updates its internal state; (4) moves on to the next example. An online learning algorithm, generally, needs a constant amount of memory with respect to the number of examples, which allows to employ these algorithms in a situation where a very large set of voluminous input data is avaiable – like in our case.

A well-known type of online learning algorithms are the so-called perceptron-like algorithms. They all share these traits: each example must be a vector $\mathbf{x}_i \in \mathbb{R}^n$; the internal state of the classifier is also represented by a vector $\mathbf{w} \in \mathbb{R}^n$; the predicted label is $y_i = sign(\mathbf{w} \cdot \mathbf{x}_i)$, and the algorithms differ on how $\mathbf{w}$ is built. Perceptron-like algorithms (for example, ALMA and Passive-Aggressive) are usually simple to implement, provide tight theoretical bounds, and have been proved to be fast and accurate in practice [10, 8]. For these reasons, we will reduce our problem to online binary

classification.

To this aim, let us represent each document d with the indicator vector of C_d, i.e., with the binary vector $\mathbf{d}$ such that $d_c = 1$ *iff* $c \in C_d$. Now, an *example* will be a pair of documents (d, d'), represented as the outer product kernel $\mathbf{d} \otimes \mathbf{d}'$: this is a matrix where the element $[\mathbf{d} \otimes \mathbf{d}']_{c,c'}$ is 1 iff the first document belongs to c and the second to c'. This $(|C| \times |C|)$-matrix[1] can be alternatively thought of as a vector of size $|C|^2$, allowing us to use them as training examples for a perceptron-like classifier, where the label is $y = 1$ iff $(d, d') \in L$ (if there is a link), and $y = -1$ otherwise. The learned vector $\mathbf{w}$ will be, if seen as a $|C| \times |C|$ matrix, the desired W appearing in (1). In other words, we are using $|C|^2$ features, in fact a kernel projection of a space of dimension $2|C|$ onto the larger space of size $|C|^2$. Similarly the weight vector to be learned has size $|C|^2$. Positive examples are those that correspond to existing links.

A Passive-Aggressive algorithm. Among the existing perceptron-like online classification frameworks, we chose the well-known Passive-Aggressive classifier, characterized by being extremely fast, simple to implement, and shown by many experiments [6, 17] to perform well on similar datasets. To cast this algorithm for our case, let us consider a sequence of pairs of documents

$$(d_1, d'_1), \ldots, (d_T, d'_T) \in D^2$$

(to be defined later). Define a sequence of matrices $W_0, \ldots, W_T$ and of slack variables $\xi_1, \ldots, \xi_T \geq 0$ as follows:

- $W_0 = 0$
- W_{t+1} is a matrix minimizing $\|W_{t+1} - W_t\| + K\xi_{t+1}$ subject to the constraint that

$$\sigma(d_t, d'_t) \cdot \sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{t+1}(c, c') \geq 1 - \xi_{t+1}, \qquad (2)$$

  where

$$\sigma(x, y) = \begin{cases} -1 & \text{if } (x, y) \notin L \\ 1 & \text{if } (x, y) \in L \end{cases},$$

  $\| - \|$ denotes the Frobenius norm and K is an optimization parameter determining the amount of aggressiveness.

The intuition behind the above-described optimization problem [8] is the following:

- the left-hand-side of the inequality (2) is positive iff W_{t+1} correctly predicts the presence/absence of the link (d_t, d'_t); its absolute value can be thought of as the confidence of the prediction;
- we would like the confidence to be at least 1, but allow for some error (embodied in the slack variable ξ_{t+1});
- the cost function of the optimization problem tries to keep as much memory of the previous optimization steps as possible (minimizing the difference with the previous iterate), and at the same time to minimize the error contained in the slack variable.

By merging the Passive-Aggressive solution to this problem with our aforementioned framework, we obtain the algorithm described in Alg. 1.

Algorithm 1 Passive-Aggressive algorithm to build the latent category matrix.

INPUT:
Categories $C_d \subseteq C$ for each document $d \in D$
A sequence $(d_1, d'_1), \ldots, (d_T, d'_T)$ of elements of $D \times D$
A parameter $K > 0$
OUTPUT:
The latent category matrix W

1. $W \leftarrow \mathbf{0}$
2. For $i = 1, \ldots, T$
   (a) $\rho \leftarrow \frac{1}{|C_{d_i}| \cdot |C_{d'_i}|}$
   (b) $\mu \leftarrow \sum_{c \in C_{d_i}} \sum_{c' \in C_{d'_i}} W_{c,c'}$
   (c) **If** $(d_i, d'_i) \in L$
   $\delta \leftarrow \rho \cdot \min(K, 1 - \mu\rho)$
   else
   $\delta \leftarrow -\rho \cdot \min(K, 1 + \mu\rho)$
   (d) For each $c \in C_{d_i}$, $c' \in C_{d'_i}$:
   $W_{c,c'} \leftarrow W_{c,c'} + \delta$

Please note that our aim is not to build a perfect classifier: instead, we will use this algorithm to find a plausible category-category matrix. This can be seen as a rivisitation of the use of classifiers to detect outliers, as described for example in [2].

Sequence of pairs. In our case, W is built through a single-pass online learning process, where we have all positive examples at our disposal (and they are in fact all included in the training sequence), but where negative examples cannot be all included, because they are too many and they would produce overfitting.

The Passive-Aggressive construction described above depends crucially on the sequence of positive and negative examples $(d_1, d'_1), \ldots, (d_T, d'_T)$ that is taken as input. In particular, as discussed in [12], it is critical that the number of negative

[1] In practice, we normalize this matrix so that it has unit $L1$-norm, both because this is a common practice in the perceptron-like algorithms and because documents belonging to few categories provide stronger signals than those that belong to many categories.

and positive examples in the sequence is balanced. Taking this suggestion into account, we build the sequence as follows: nodes are enumerated (in arbitrary order), and for each node $d \in D$, all arcs of the form $(d, -) \in E$ are put in the sequence, followed by an equal number of pairs of the form $(d, -) \notin E$ (for those pairs, the destination nodes are chosen uniformly at random). Of course, if $m = |E|$ is the number of links, then $T = 2m$ and the sequence contains all the m links along with m non-links.

Obviously, there are other possible ways to define the sequence of examples and to select the subset of negative examples. We suggest some of them in Section 7. However, we chose to adopt this technique – single pass on a balanced random sub-sample of pairs – in order to test our methodology with a single, natural and computationally efficient approach.[2]

4. A NAIVE WAY TO BUILD THE CATEGORY MATRIX

Let us describe an alternative, easier, naive variant of how the latent category matrix W could be obtained. Recall that the purpose is to use equation (1) to compute the expectedness of a link (d, d').

For a given category c, let D_c be the set of documents that have the category c; let also $\mathcal{E}_{c,d}$ represent the event that d belongs to the category c (i.e., $c \in C_d$ or, equivalently, $d \in D_c$). Now for any two categories c and c' one can compute the probability that there is a link between two documents that belong to those categories as

$$p_{c,c'} = P[(d, d') \in L \mid \mathcal{E}_{c,d} \text{ and } \mathcal{E}_{c',d'}].$$

This quantity can be naively estimated as the fraction of pairs (d, d') such that $\mathcal{E}_{c,d} \wedge \mathcal{E}_{c',d'}$ that happen to be links. In other words,

$$p_{c,c'} = \frac{|\{(D_c \times D_{c'}) \cap L\}|}{|D_c| \cdot |D_{c'}|}.$$

For a specific pair of documents (d, d'), the probability of the presence of a link is given by

$$P\left[(d, d') \in L \middle| \bigcap_{c \in C_d} \mathcal{E}_{c,d} \text{ and } \bigcap_{c' \in C_{d'}} \mathcal{E}_{c',d'}\right].$$

Now, under some independence assumptions[3], the latter can be expressed as

$$\prod_{c \in C_d} \prod_{c' \in C_{d'}} P\left[(d, d') \in L \middle| \mathcal{E}_{c,d} \text{ and } \mathcal{E}_{c',d'}\right] = \\ = \prod_{c \in C_d} \prod_{c' \in C_{d'}} p_{c,c'}.$$

[2]We carried out experiments performing more passes on the same subsample; it slightly increased (less than 2%) the *accuracy* of W – i.e., the number of pairs that are correctly classified. However, it is dubious whether the increased time cost is worth the limited improvement in terms of unexpectedness mining.

[3]More precisely, we are assuming that $\mathcal{E}_{c,d}$ and $\mathcal{E}_{c',d'}$ are independent, whenever $c \neq c'$ or $d \neq d'$, and also that they are independent even under the knowledge that $(d, d') \in L$.

Applying a logarithm, this is rank-equivalent to

$$\sum_{c \in C_d} \sum_{c' \in C_{d'}} w_{c,c'}$$

where

$$w_{c,c'} = \log p_{c,c'} = \log \frac{|\{(D_c \times D_{c'}) \cap L\}|}{|D_c| \cdot |D_{c'}|}$$

This is yet another way to define the matrix W used in the LlamaFur algorithm; the resulting expectedness score for link (d, d') is given by (1), and will be referred to as Naive-LlamaFur.

5. USING THE CATEGORY MATRIX

Let us now call W the category matrix obtained at the end of the learning process (that is, $W = W_T$, according to the notation of Section 3), or equivalently the matrix built using the naive approach of Section 4. This matrix allows one to sort the links $(d, d') \in L$ in increasing order of $\sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{c,c'}$ (i.e., by increasing explainability): the first links are the most unexpected.

In particular, in the case of the learning approach of Section 3, one can build a graph $G^* = (D, L^*)$ whose links are the set L^* of pairs (d, d') such that

$$\sum_{c \in C_{d_t}} \sum_{c' \in C_{d'_t}} w_{c,c'} \geq 0.$$

In a standard binary-classification scenario, G^* would be the graph G that our classifier learned. In particular, the elements of the set $L \backslash L^*$ ($L^* \backslash L$, resp.) are the false negative (false positive, resp.) instances.

But ours is *not* a link-prediction task, and we do not expect in any sense that L and L^* are similar. In particular, we shall certainly observe a phenomenon that we can call *generalization effect*: suppose that it frequently happens that a document assigned to a category c (e.g., an actor) contains links to documents assigned to another category c' (e.g., a movie). This will probably make $w_{c,c'}$ very large, and so we may falsely deduce that *every* document assigned to c (every actor) contains a link to *every* document assigned to c' (every movie).

The generalization effect will, by itself, make L^* much larger than L (i.e., it will produce many false positive instances), but we do not care much about this aspect. Our focus is *not* on trying to reconstruct L, but rather in understanding which elements of L are difficult to explain based on the categories of the involved documents. We say that a link $(d, d') \in L$ is *explainable* iff $(d, d') \in L^*$; the set of explainable links is therefore $L \cap L^*$. On the contrary, the elements of $L \setminus L^*$ are called unexplainable, and these are the links we want to focus on.

In Figure 1 we show two small examples of how the matrix W learned as in Section 3 looks like, when considering the Wikipedia dataset (for a full explanation of how the dataset was built, see Section 6): in the picture, we display the 18 neighbours closer to two starting categories ("Science Fiction Films" and "Keyboardists"); the width of the arc from c to

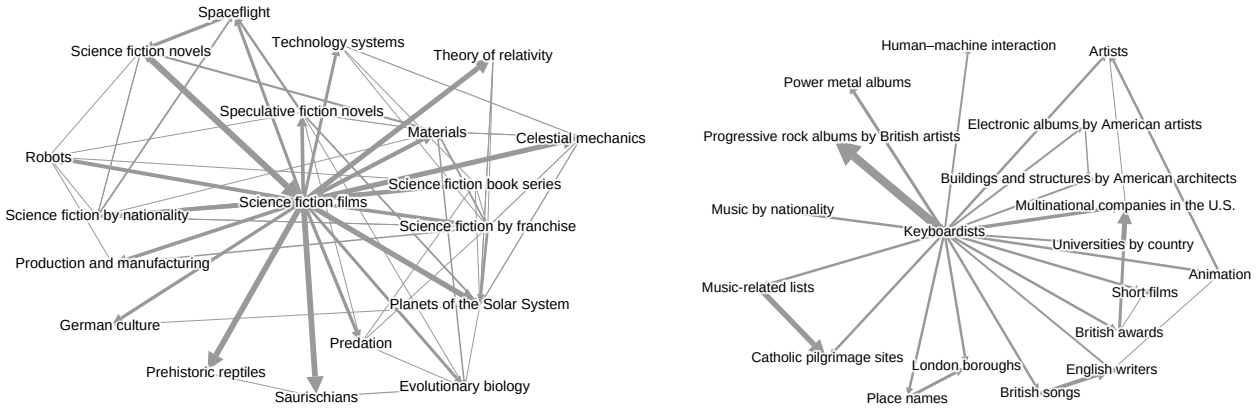


Figure 1: Two fragments of the latent category graph induced by LlamaFur matrix W, representing the 18 closer neighbors of categories "Science Fiction Films" and "Keyboardists", respectively. The width of the arc from c to c' is proportional to $w_{c,c'}$, and arcs with $w_{c,c'} \leq 1$ are not shown.

c' is proportional to $w_{c,c'}$, and arcs with $w_{c,c'} \leq 1$ are not shown.

For example, from the picture it is clear that a link from a page of a science-fiction film to a page of a science-fiction novel is highly expected, as it is one from a page of a keyboardist to one of a british progressive rock album.

The rougher version induced by Naive-LlamaFur is shown in Figure 2.

6. EXPERIMENTS

Given its increasing importance in knowledge representation [20], we used the English edition of Wikipedia as our testbed. In praticular, we employed the `enwiki` snapshot[4] of February 3, 2014 to obtain:

- the document graph, composed by 4 514 662 Wikipedia pages, with 110 699 703 arcs; every redirect was merged to its target page;
- the full categorization of pages: a map associating every page to one of the 1 134 715 categories;
- the category pseudo-tree: a graph built by Wikipedia editors, with the aim of assigning each category to a "parent" category.

Wikipedia categories. The first problem is that the categorization on Wikipedia is quite noisy and, in fact, a continuous work-in-progress: therefore, categories may contain only one (or even no) page, they might be duplicates of each other, and so on. The obvious solution would be to use the pseudo-tree to find the top categories; but the category tree is a work-in-progress itself. Not only – as stated[5] by Wikipedia – "categories can be sub-categories of themselves", but cycles are also present: for example, the largest strongly connected component has 6 833 categories, all direct or indirect subcategories of one another.

We therefore cleansed the page categorization as follows: we computed the harmonic centrality measure [4] on the category pseudo-tree, and considered only the set C of the 20 000 most central categories. To give an idea about the effectiveness of this simple method in capturing the generality of categories, we report in Table 1 the first and the last categories on our list[6].

We then computed, for every category c, the category $\iota(c) \in C$ closest to c in the pseudo-tree, and re-categorized all the pages applying $\iota(-)$ to its original categories. If there is no $c' \in C$ connected to c, $\iota(c)$ is undefined and we simply discarded c. In Table 2 we show some examples of this re-categorization of pages.

In the end, we obtained a set C of 20 000 categories, and a map associating each Wikipedia page d to $C_d \subset C$; on average, each page belongs to 4 categories. We procedeed then to apply LlamaFur to extract the latent category matrix W; the ratio $|L \cap L^*|/|L|$ – that is, how many existing links are explained by W – is equal to 86%. We illustrated previously in Fig. 1 some fragments of W. Finally, we proceeded to assign our unexpectedness score to each link.

Evaluation methodology. We want to evaluate the effectiveness of LlamaFur using the standard framework commonly adopted in Information Retrieval. In our context, a *query* is a document, the possible *results* are the hyperlinks that the document contains, and a result is *relevant* for our problem if it represents an unexpected link. The scenario we have in mind is that of a user wishing to find surprising links in a certain Wikipedia page.

[4]This dataset is commonly referred to as `enwiki-20140203-pages-articles` according to Wikipedia naming scheme.

[5]See *"Known issues"* on `en.wikipedia.org/wiki/Wikipedia:FAQ/Categorization`.

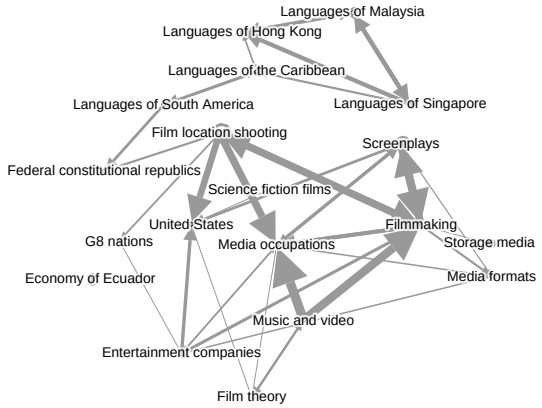
[6]We also excluded utility categories, like *"Categories by country"* and *"Main topic classifications"* – originally highly ranked.

Rank	Category	Rank	Category
1	Countries	19981	Maldives
2	Society	19982	Government buildings on the National Register of Historic Places
3	Nationality	19983	Illinois waterways
4	Political geography	19984	Bodies of water of Illinois
5	Culture	19985	2002 in association football
6	Humans	19986	Electronica albums by British artists
7	Social sciences	19987	Visitor attractions in Arkansas by county
8	Structure	19988	Years of the 20th century in Europe
9	Human–geographic territorial entities	19989	Commonwealth Games events
10	Contents	19990	Albums by English artists by genre
11	Geographic taxonomies	19991	American football in Pennsylvania
12	Fields of history	19992	Ethnic groups in Poland
13	Places	19993	Card games
14	Humanities	19994	Central African people
15	Continents	19995	Deaths by period
16	Political concepts	19996	Visitor attractions in Vermont
17	Human geography	19997	Ancient roads and tracks
18	Subfields of political science	19998	People in finance by nationality
19	Articles	19999	Populated places in Greater St. Louis
20	Subfields by academic discipline	20000	Religion in Poland

Table 1: Topmost and bottommost wikipedia categories according to their harmonic centrality in the Wikipedia category graph.

Original category c	**Substitution** $\iota(c) \in C$
Southern Tang poets	Poets by nationality
Antsiranana Province	Country subdivisions of Africa
Fellows of Magdalen College, Oxford	University of Oxford
Actresses from Greater Manchester	Greater Manchester
Guyanese slaves	History of South America
Swiss manuscripts	Swiss culture
Wilson Pickett songs	Songs by artist
Baroque architecture in Austria	Baroque architecture by country
Eastern Collegiate Roller Hockey Association	∄
Art schools in Washington (state)	Washington (state) culture
Rivers of Kostroma Oblast	Rivers by country
Flamenco compositions	Spanish music
Oil fields of Gabon	Geology of Africa
Basketball teams in Georgia (U.S. state)	Basketball teams in the United States by state
2004 in Australian motorsport	2004 in sports
Populated places established in 1821	∄
Elections in Southwark	Local government in London
Permanent Representatives of Norway to NATO	Ambassadors of Norway
Basketball in Turkey	Basketball by country
Balli Kombëtar	∄

Table 2: An excerpt of the re-categorization process. We write ∄ if there is no category in C connected to c.



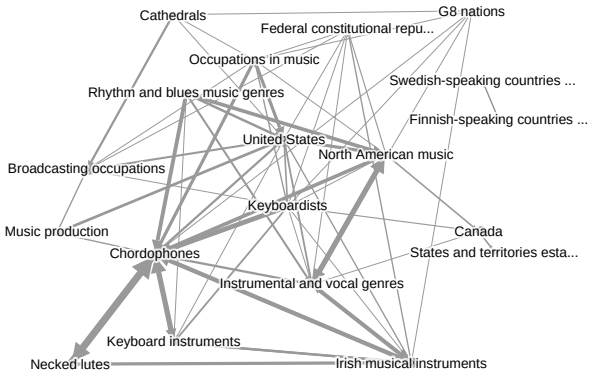


Figure 2: Two fragments of the latent category graph induced by Naive-LlamaFur matrix, representing the 18 closer neighbors of categories "Science Fiction Films" and "Keyboardists", respectively. The width of the arc from c to c' is proportional to $w_{c,c'}$, and the lighter arcs are not shown. For comparison with LlamaFur, see Figure 1.

Label	Fraction
Totally Unexpected	2.3%
Unexpected	8.9%
Expected	30.8%
Totally Expected	58.0%

Table 3: Distribution of labels obtained from human evaluation.

In order to compare the results obtained by LlamaFur with the existing state-of-the-art for similar problems, we performed a user study based on the same pooling method adopted for many standard collections such as TREC (`trec.nist.gov`): we considered a random sample of 237 queries (i.e., Wikipedia documents); for each query we took, among its t possible results (i.e., links), the top-$\lfloor \alpha \cdot t \rfloor$ most unexpected ones according to each system under comparison (see below); all the resulting links were evaluated by human beings. We set $\alpha = 0.1$, and obtained about 3 620 links.

The human evaluators were asked to categorize each link into one of four classes ("totally expected", "slightly expected", "slightly unexpected" and "totally unexpected"). After the human evaluation, we only considered the queries that have at least one irrelevant ("totally/slightly expected") and one relevant ("totally/slightly unexpected") result according to the evaluation, obtaining a dataset with 117 queries. In this dataset, on average each query has 3.45 relevant results over 20.56 evaluated links. The distribution of labels is reported in Table 3.

Baselines and competitors. In our comparison, LlamaFur is tested in combination and against a number of baselines and competitors. In particular, we considereded LlamaFur and its naive variant, Naive-LlamaFur, along with some of the other (un)expectedness measures proposed in the literature.

Albeit there are, at the best of our knowledge, no algorithms specifically devoted to determining unexpected links, we can adapt some techniques used for unexpected documents to our case. All of those methods try to measure the unexpectedness of a document d among a set of retrieved documents R. In our application, we are considering a link (d', d) and taking R to be the set of all documents towards which d' has a hyperlink.

- *Text-based methods.* In the literature, all of the measures of unexpectedness are based on the textual content of the document under consideration.
  - The first index, called M2 in [11] (a better variant of $M1$, the measure proposed in [14]), is defined as:
    $$\mathrm{M2}(d) = \frac{\sum_t U(d, t, R)}{m}$$
    where m is the number of terms in the dictionary, and $U(d, t, R)$ is the maximum between 0 and the difference between the normalized term frequency of term t in document d and the normalized term frequency of t in R (the set of all retrieved documents). The normalized term frequency is the frequency of a term divided by the frequency of the most frequent term.
  - The second index, called M4 in [11] (where they prove that it works better than M2 in their context), is the
    $$\mathrm{M4}(d) = \max_t \mathrm{tf}(t) \cdot \log \frac{|R|}{\mathrm{df}(t)}$$
    where $\mathrm{tf}(t)$ is the normalized term frequency of term t in d, and $\mathrm{df}(t)$ the number of documents in R where t appears.
- *Link-prediction methods.* A completely different, alternative approach to the problem is based on *link prediction*: how likely is it that the link (d', d) is created, if we assume that it is not there? Among the many

techniques for link prediction [15], we tested the well-known *Adamic-Adar index* [1] (AA, in the following), defined[7] by

$$\mathrm{AA}(d, d') = \sum_{d'' \in \Gamma(d) \cap \Gamma(d')} \frac{1}{\log |\Gamma(d'')|},$$

where $\Gamma(d)$ is the set of documents which d links to.

- *Combinations.* Besides testing all the described techniques in isolation, we tried to combine them linearly. Since each unexpectedness measure exhibits a different scale, we first need to normalize each measure by taking its *studentized residual*[8] [7].

Results. In the following, we are only going to discuss the best algorithms and combinations, besides some of the most interesting alternatives. The raw average bpref [5] values are displayed in Table 4. Figure 4 shows, for each algorithm, how many queries have obtained a certain bpref value; for the sake of readability, we have grouped bpref values into four groups. Ideally, an algorithm should produce as many large bpref values as possible. LlamaFur is the one single algorithm that goes closer to the target, whereas M4 and AA are the second best with a small margin. Naive-LlamaFur is worse, wheras M2 is the overall worst. Interestingly, combining pure link prediction methods (AA) with LlamaFur significantly improves AA of about 28%.

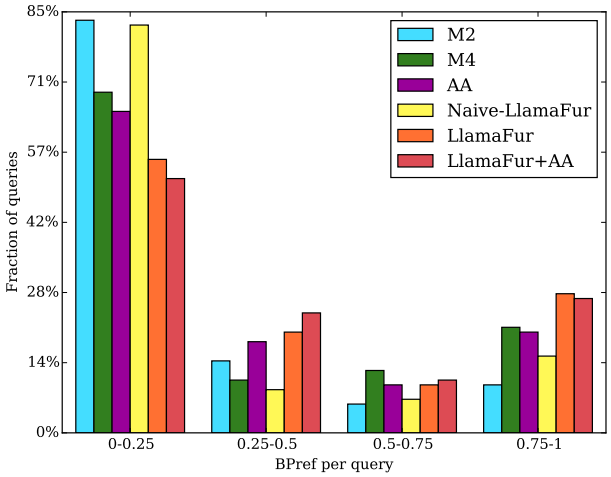


Figure 4: Distribution of the bpref measure over each evaluated query. Average values are shown in Table 4.

Some complementary information about the behaviour is provided by the precision-recall graph of Figure 3: first of all, LlamaFur, AA, M4 and their combinations have larger precision than the remaining ones for almost all the recall levels; on the other hand LlamaFur +AA is the best method for recall values up to 50%, and LlamaFur has definitely better precision than AA until 30% of recall.

In fact, M4, AA and LlamaFur seem to be complementary to one another; in some sense, this is not surprising given that they stem from completely different sources of information: one is based on the textual content, another on the pure link graph and the latter on the category data.

Some further clue on the behaviour of LlamaFur is provided by Figures 5 and 6, where the distribution of LlamaFur and LlamaFur + AA expectedness values is shown for each of the four labels provided by the human evaluation. The red line is the median.

[7]The formula is applied to the symmetric version of the graph, in our case; note that this (like LlamaFur) is a measure of expectedness, whereas M2 and M4 are measures of unexpectedness.

[8]The (internally) studentized residual is obtained by dividing the residual (i.e., the difference from the sample mean) by the sample standard deviation.

Algorithm	Average bpref	Input data
AA	0.288	graph
M2	0.179	bag of words
M4	0.290	bag of words
Naive-LlamaFur	0.216	graph, categories
LlamaFur	0.364	graph, categories
LlamaFur + AA	0.372	graph, categories

Table 4: Average values for bpref.

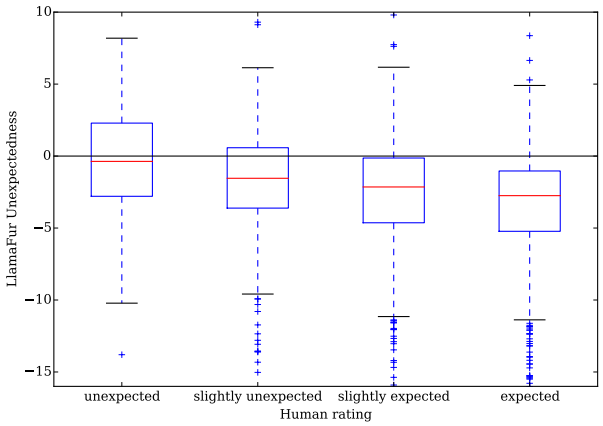


Figure 5: Comparison of the unexpectedness evaluated by LlamaFur with equation (1) over the different labels obtained from human evaluation.

7. CONCLUSIONS AND FUTURE WORK

In this work we presented a technique to find unexpected links in hyperlinked document corpora based on the determination of a latent category matrix that explains common links; the latter is built using a perceptron-like technique. We show that our method provides better accuracy than most existing text-based techniques, with higher efficiency and relying on a much smaller amount of information. An interesting question is whether the latent category matrix can be used to improve link prediction *per se*, i.e. if it is useful to find links and not only unexpected ones: this problem

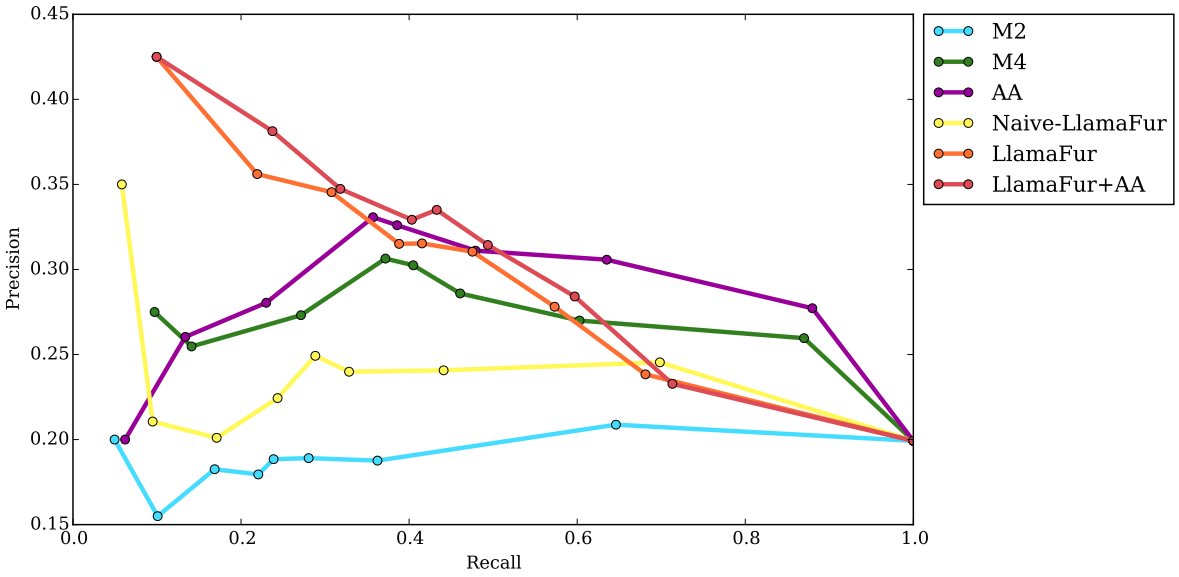


Figure 3: Average precision-recall values evaluated after the 1st, 2nd, 5th, 8th, 10th, 15th, 25th, 50th, and 100th percentiles for each query.

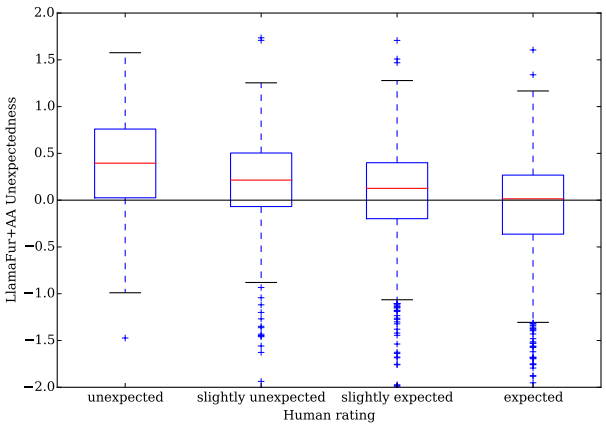


Figure 6: Comparison of the unexpectedness score evaluated by LlamaFur + AA over the different labels obtained from human evaluation.

requires that one finds a way to bypass the generalization effect that the matrix produces.

Another possible direction would be to try different approach to the classification problem described in Section 3, in order to improve its effectiveness. To this aim, one could recast the problem as a cost-sensitive classification where false negatives are more costly than false positives. Other useful techniques include active learning [16]: since we need a subset of the non-linked pairs as counter-examples, active learning would select the more effective ones. An alternative approach to the same task would be to employ one-class learning [13]. This is left as future work.

8. REFERENCES

[1] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25:211–230, 2001.

[2] Charu C Aggarwal. *Outlier analysis*. Springer Science & Business Media, 2013.

[3] Malik Agyemang, Ken Barker, and Reda Alhajj. Hybrid approach to web content outlier mining without query vector. In A. Min Tjoa and Juan Trujillo, editors, *DaWaK*, volume 3589 of *Lecture Notes in Computer Science*, pages 285–294. Springer, 2005.

[4] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014.

[5] Chris Buckley and Ellen M Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[6] Vitor R Carvalho and William W Cohen. Single-pass online learning: Performance, voting schemes and online feature selection. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 548–553. ACM, 2006.

[7] R. Dennis Cook and Sanford Weisberg. *Residuals and*

Influence in Regression. Monographs on Statistics and Applied Probability, 18. Chapman and Hall/CRC, 1983.

[8] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, 2006.

[9] Damien Eklou, Yasuhito Asano, and Masatoshi Yoshikawa. How the web can help wikipedia: A study on information complementation of wikipedia by the web. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '12, pages 9:1–9:10. ACM, 2012.

[10] Claudio Gentile. A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.*, 2:213–242, 2002.

[11] François Jacquenet and Christine Largeron. Discovering unexpected documents in corpora. *Knowledge-Based Systems*, 22(6):421 – 429, 2009.

[12] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[13] Shehroz S Khan and Michael G Madden. A survey of recent trends in one class classification. In *Artificial Intelligence and Cognitive Science*, pages 188–197. Springer, 2010.

[14] Bing Liu, Yiming Ma, and Philip S. Yu. Discovering unexpected information from your competitors' web sites. In Doheon Lee, Mario Schkolnick, Foster J. Provost, and Ramakrishnan Srikant, editors, *KDD*, pages 144–153. ACM, 2001.

[15] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150 – 1170, 2011.

[16] Edwin Lughofer. Single-pass active learning with conflict and ignorance. *Evolving Systems*, 3(4):251–271, 2012.

[17] Corrado Monti, Alessandro Rozza, Giovanni Zappella, Matteo Zignani, Adam Arvidsson, and Elanor Colleoni. Modelling political disaffection from twitter data. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 3. ACM, 2013.

[18] Akiyo Nadamoto, Eiji Aramaki, Takeshi Abekawa, and Yohei Murakami. Content hole search in community-type content. In *Proceedings of the 18th international conference on World wide web*, pages 1223–1224. Association for Computing Machinery, 2009.

[19] S.P. Ponzetto and M. Strube. Deriving a large scale taxonomy from wikipedia. *Proceedings of the National Conference on Artificial Intelligence*, 2:1440–1445, 2007.

[20] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(3):203 – 217, 2008. World Wide Web Conference 2007Semantic Web Track.

[21] Kosetsu Tsukuda, Hiroaki Ohshima, Mitsuo Yamamoto, Hirotoshi Iwasaki, and Katsumi Tanaka. Discovering unexpected information on the basis of popularity/unpopularity analysis of coordinate objects and their relationships. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 878–885. ACM, 2013.