Alleviating cold-user start problem with users' social network data in recommendation systems

Eduardo Castillejo and **Aitor Almeida** and **Diego López-de-Ipiña** [1]

Abstract. The Internet and the Web 2.0 have radically changed the way of purchasing items, provoking the fall of geographic selling barriers all over the world. So large is the amount of data and items we can find in the Web that it turned out to be almost unmanageable. Due to this situation many algorithms have emerged trying to filter items for e-commerce users based in their tastes. In order to do this, these systems need information about the tastes of the users as input. This limitation is reduced as the users interaction with these systems increases. The main problem arises when new users enter a recommendation platform for the first time. The so called cold-start problem causes unsatisfactory random recommendations, which goes against these systems' purpose. Cold-start includes users entering new systems, items, and even new systems. This situation challenges for new ways of obtaining user data. Social networks can be seen as huge information databases sources, and social network analysis would help us to do it using different techniques. In this paper, we present a solution which uses social network user data to generate first recommendations, alleviating the cold-user limitation. Besides, we have demonstrate that it is possible to reduce the cold-user problem applying our solution in a recommendation system environment.

1 INTRODUCTION

The amount of information in the world is increasing far more quickly than our ability to process it [17]. Common users of Web based systems usually have to deal with such amount of data that their interaction can become slow, ending in serious loss of users' attention, which also means losses of sells and user satisfaction. For example, buying a CD or a vinyl in a music store has been an habit from the 70s and 80s. People used to go to the store and navigate through tens or hundreds of albums seeking those which fit with their tastes. But nowadays, with all possibilities Internet offers to every user this amount of items has been increased to millions, even more.

Therefore, taking advantage of the possibilities the Web 2.0 provides to us all, researches started to design algorithms which were able to filter the information (or items) to the user. These algorithms started to compound what we today know as recommender systems. These systems use the opinions of a community of users to help individuals in that community to more effectively identify content of interest from a potentially overwhelming set of choices [16]. Within past years there have been many progresses in this area [4]. Some systems, such as YouTube, started to store some information about users' searches to infer their tastes [8, 5] managing some explicit and implicit information about users interactions. Others, such as Amazon.com[2], take into account the users' ratings and purchases [12]. Both points of view are different sides of the same coin and follow the same purpose: to present to the user the most suitable amount of items.

Despite the advances in this area there are some intrinsic problems that are still unsolved. Probably the most important one is the so called cold-user problem. It emerges every time a new user interacts with recommender system by the first time. Without any search, rating or purchase a recommender system is unable to find any interesting items. Sometimes it even has been necessary many ratings before being able to provide a reasonable recommendation [2].

Given the problem we asked ourselves a simple question: Is there any other way to infer user tastes without their active participation in the system? We think there is, and in this work we pretend to reduce the cited problem taking into account users' social interactions.

Social networks strength lies in the possibility of establishing some social relationships among people, companies and other groups using the Internet as the bridge of communication. The range of accessible social networks is very diverse, from those which just take into account the user location in order to rate a place, bar or store, to those which allows users to share not only their thoughts or beliefs, even their personal pictures, videos and music. There is such amount of information of the users in these networks that new challenges arise to take advantage of it.

This way, our proposal seizes the opportunity of exploiting social network data in order to reduce the cold-start problem in any recommendation system. Applying social network analysis techniques we are able to get some recommendations to the user (with certain accuracy) based on the relationships with others in social networks.

The remainder of this paper is structured as follows: first, in Section 2 we analyse the current state of the art in recommendation systems and the most popular techniques to avoid the cold-user or cold-start problem. Next we present our methodology for collecting valuable data from social networks taking into account users' relationships (Section 3). In Section 4 we analyse the results obtained from our proposal. Finally, in Section 5, we summarize our experiences and discuss the conclusions and future work.

2 RELATED WORK

Since the mid-1990s recommender systems have become an important research area attracting the attention of e-commerce companies. Amazon [12], Netflix[3] and Yahoo! Music[4] [6] are widespread exam-

[1] Deusto Institute of Technology - DeustoTech, University of Deusto, Avda. Universidades 24, 48007 - Bilbao, Spain. email: {eduardo.castillejo, aitor.almeida, dipina}@deusto.es

[2] http://www.amazon.com
[3] http://www.netflixprize.com/
[4] http://music.yahoo.com/

Table 1: Some of the best known metrics in social network analysis.

Metric	Description
Betweenness	It takes into account the connectivity of the node's neighbours to reflect the number. of people who a person is connecting indirectly through their direct links.
Centrality	It gives a rough indication of the social power of a node based on how well they "connect" the network.
Closeness	It reflects the ability to access information through the "grapevine" of network members.
Cohesion	It measures the degree to which actors are connected directly to each other by cohesive bonds.
Degree	The count of the number of ties to other actors in the network.
Eigenvector centrality	A measure of the importance of a node in a network.

ples on making recommendations to its users based on their tastes and previous purchases. Although these systems have evolved becoming more accurate, the main problem is still out there: to estimate the rating of an item which has not been seen by users. This estimation is usually based on the rest of items rated by the current user or on the ratings given by others where the rating pattern is similar to the user's one. Although there are different kinds of recommendation systems (content-based, collaborative filtering and hybrid techniques) [4] they all suffer from the same main limitations: sparsity and scalability [17] and cold-start problems [15].

Some authors have tried to avoid the problem of cold-start users by asking them a series of questions about their tastes or by proposing some studied items in order to get any rating [13, 7]. As we presume these solutions can usually cause displeasure on the users, becoming tedious and cumbersome activities. An-Te Nguyen et al. [4] have tried to reduce the cold-user problem by exploiting available data (e.g. age, occupation, location, etc.). In [18] authors present a metric (the CROC curve) to improve the evaluation of a recommender system performance.

Moreover, some authors have improved their recommendation algorithms combining users' social data from social networks [10, 9]. Although they don't tackle the cold-start problem, their idea of using the available social information of users as an input represents a new starting point for these systems (e.g. Foursquare[5] adds information about user geolocation). There is also an open research in the Carnegie Mellon University's School of Computer Science about how do people really inhabit their cities based on Foursquare data. The project groups check-ins by physical proximity and it measures "social proximity" by how often different people check in similar places. This way resulting areas are dubbed [3].

But in a social network there is more beyond users, relationships and data. Social network analysis (SNA) refers to methods used to analyse social networks, social structures made up of individuals called "nodes", which are connected by different representations of relationships (e.g. friendship, kinship, financial exchange, etc.). Figure 1 represents an example of a social network graph in which different nodes are connected each other by relation lines called "links". Once we have empirical data on a social network new questions arise: Which nodes are the most central members? Which are the most peripheral? Which people are influenced by others? Which connections are most crucial? These questions and their answers represent the basic domain of SNA [14]. There are many metrics which measure different aspects in a social network taking into account the nodes and their edges. Table 1 shows some of the best known metrics in SNA. As depicted in Section 3 we have chosen the eigenvector centrality metric to face up to the cold-start problem. A variant of eigenvector centrality is used by Google search engine to rank Web pages [11], but it is based in the premise that the system already has data from the user to work with.

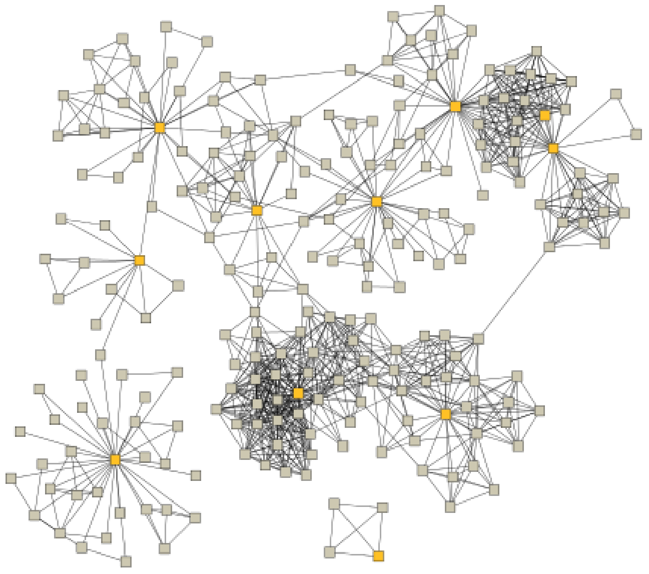


Figure 1: An example of a collaborative social network. Squares represent nodes (people) and the edges represent social ties between them. Yellow squares represent important nodes which relate different sub-graphs [14].

3 PROPOSED SOLUTION

This section details the developed system to enable the generation of generic recommendations to the user based in the rest of the users who checked in the same venue with Foursquare. To get the rest of the users (the network nodes) who checked in the same place we have used the Foursquare API. Once we have the nodes we calculate those which are the most important in the network at the current venue and then we obtain the recommendations which fit better to the user linking using probabilistic.

3.1 Foursquare API

Foursquare is a location-based social networking website which allows users to "check in" at venues using their smartphones. The Foursquare API[6] gives access to all of the data used by the Foursquare mobile applications. Thanks to it developers can request some user data (e.g. location, friends, last check-ins, etc.). We have developed an Android mobile application which allows users to check in desired venues as they would do with Foursquare official application or website (see Figure 3). Once the user checks in any venue, our algorithm is launched: First, we must authenticate the user with the Oauth protocol (required by the Foursquare API). Then we are able to get the nodes who have checked in the current

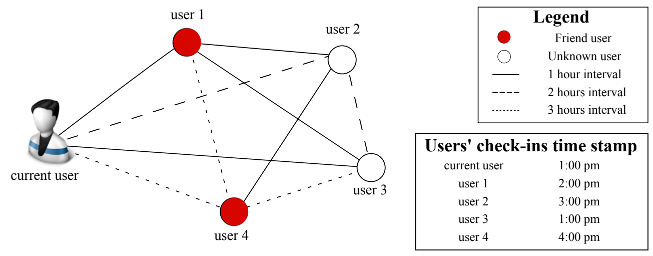
[5] http://www.foursquare.com

[6] https://developer.foursquare.com/

venue. To do this it is necessary to use the "herenow"[7] API endpoint aspect, which responses a count and items (where items are Checkin[8] responses) in JSON format. For example, doing a check-in at the Colosseo, in Rome, we obtained a JSON "herenow" response containing 4 items representing 4 different people who had previously checked in the same venue (see Figure 2). That's the point of the "herenow" endpoint, to get the previous check-ins done by other users at the current checked in venue.

Therefore, we can build a 5×5 square adjacency matrix (3) representing the network graph at the current venue for each user (the first column relates the current user with the others). Each A_{ij} element will be tied to another with a default weight value of 1. Then we start looking to the "createdAt" field of every Checkin object in the "herenow" JSON response. This field is a long number, and it's value is based in the the Unix epoch time (it stores the number of seconds that have elapsed since midnight Coordinated Universal Time (UTC), January 1, 1970). We have split the importance of every check-in in 3 time intervals, adding weights to the values in the matrix in the corresponding A_{ij} position. We decided this because a Foursquare check-in has a lifetime of 3 hours approximately, and that is the reason why we have chosen a 3 hours time interval to give weights to the check-ins. We also believe that people are defined by their acts, and this is why we defend the temporal closeness approach. Being at same places at similar time periods can be used as a tool for relate different people with some and probable similar tastes.

To complete the weights assignation we must take into account that the "herenow" request returns not only the list of the people who have checked in the same venue but also a list of our Foursquare friends who have also done it. Accordingly to this, we believe that our friends have an extra weight (or importance) in the network. The following (1) and (2) equations detail the possible weights distribution when a user checks in a venue. Given priority to user's friends we can see how the maximum weight for an unknown user is 3, although for a friend scales up to 6. We have implemented such algorithm because being friends in a social network does not directly imply that users' tastes are the same.



Users' check-ins time stamp	
current user	1:00 pm
user 1	2:00 pm
user 2	3:00 pm
user 3	1:00 pm
user 4	4:00 pm

Figure 2: Example of a graph built from a user check-in where there are 4 more Foursquare users who had checked in the same venue. Matrix A'' (4) represents the same graph.

$$Unknown\_user = \left\{ \begin{array}{cc} 1 & \text{default weight} \\ +3 & \text{if check-in interval} \leq 1 \text{ hour} \\ +2 & \text{if } 1 \text{ hour} < \text{check-in interval} \leq 2 \text{ hour} \\ +1 & \text{if } 2 \text{ hour} < \text{check-in interval} \leq 3 \text{ hour} \end{array} \right\} \quad (1)$$

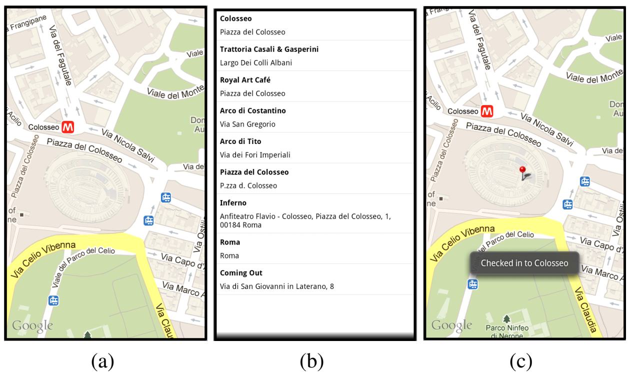
$$Friend\_user = \left\{ \begin{array}{cc} 3 & \text{default weight} \\ +3 & \text{if check-in interval} \leq 1 \text{ hour} \\ +2 & \text{if } 1 \text{ hour} < \text{check-in interval} \leq 2 \text{ hour} \\ +1 & \text{if } 2 \text{ hour} < \text{check-in interval} \leq 3 \text{ hour} \end{array} \right\} \quad (2)$$

[7] https://developer.foursquare.com/docs/venues/herenow
[8] https://developer.foursquare.com/docs/responses/checkin

The following matrices show how the first default values are added just when the adjacency matrix A is built and then how it is completed with the weights assignations (A''). A' is built by adding to A the corresponding values of the relationships between the current user and the others (see Figure 2 and equations (1) and (2)). Regarding at node A''_{01} we can see that it has a value of 6. This is because of the combination of the relationship between the current user and the "user 1" (they are friends) and the check-in time interval (1 hour).

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \quad (3)$$

$$A' = \begin{pmatrix} 0 & 3 & 1 & 1 & 3 \\ 3 & 0 & 1 & 1 & 3 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 3 & 3 & 1 & 1 & 0 \end{pmatrix} A'' = \begin{pmatrix} 0 & 6 & 3 & 4 & 4 \\ 6 & 0 & 3 & 3 & 1 \\ 3 & 3 & 0 & 2 & 3 \\ 4 & 3 & 2 & 0 & 1 \\ 4 & 1 & 3 & 1 & 0 \end{pmatrix} \quad (4)$$



Colosseo
Piazza del Colosseo
Trattoria Casali & Gasperini
Largo Dei Colli Albani
Royal Art Café
Piazza del Colosseo
Arco di Costantino
Via San Gregorio
Arco di Tito
Via dei Fori Imperiali
Piazza del Colosseo
P.zza d. Colosseo
Inferno
Anfiteatro Flavio - Colosseo, Piazza del Colosseo, 1, 00184 Roma
Roma
Roma
Coming Out
Via di San Giovanni in Laterano, 8

Figure 3: A user checks in a venue with developed Android application. First a map is shown (a). Once the user touches the screen a near venues list is presented (b). Finally, when a venue is selected, the check-in is performed (c).

3.2 Eigenvector centrality

As we have depicted in Section 2 a network is a graph made up of points, nodes or vertices tied each other by edges. We can also represent these graphs by a so called adjacency symmetric $(n \times n)$ matrix, where n is the number of nodes. This matrix has elements

$$A_{ij} = \left\{ \begin{array}{cc} 1 & \text{if there is an edge between vertices } i \text{ and } j \\ 0 & \text{otherwise} \end{array} \right\} \quad (5)$$

Eigenvector centrality is a more sophisticated version of the degree metric, which is a simple way to measure the the influence or importance of a node [14]. The degree k_i of a node i is

$$k_i = \sum_{j=1}^{n} A_{ij}, \quad (6)$$

where A is the adjacency matrix which represents the ties between nodes i and j.

Since degree centrality gives a simple count of the number of ties a node has, eigenvector centrality acknowledges that not all connections are equal. Therefore, and because some edges represent stronger connections than others, the edges can be weighted. To sum up, connections to nodes which are themselves influential to others will lend a node more influence than connections to less influential nodes. Denoting the centrality of a node i by x_i, then it is possible to make x_i proportional to the average of the centralities of i's network neighbours:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j, \tag{7}$$

where λ is a constant. This equation can be also rewritten defining the vector of centralities $x = (x_1, x_2, ...)$:

$$\lambda \mathbf{x} = \mathbf{A} \cdot \mathbf{x}, \tag{8}$$

where **x** is an eigenvector of the adjacency matrix with eigenvalue λ.

All these calculations are computed in the user's device. The JAMA [1] library has helped us to easily obtain the most important nodes of the graph (we have also used an online tool[9] to rapidly check the JAMA obtained values). The following method uses this library to obtain the eigenvectors to the given adjacency matrix. First the corresponding eigenvalues are estimated. Then we extract those values of the eigenvectors which are related with the highest value of the obtained eigenvalues, which corresponds to the most important node of the grid. Applying this eigenvector calculation to the A'' matrix from Section 3.1 we obtain that the highest eigenvalue $\lambda_1 = 12.502$, and the corresponding eigenvector to this eigenvalue is

$$e_1 = \begin{pmatrix} 0.569 \\ 0.491 \\ 0.401 \\ 0.392 \\ 0.349 \end{pmatrix}, \tag{9}$$

where the first value corresponds to the current user (and it is also the highest value), so we have to ignore it. The next highest value is the one we will take into account as the most important node of the grid.

We have encapsulated the Foursquare user object into a new "CompactUser" which also has a set of recommendations assigned to it, each one composed by a series of items. Taking as example the generic categories of Amazon.com we have tested our solution using a few controlled users who are friends in Foursquare and some random generated users in order to have a controlled scenario. Results are detailed in Section 4. Once the recommendations of the most important users are obtained, we upload them to a web server by Google AppEngine[10] using a simple Python service. For each user we store all possible recommendations (we manage nine main categories) and we update the estimate and the probability of fitting with his tastes. To evaluate this the developed application asks first about user's tastes among the cited categories. This information is stored in a SQLite database (this is just to evaluate the solution).

The service responses a JSON object with the recommendations and their likelihood probabilities for the user. This JSON object is parsed in the device side in order to generate the corresponding recommendations to the user.

[9] http://www.bluebit.gr/matrix-calculator/
[10] https://appengine.google.com/

4 RESULTS

Our solution has been evaluated by presenting to our test users the default categories that Amazon.com uses and another list with our categories recommendations. Once our users have compared both lists, they have fulfilled a questionnaire to capture their satisfaction level with the obtained results. Amazon.com default recommendations are the following:

- Kindle related products
- Clothing trends
- Products being seen by other customers
- Best watches prices
- Laptops best prices
- Top seller books

These recommendations (not categories) are not based in any user preference. On the contrary our list of recommendations establish a new order within Amazon.com's categories, indicating the probability of each one to be in the tastes of users. Navigating to Amazon.com website with privacy mode enabled allows us to see default recommendations, without taking into account any previous purchase or search. Our objective is focused in recommend items from Amazon.com categories (listed in Table 3).

Testing our solution with real users we obtained a certain approximation to their tastes. Despite the few users, check-ins and data we have access to, the system is capable of generating first generic Amazon.com recommendations (as we have already detailed in this section). Table 3 show the probabilities obtained for a user with the following tastes (see the Amazon.com whole categories listed in Table 2):

- Automotive & industrial
- Movies, music, games
- Electronics & computers
- Sports & outdoors

The middle column shows the system generated probability for each category with just an input of 3 user check-in. On the contrary the right column values corresponds to the same user doing 5 check-ins. These values are more refined, being more in accordance with the user known tastes.

Finally users have to fulfil a questionnaire rating the presented categories. This rating includes mandatory and controlled answers, from 1 to 4. This way we can compare the results with the obtained probabilities. Table 2 shows the probabilities calculated for a user from the resulting questionnaire answers. Then Table 4 compares both probabilities and calculates the approximation of each estimation.

It is important to emphasize that a new matrix is built with every user check-in. This means that previous matrices are overwritten. However, the probabilities are dynamic, and they are refined every time the user checks in a venue.

The more approximate to 0.0 ϵ is, the more accurate our solution becomes. There are some values of ϵ which shows that there are needed more check-ins to refine the obtained probabilities. On the one hand, in case of 3 check-in results the worst ones are for "Automotive & industrial", "Grocery, health & beauty", "Toys, kids & baby" and "Sports & outdoors". This means that if the user is interested in "Sports & outdoors" the system could not recommend any item from this category, or even worse, recommend items from "Grocery, health & beauty". On the other hand, 5 check-in error column is more accurate and its values are closer to 0.0. This test shows how more check-ins come out onto more refined recommendations.

Table 2: Results obtained from a user questionnaire answers. Left column corresponds to the recommendations names. Right column shows the taste probabilities obtained from the questionnaire.

Recommendation	Probability
Home, garden & tools (1)	0.04761904
Clothing, shoes & jewelry (2)	0.09523809
Books (3)	0.14285714
Electronics & computers (4)	0.19047619
Automotive & industrial (5)	0.14285714
Movies, music, games (6)	0.19047619
Grocery, health & beauty (7)	0.04761904
Toys, kids & baby (8)	0.04761904
Sports & outdoors (9)	0.14285714

Table 3: Results obtained from our system for one user performing 3 and 5 check-ins.

Rec.	Probability (3 check-ins)	(5 check-ins)
(1)	0.04347826	0.02222222
(2)	0.04761905	0.13333334
(3)	0.1904762	0.13333334
(4)	0.1904762	0.17391305
(5)	0.0	0.16521739
(6)	0.15	0.16382979
(7)	0.13043478	0.08510638
(8)	0.17391305	0,05319149
(9)	0.0	0,05319149

Table 4: Comparison of probabilities obtained from the questionnaire and the probabilities calculated with the proposed solution using 3 and 5 check-in results.

Rec.	App. 3 check-ins	5 check-ins	3 check-in ε	5 check-in ε
(1)	0.91304360	0.466666695	0.086956393	0.533333305
(2)	0.50000005	1.400000147	0.499999947	-0.400000147
(3)	1.33333342	0.933333399	-0.333333426	0.066666601
(4)	1.00000005	0.913043515	-0.000000053	0.086956485
(5)	0.0	1.156521753	1.0	-0.156521753
(6)	0.78750000	0.8601064	0.212499998	0.1398936
(7)	2.73913081	1.787234266	-1.73913081	-0.787234266
(8)	3.65217463	1.117021469	-2.65217463	-0.117021469
(9)	0.0	0.372340437	1.0	0.627659563

Approximation (App.) = (Solution probability / Questionnaire probability)
Deviation error (ε) = 1 - approximation

5 CONCLUSIONS AND FUTURE WORK

This paper explores the possibility of using relevant data from users' social network to alleviate the cold-user problems in a recommender system domain. The proposed solution extracts the most valuable node in the graph generated by check in a venue with an Android application using the Foursquare API. By obtaining the recommendations to this node we estimate the probability of some categories to be similar to users tastes.

In the near future we will take into account data not only from Foursquare. Other social networks with accessible APIs will be useful enough to determinate more accurately the preferred items for a user. Moreover, combining different social network analysis metrics can come out onto more accurate results. Another important aspect is to take into account more than the most valuable node for doing recommendations.

It will be also interesting to store the obtained matrices for each venue and update them with every check-in. By now matrices are not stored, so they are not dynamic, which means that a new matrix is built every time the user checks in a venue, overwriting any other previous matrix.

Finally, it becomes necessary to test the solution among a higher number of users, increasing their tastes possibilities and the offered items. We are limited by our environment because of the small amount of users and check-ins available. Therefore a dissemination of this work would be very useful to get more real data.

REFERENCES

[1] Jama : A java matrix package. `http://math.nist.gov/javanumerics/jama/`.

[2] Movielens movies recommendation service. `http://movielens.umn.edu`.

[3] Using foursquare data to redefine a neighborhood. `http://www.technologyreview.com/web/40224/`.

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

[5] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, 'Video suggestion and discovery for youtube: taking random walks through the view graph', in *Proceedings of the 17th international conference on World Wide Web*, pp. 895–904. ACM, (2008).

[6] P.L. Chen, C.T. Tsai, Y.N. Chen, K.C. Chou, C.L. Li, C.H. Tsai, K.W. Wu, Y.C. Chou, C.Y. Li, W.S. Lin, et al., 'A linear ensemble of individual and blended models for music rating prediction', in *KDDCup 2011 Workshop*, (2011).

[7] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, 'Combining content-based and collaborative filters in an online newspaper', in *Proceedings of ACM SIGIR Workshop on Recommender Systems*, pp. 1–11. Citeseer, (1999).

[8] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al., 'The youtube video recommendation system', in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 293–296. ACM, (2010).

[9] H. Kautz, B. Selman, and M. Shah, 'Referral web: combining social networks and collaborative filtering', *Communications of the ACM*, **40**(3), 63–65, (1997).

[10] I. Konstas, V. Stathopoulos, and J.M. Jose, 'On social networks and collaborative recommendation', in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 195–202. ACM, (2009).

[11] A.N. Langville, C.D. Meyer, and P. FernÁndez, 'Google's pagerank and beyond: The science of search engine rankings', *The Mathematical Intelligencer*, **30**(1), 68–69, (2008).

[12] G. Linden, B. Smith, and J. York, 'Amazon. com recommendations: Item-to-item collaborative filtering', *Internet Computing, IEEE*, **7**(1), 76–80, (2003).

[13] P. Melville, R.J. Mooney, and R. Nagarajan, 'Content-boosted collaborative filtering for improved recommendations', in *Proceedings of the National Conference on Artificial Intelligence*, pp. 187–192. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, (2002).

[14] M.E.J. Newman, 'The mathematics of networks', *The New Palgrave Encyclopedia of Economics*, **2**, (2008).

[15] A.T. Nguyen, N. Denos, and C. Berrut, 'Improving new user recommendations with rule-based induction on cold user data', in *Proceedings of the 2007 ACM conference on Recommender systems*, p. 121–128, (2007).

[16] P. Resnick and H. R. Varian, 'Recommender systems', *Communications of the ACM*, **40**(3), 56–58, (1997).

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, 'Item-based collaborative filtering recommendation algorithms', in *Proceedings of the 10th international conference on World Wide Web*, p. 285–295, (2001).

[18] 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*, p. 253–260, (2002).