The 22nd International Joint Conference on Artificial Intelligence

Proceedings of the International Workshop on Social Web Mining[1]

http://users.cecs.anu.edu.au/~sguo/swm.html

Universitat de Barcelona, Spain, 18 July 2011

Editors:

Francesco Bonchi
Wray Buntine
Ricard Gavaldà
Shengbo Guo

[1]Sponsored by PASCAL 2

Preface

Social networks have drawn increasingly larger number of users worldwide for the last decade, thanks to the success deployment of social platforms such as Facebook, LinkedIn, Flickr, Delicious, Douban, and many others. Information underlying the social networks has proved to be extremely beneficial for diverse applications such as recommender systems, electronic commerce, and even search engines, just to name a few.

The rapid growth of social networks brings unprecedent commercial opportunities to the machine learning and data mining communities, but there are many open problems for analyzing social networks. Some are:

- How could one build optimal models for social networks such as Facebook?
- How can one handle the privacy issue caused by utilizing social interactions for making recommendation?
- How could one model a user's preferences based on his/her social interactions?

The goal of this workshop is to gather scientific researchers and industry to review the state of the art in social web mining, discuss the challenges, exchange ideas, and promote collaborations across different groups.

The call for papers for our workshop has attracted many good submissions. All submitted papers were thoroughly reviewed by the Program Committee, and we finally decided to accept 7 papers. We thank the Program Committee members for their carefully reviewing all the submissions. We also thanks all the authors for their contributions. Finally, we would also like to PASCAL 2 for the sponsorship.

Program Committee Co-Chairs:
Francesco Bonchi, Yahoo! Research Barcelona, Spain
Wray Buntine, NICTA - ANU, Australia
Ricard Gavaldà, Technical University of Catalonia, Spain
Shengbo Guo, Xerox Research Centre Europe, France

Program Committee

- Jean-Marc Andreoli, Xerox Research Centre Europe, France
- Cedric Archambeau, Xerox Research Centre Europe, France
- Francesco Bonchi, Yahoo! Research Barcelona, Spain
- Guillaume Bouchard, Xerox Research Centre Europe, France
- Wray Buntine, NICTA - ANU, Australia
- Tiberio Caetano, NICTA - ANU, Australia
- Wei Chen, Microsoft Research Asia, China
- Boris Chidlovskii, Xerox Research Centre Europe, Grenoble, France
- Peter Christen, Australian National University, Australia
- Nello Cristianini, University Of Bristol, UK
- Shengbo Guo, Xerox Research Centre Europe, France
- Hakim Hacid, Alcatel-Lucent Bell Labs, France
- Jian Huang, Google Pittsburgh, USA
- Jure Leskovec, Stanford University, USA
- Ernesto William De Luca, Technical University of Berlin - DAI-Labor, Germany
- Sherif Sakr, NICTA - UNSW, Australia
- Scott Sanner, NICTA - ANU, Australia
- Markus Schedl, Johannes Kepler University Linz, Austria
- Fabrizio Silvestri, ISTI CNR, Italy
- Julia Stoyanovich, University of Pennsylvania, USA
- Aixin Sun, Nanyang Technological University, Singapore
- Paul Thomas, CSIRO, Australia
- Antti Ukkonen, Yahoo! Research Barcelona, Spain
- Jie (Jessie) Yin, CSIRO, Australia
- Yi Zhang, University of California, Santa Cruz, USA
- Onno Zoeter, Xerox Research Centre Europe, Grenoble, France

Program

9:00 - 9:15	Welcome
9:15 - 10:15	*Recommending information sources to information seekers in Twitter* Marcelo G. Armentano, Daniela Godoy, Analia A. Amandi. *Analyzing the potential of Microblogs for spatio-temporal popularity estimation of music artists*, Markus Schedl. *Measuring semantic similarity using a multi-tree model*, Behnam Hajian, Tony White.
10:15 - 11:00	*The Blogosphere at a glance—content-based structures made simple*, Olof Gornerup, Magnus Boman. *Classification of social network sites based on network indexes and communication patterns*, F. Toriumi, I. Okada, H. Yamamoto, H. Suwa, K. Izumi, Y. Hashimoto.
11:00 - 11:30	Coffee break
11:30 - 12:30	**Invited talk**: *Transfer learning in social recommendations*, Qiang Yang.
12:30 - 13:00	*System-wide effectiveness of active learning in collaborative filtering*, Mehdi Elahi, Francesco Ricci, Valdemaras Repsys.
13:00 - 14:30	Lunch break
14:30 - 16:30	**Invited talk**: *Android App discovery: supporting users through the app discovery funnel*, Bhaskar Mehta. **Invited talk**: Social recommender systems: a graphical model based approach, Jurgen Van Gael.
16:30 - 17:00	Coffee break
17:00 - 18:00	**Invited talk**: *Understanding social media*, Ricardo Baeza-Yates.
18:00 - 18:30	Closing discussion

The Blogosphere at a Glance—Content-Based Structures Made Simple

Olof Görnerup[1] and **Magnus Boman**[1,2]
[1]Swedish Institute of Computer Science (SICS), SE-164 29 Kista, Sweden
[2]Royal Institute of Technology (KTH/ICT/SCS), SE-164 40 Kista, Sweden
{olofg, mab}@sics.se

Abstract

A network representation based on a basic word-overlap similarity measure between blogs is introduced. The simplicity of the representation renders it computationally tractable, transparent and insensitive to representation-dependent artifacts. Using Swedish blog data, we demonstrate that the representation, in spite of its simplicity, manages to capture important structural properties of the content in the blogosphere. First, blogs that treat similar subjects are organized in distinct network clusters. Second, the network is hierarchically organized as clusters in turn form higher-order clusters: a compound structure reminiscent of a blog taxonomy.

1 Introduction

Several tools and algorithms have been developed for harnessing the vast amount of data that constitutes the blogosphere (cf. [Agarwal and Liu, 2008; 2009]), e.g., by collecting, relating and visualizing blog entries [Tauro *et al.*, 2008; Llor *et al.*, 2007; Uchida *et al.*, 2007; Bross *et al.*, 2010] or tag clouds, [Fujimura *et al.*, 2008], or by classifying blogs in terms of interblog communication and community stability [Chi *et al.*, 2007], sense of community among bloggers [Chin and Chignell, 2006], discussion keyword correlation [Bansal *et al.*, 2007], and a host of machine learning and statistics approaches, cf. [Tsai, 2011]. To date, however, almost all these tools and algorithms require human intervention and considerable time investment to overcome problems with bootstrapping, tuning, and not least semantics. Understanding a graph, perhaps with thousands of vertices and edges, pertaining to describe relevance to one's own blog according to some set of possibly esoteric or advanced criteria is not straightforward. We address this problem by presenting a method for generating a network of relevant blogs by means of the simplest similarity criterion there is: word overlap. We will demonstrate that even this naïve approach allows us to capture fundamental and important structural properties of the blogosphere.

2 Method

We represent the blogosphere as a network, where nodes constitute blogs, and where blogs are linked if they have similar textual content. Links are weighted, where the strength of a link is given by a similarity measure.

2.1 Similarity measure

To estimate the similarity between blogs we simply compare the overlap of occurring words. Given two blogs i and j, let $\mathcal{W}_i$ denote a set of words (to be specified below) that occur in i, and $\mathcal{W}_j$ a set of words that are used in j. The similarity s_{ij} between i and j is then defined as the Jaccard index

$$s_{ij} = \frac{|\mathcal{W}_i \cap \mathcal{W}_j|}{|\mathcal{W}_i \cup \mathcal{W}_j|}. \quad (1)$$

In other words, s_{ij} is the fraction of all words in $\mathcal{W}_i$ and $\mathcal{W}_j$ that are shared by the two sets. It holds that $0 \leq s_{ij} \leq 1$, where $s_{ij} = 1$ if $\mathcal{W}_i$ and $\mathcal{W}_j$ are identical and $s_{ij} = 0$ if they do not share a single word. This similarity measure is equivalent to Tversky's Ratio model [Tversky, 1977], which has been found to be a good trade-off between simplicity and performance among text document similarity measures [Lee *et al.*, 2005].

2.2 Word filtering

We do not consider the full word sets of blogs—literally *all* occurring words—for several reasons. Comparing very common words ("the", "it", "do", etc.) will only provide a negligible amount of similarity information. The use of uncommon words, on the other hand, is likely to tell us a lot about the characteristics of a blog. However, at the same time we do not want to consider words that are too uncommon—for instance those occurring only a handful of times in the blogosphere during the course of several months—since these are often misspellings and typos that only add noise to the statistics. Another reason for not considering all words is a pragmatic one. Analyzing tens of thousands of blogs can be computationally expensive. By utilizing Zipf's law [Zipf, 1949], which implies that a few of the most common words represent a large majority of word occurrences[1], the computational cost is drastically reduced.

[1]More specifically, the frequency of a word is inversely proportional to its rank; $f_n \sim 1/n^a$, where n is the rank ($n = 1$ for the most common word, $n = 2$ for the second most common word, etc.) and a is some exponent.

2.3 Network structure

The global structure of a similarity network may provide valuable information about how blogs and groups of blogs are related with respect to contents. We have focused on two network properties: Community structure and hierarchical organization.

Complex networks typically exhibit communities, where nodes are clustered in groups [Newman, 2003]. Characteristic for community structures is that there are significantly higher densities of edges within communities than between them. This property may be quantified as follows [Newman and Girvan, 2004]: Let $\{v_1, v_2, ..., v_n\}$ be a partition of a set of vertices into n groups, r_i the degree of edge weights (i.e., similarities) internal to v_i (the sum of internal weights over the sum of all weights in the network) and s_i the degree of weights of edges that start in v_i. The degree of community structure is then defined as

$$Q = \sum_{i=1}^{n} (r_i - s_i^2). \quad (2)$$

To infer clusters in the blog network we have employed an agglomerative clustering technique [Clauset, 2005], that aims to find cluster assignments—a partition of the set of vertices—that maximizes the community structure measure Q.

Another method by Clauset *et al.* [Clauset *et al.*, 2008] has been used to identify the hierarchical structure of the blog network. This method combines a maximum likelihood approach with a Monte Carlo sampling procedure to infer likely hierarchical models of the network.

2.4 Case study: The Swedish blogosphere

We have tested our approach on the Swedish blogosphere. The API of the blog search engine *Twingly*[2] has been used for collecting blog posts from a five-month period. The posts were fetched and aggregated (i.e., for each blog, posts were concatenated). In the spirit of keeping things simple we refrained from applying *ad hoc* textbook pre-processing such as stemming and relied on basic word frequency statistics to filter out words: First we discarded all words occurring less than ten times. Of the remaining words we then kept those that occurred in the fifth percentile of the frequency distribution. For each blog, we collected its set of those occurring words. Blogs that had word sets of size 25 or larger were kept. This ensured a meaningful similarity measure and also filtered out a considerable amount of spam blogs. At this point, 21564 blogs remained. We have varied the above parameters in sensitivity analyses, and the results reported here appear to be stable.

3 Results

The content-based blog network is found to have a distinct clustered structure. We have visualized this by plotting edges with a weight above a certain threshold (i.e. only relations between highly similar blogs are shown) such that blog communities crystallize into separate subnetworks. See Fig. 1, where we plot the acquired network with various weight thresholds. By inferring communities and then inspecting the actual content of blogs within communities, we find that the clusters reflect topics domains such as politics, books, technology, or music, cf. Fig. 2. Note that spam blogs, *splogs*, also form separate clusters. Splogs are in fact particularly tightly knit, presumably since they tend to contain homogenous sets of words.

Furthermore, when employing Clauset *et al.*'s hierarchy inference algorithm, we find that clusters indeed are organized in higher order (meta-) clusters. An example of the hierarchical organization of the blog network is depicted in Fig. 3 in the form of a consensus dendrogram—i.e., a dendrogram that is consistent with several inferred hierarchical models—of a "food and beverages" cluster. There we see that food and beverages are separated into two clusters, and the beverage cluster in turn consists of a wine and a beer cluster. Again, the validity of acquired hierarchies is evaluated by inspection.

4 Discussion and outlook

We have shown that the signal in raw blog data is so strong that even our basic similarity measure—word occurrence overlap—is capable of capturing valuable structural information. The measure is computationally tractable and enables efficient categorization of blogs when used in concurrence with fast graph clustering algorithms. We grant that there are more advanced—and possibly more accurate—(document) similarity measures [Agarwal *et al.*, 2008; Elsas *et al.*, 2008; Lee *et al.*, 2005; Macdonald and Ounis, 2008]. However, we believe that the minimal (non-trivial) measure employed here is suitable as a baseline when studying blog similarity networks. The measure is admittedly simplistic, yet this is also its strength since it decreases the risk of causing hidden representation-dependent artifacts that are more difficult to identify when using more advanced similarity measures.

Because of the rapid growth of data in the blogosphere, there is a strong demand from industry as well as from research for simple means to harvesting blog data. Our approach is obviously among the simplest possible, but we have not discussed any explict applications here, since employment is not our chief concern. Neither have we provided any analyses of computational complexity, because such analyses will be application-driven and will likely contain very detailed average-case, rather than general worst-case, complexity measures.

An issue that needs to be addressed in future work is that of validation. How can we know that acquired blog clusters are meaningful? So far, our approach has been to examine a random sample of blogs and subjectively confirm that their contents is consistent within inferred blog clusters. Such empirical evaluations can be problematic, however. In some cases, a manual classification may be considered as clear cut (e.g., identifying that two blogs that solely treat Belgian beer belong to the same cluster), but not always. A more quantitative measure that validates the result is therefore desirable. This can, on the other hand, also be turned into an epistemological question. One can for example imagine cases when the blog classes acquired from the similarity network can be used to evaluate *other* blog classifications (including our own subjec-

[2] http://www.twingly.com/

tive one). However, in this discussion we have more pragmatic and application-oriented evaluation methods in mind.

We have treated only a few structural aspects of the blog network here. These deserve more attention, as do the dynamics and evolution of the networks: How does information diffuse and change in the network, and how does the network structure itself change over time? For instance, through an analysis along these lines one may perhaps trace how emerging trends or news proliferate in and between specific topic domains of the blog similarity network.

Another possible future direction concerns splog detection. We have observed that splogs emerge as separate categories. If an individual blog is identified as a splog (e.g., by examining the distribution of blog similarities), it is likely that its associated blog cluster also consists of splogs. If such a relation proves to hold true in general, it enables splog detection and removal at the level of blog clusters rather than individual blogs, which presumably would be much more efficient.

As the network representation of the blogosphere is found to be hierarchically structured, it may pave the way for applications that operate on different levels of resolution; from blogs to groups of similar blogs, to groups of groups of blogs, and so forth. The hierarchical organization also enables a top down approach to blog navigation that starts at a coarse level of blog categories and then narrows down to finer scales. Monitoring may also be more efficient and accurate if limited to a specific and relevant topic-domain of blogs. That is, although the blogosphere may seem overwhelming at times, it is in fact intrinsically structured in terms of content as to enable effective navigation and monitoring.

Acknowledgements

OG was funded by The Internet Infrastructure Foundation (.SE). The authors thank Twingly for providing blog data and Aaron Clauset for sharing source code for the hierarchical structure inference algorithm and for the radial dendrogram visualization script used for rendering Fig. 3. The authors also thank Jussi Karlgren for providing some of the references to earlier work.

References

[Agarwal and Liu, 2008] Nitin Agarwal and Huan Liu. Blogosphere—research issues, tools, and applications. *SIGKDD Explorations*, 10(1):18–31, 2008.

[Agarwal and Liu, 2009] Nitin Agarwal and Huan Liu. *Modeling and Data Mining in Blogosphere*. Morgan and Claypool Publishers, 2009.

[Agarwal *et al.*, 2008] Nitin Agarwal, Huan Liu, Lei Tang, and Philip S. Yu. *Identifying the Influential Bloggers in a Community*. In *Proceedings of the international conference on Web search and web data mining*. ACM, New York, 2008.

[Bansal *et al.*, 2007] Nilesh Bansal, Nick Koudas, Fei Chiang, and Frank Wm. Tompa. Seeking stable clusters in the blogosphere. In *Proceedings of the 33rd international conference on Very large data bases*, pages 806–817. VLDB Endowment, 2007.

[Bross *et al.*, 2010] Justus Bross, Matthias Quasthoff, Philipp Berger, Patrick Hennig and Christoph Meinel Mapping the Blogosphere with RSS-Feeds. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 453–460. IEEE Computer Society, Washington, DC, 2010.

[Chi *et al.*, 2007] Yun Chi, Shenghuo Zhu, Xiaodan Song, Junichi Tatemura, and Belle Tseng. Structural and temporal analysis of the blogosphere through community factorization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 163–172. ACM, New York, 2007.

[Chin and Chignell, 2006] Alvin Chin and Mark Chignell. A social hypertext model for finding community in blogs. In *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 11–22. ACM, New York, 2006.

[Clauset *et al.*, 2008] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

[Clauset, 2005] Aaron Clauset. Finding local community structure in networks. *Physical Review E*, 72:026132, 2005.

[Elsas *et al.*, 2008] Jonathan L. Elsas, Jaime Arguello, Jamie Callan, and Jaime G. Carbonell. *Retrieval and feedback models for blog feed search*. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, New York, 2008.

[Fujimura *et al.*, 2008] Ko Fujimura, Shigeru Fujimura, Tatsushi Matsubayashi, Takeshi Yamada and Hidenori Okuda. Topigraphy: visualization for large-scale tag clouds. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1087–1088. ACM, New York, 2008.

[Lee *et al.*, 2005] Michael D. Lee, Brandon Pincombe, and Matthew Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1254–1259. Erlbaum, 2005.

[Llor *et al.*, 2007] Xavier Llor, Noriko Imafuji Yasui, Michael Welge, and David E. Goldberg. Human-centered analysis and visualization tools for the blogosphere. In *Proceedings of the Digital Humanities 2007*, 2007.

[Macdonald and Ounis, 2008] Craig Macdonald and Iadh Ounis. Key blog distillation: ranking aggregates. In *Proceedings of the 17th ACM Conference on Information and knowledge management*. ACM, New York, 2008.

[Newman, 2003] Mark Newman. The Structure and Function of Complex Networks. *SIAM Review*, 2:167–256, 2003.

[Newman and Girvan, 2004] Mark Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 2004.

[Tauro *et al.*, 2008] Candida Tauro, Sameer Ahuja, Manuel A. Prez-Quiones, Andrea Kavanaugh, and Philip Isenhour. Vizblog: Discovering conversations in the blogosphere. In *Technology demonstration at Directions and Implications of Advanced Computing - Conference on Online Deliberation*, University of California, Berkeley, 2008.

[Tsai, 2011] Flora S. Tsai. Dimensionality reduction techniques for blog visualization. *Expert Systems with Applications*, 38(3):2766–2773, 2011.

[Tversky, 1977] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.

[Uchida *et al.*, 2007] Makoto Uchida, Naoki Shibata, and Susumu Shirayama. Identification and visualization of emerging trends from blogosphere. In *Proceedings of International Conference on Weblogs and Social Meida*, pages 305–306, 2007.

[Zipf, 1949] George K. Zipf. *Human Behaviour and the Principle of Least Effort*. Addison Wesley, Cambridge MA, 1949.

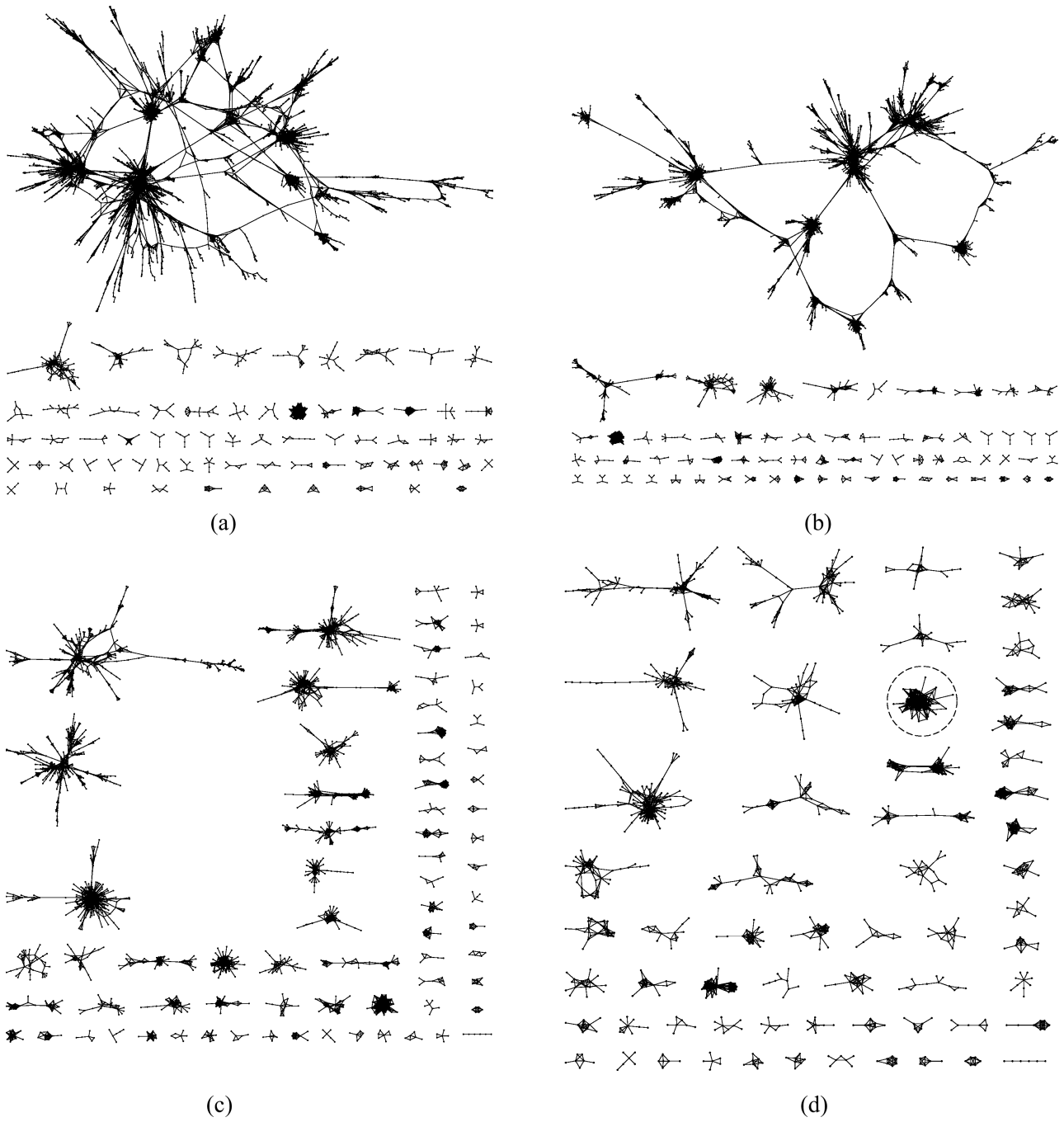


Figure 1: Visualization of the Swedish blogosphere, where blogs with similarities $\geq \gamma$ are shown. (a) $\gamma = 0.04$. (b) $\gamma = 0.045$. (c) $\gamma = 0.055$. (d) $\gamma = 0.07$. A spam blog cluster is enclosed within a dashed circle.

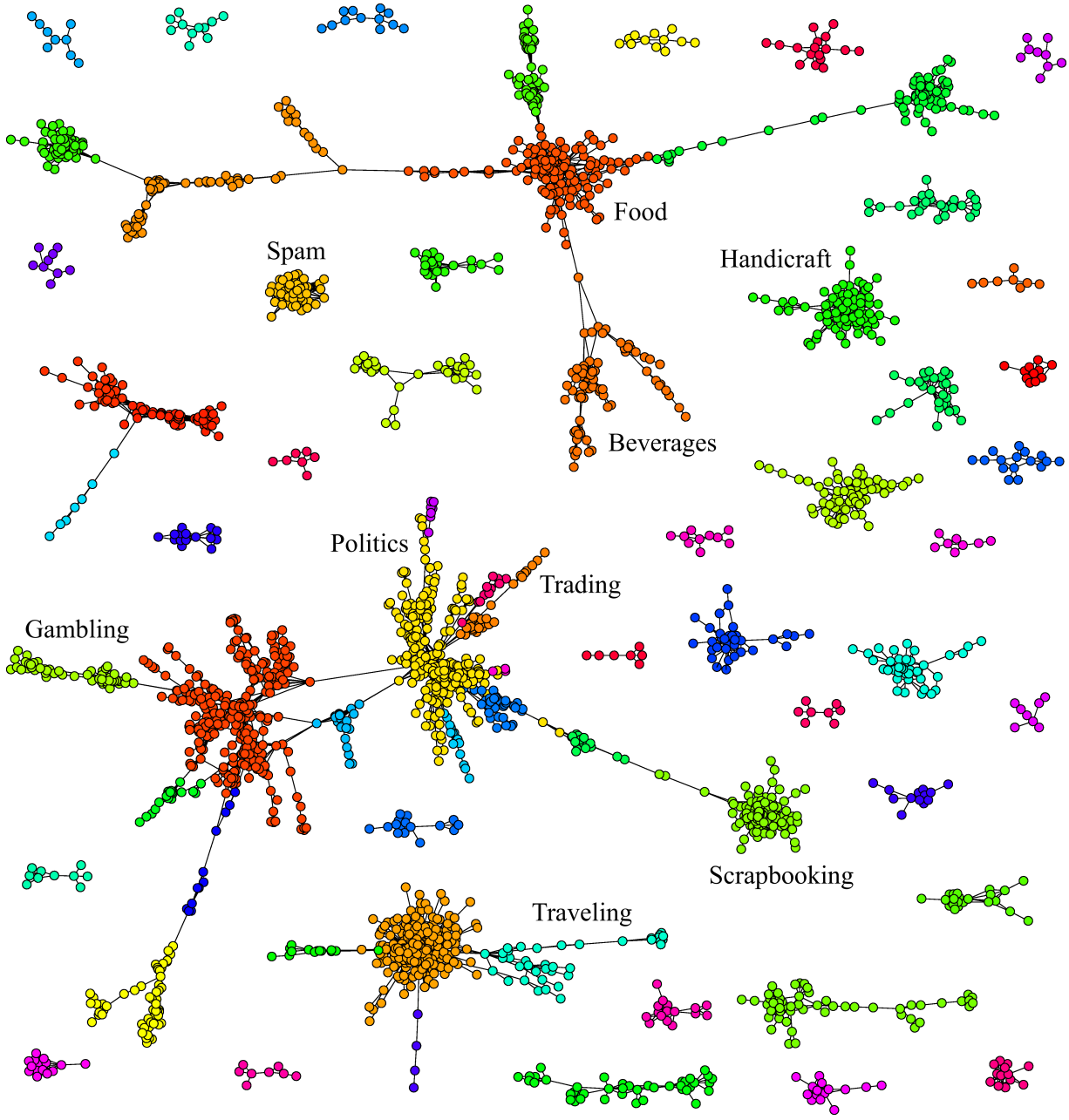


Figure 2: Content-based visualization of Swedish blogs. Blog categories (color-coded) are derived as network communities. Some example categories are labeled. For sake of clarity, only edges with weights larger than or equal to 0.05 are shown.

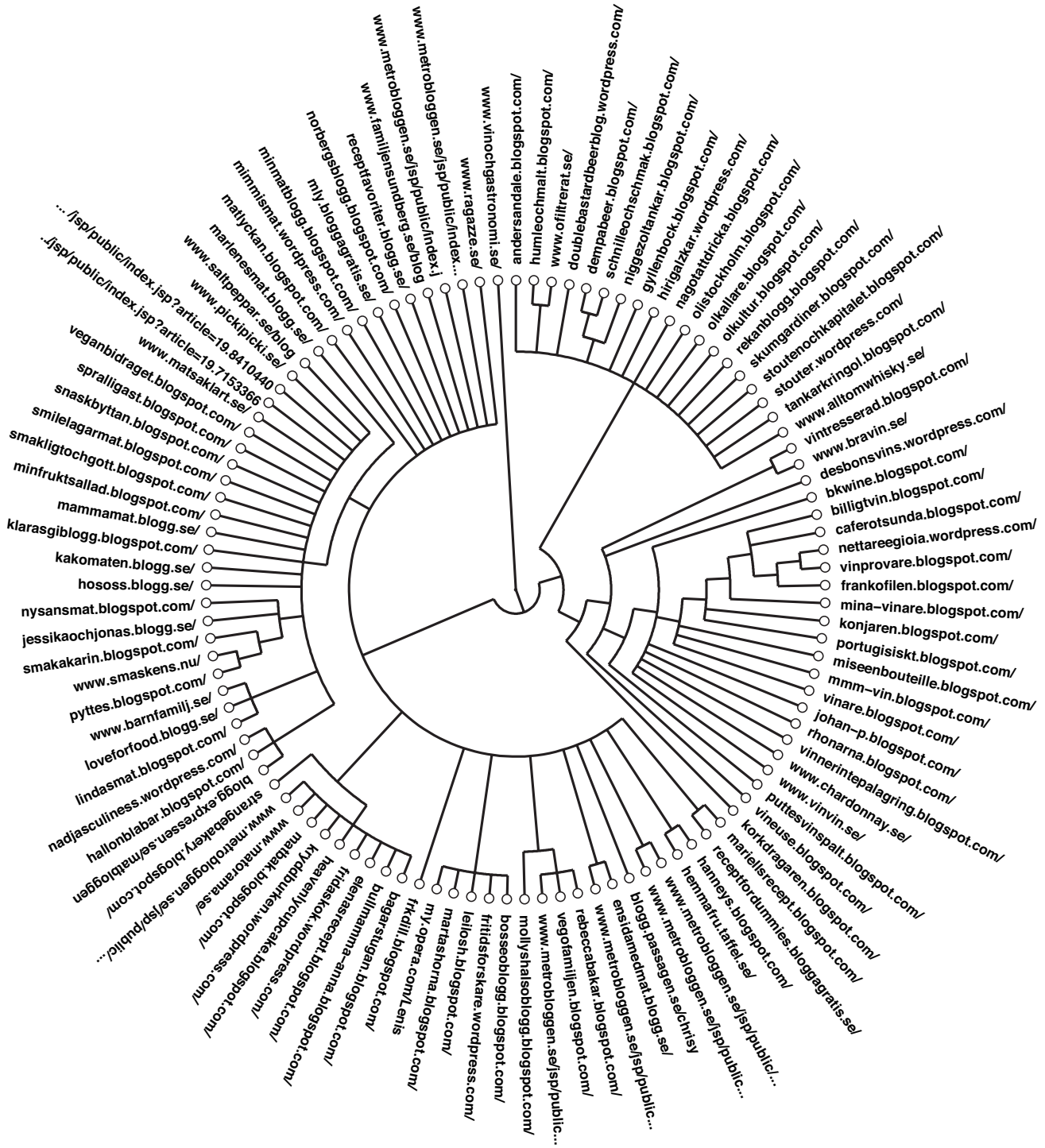


Figure 3: Consensus dendrogram of the food-and-beverages cluster, where blogs are organized in separate vine, beer and food clusters. Leaf nodes are labeled with corresponding blog URLs.

System-Wide Effectiveness of Active Learning in Collaborative Filtering

Mehdi Elahi, Francesco Ricci, Valdemaras Repsys,
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
mehdi.elahi@stud-inf.unibz.it, fricci@unibz.it, valdemaras@gmail.com

Abstract

The accuracy of a collaborative-filtering system largely depends on two factors: the quality of the recommendation algorithm and the number and quality of the available product ratings. In general, the more ratings are elicited from the users, the more effective the recommendations are. However, not all the ratings are equally useful and specific techniques, which are defined as rating elicitation strategies, can be used to selectively choosing the items to be presented to the user for rating. In this paper we consider several rating elicitation strategies and we evaluate their system utility, i.e., how the overall behavior of the system changes when new ratings are added. We discuss the pros and cons of different strategies with respect to several metrics (MAE, precision, NDCG and coverage). It is shown that different strategies can improve different aspects of the recommendation quality.

1 Introduction

Choosing the right product to consume or purchase is nowadays a challenging problem due to the growing variety of eCommerce services and the informational globalization. Recommender Systems (RSs) aim at addressing this problem providing personalized suggestions for digital content, products or services, that better match the user's needs and constraints than the mainstream products [Resnick and Varian, 1997] [Ricci *et al.*, 2011b].

In this paper we are concerned with collaborative filtering (CF) RSs [Koren, 2008] [Li *et al.*, 2008]. A CF system uses ratings for items provided by a population of users to predict, for a target user, what are the items with the highest ratings that he has not considered yet and recommend them to the user. The CF rating prediction accuracy does depend on the characteristics of the prediction algorithm, but also on the ratings known by the system. The more (informative) ratings are available the higher the recommendation accuracy is. Therefore, it is important to keep acquiring from the users new and useful ratings in order to maintain or improve the quality of the recommendations. In this work we concentrate on this aspect: understanding the behavior of several ratings acquisition strategies, such as "provide your ratings for these top ten movies". We aim at enlarging the set of available data in the optimal way for the whole system performance by asking the most useful ratings to the right users.

We created a software which simulates the real process of rating elicitation in a community of users (Movielens and Netflix), the consequent rating database growth starting from a relatively small one (cold-start), and the system adaptation (retraining) to the new set of data.

In this paper we define and test some "pure" strategies, i.e., implementing a single heuristic, but also strategies that we call "partially randomized", which, in addition to asking to the (simulated) users to rate the items selected by a "pure" strategy, they ask to rate some randomly selected items as well. Randomized strategies can introduce diversity in the item list presented to the user But, more importantly, they have been introduced to cope with the non monotonic behavior of the system effectiveness that we observed during the simulation of certain "pure" strategies. In fact, we have discovered (as hypothesized by [Rashid *et al.*, 2002]) that certain strategies, for instance, requesting to rate the items with the largest predicted ratings, may generate a system-wide bias, and ultimately the addition of the ratings proposed by these strategies can increase, rather than decrease, the system error.

In these simulations we used a state of the art Matrix Factorization rating prediction algorithm [Koren and Bell, 2011] [Timely Development, 2008]. Hence the results here presented can provide useful guidelines for managing real RSs that nowadays largely rely on that technique.

Rating elicitation has been also tackled in a few previous works [Sean M. McNee and Riedl, 2003; Rashid *et al.*, 2002; Carenini *et al.*, 2003; Jin and Si, 2004; Harpale and Yang, 2008] but these papers focused on a different problem, namely the benefit of the rating elicitation process for a single user, e.g., in the sign up stage [Rashid *et al.*, 2002]. Conversely, we consider the impact of an elicitation strategy on the system-wide behavior, e.g., the overall prediction accuracy (more details are provided in section 6). In general, rating elicitation has been ignored by the mainstream RSs research. A possible explanation is because of the erroneous assumption that a RS cannot control what items the users will rate. Actually this is not true, surely RSs user interfaces can be designed so that users navigating through the existing items can rate them if they wish. But new ratings can also be acquired by explicitly asking users. In fact, it is common

practice for RSs to ask the users to rate the recommended items: mixing recommendation with users' preferences elicitation. We will show that this approach has a potentially dangerous impact on the system effectiveness, hence a careful selection of the elicitation strategy is in order.

The main contribution of our research is the introduction and empirical evaluation of a set of rating elicitation strategies for collaborative filtering with respect to their system-wide utility. Some of these strategies are new and some come from the literature and the common practice. Another important contribution of this paper is due to the fact that we measured the effect of each strategy on several RSs evaluation measures showing that the best strategy depends on the evaluation measure. Previous works focussed only on the rating prediction accuracy (Mean Absolute Error), and on the number of acquired ratings. We analyze those aspects, but in addition we consider the recommendation precision, the coverage and the goodness of the recommendations' ranking, measured with normalized discounted cumulative gain (NDCG). These measures are more interesting and useful for determining the true value of the recommendations for the user.

Moreover, in our work we explore another new aspect, i.e., the performance of the elicitation strategies taking into account the size of the rating database and we show that different strategies can improve different aspects of the recommendation quality at different stages of the rating database development. In fact, we show that in some stages an elicitation strategy may induce a bias on the system and ultimately a decrease of the recommendation effectiveness. In addition, previously conducted evaluations assume rather artificial conditions, i.e., that all the users and the items have some ratings since the beginning of the evaluation process. In other words, they did not faced the new-item and the new-user problem). We instead generate initial conditions for the rating data set in a pure random way, hence, in our experiments, new users and new items are present as it happens in real conditions.

In conclusion in this paper we provide a realistic, comprehensive evaluation of several applicable rating elicitation strategies, providing guidelines and conclusions that would help their exploitations in real RSs. The rest of the paper is structured as follow. In section 2 we introduce the rating elicitation strategies that we have analyzed, and in section 3 we present the simulation procedure that we designed to evaluate their effects. The results of our experiments are shown in section 4. In section 6 we review some related researches, and finally in section 7 we summarize the results of this research and we outline some future work.

2 Elicitation Strategies

A rating dataset R is a $n \times m$ matrix of real values (ratings) with possible null entries. The variable r_{ui}, denotes the entry of the matrix in position (u, i), and contains the rating assigned by user u to item i. r_{ui} could store a null value representing the fact that the system does not know the opinion of the user on that item. In the Movielens and Netflix datasets the rating values are integers between 1 and 5 included. A rating elicitation strategy S is a function $S(u, N, K, U_u) = L$ which returns a list of items $L = \{i_1, \ldots, i_M\}$ whose ratings should be asked to the user u, where N is the maximum number of ratings to be elicited, K is the dataset of known ratings, i.e., the ratings (of all the users) that have been already acquired by the RS. K is also a $n \times m$ matrix containing entries with real or null values. The not null entries represent the knowledge of the system at a certain point of the RS evolution. Finally, U_u is the set of items whose ratings have not yet requested to u, hence potentially interesting. Hence one must enforce that $L \subset U_u$ and an elicitation strategy will not ask to a user to rate two times the same item; hence the items in L, which are returned by S, must be removed from U_u.

Every strategy analyzes the dataset of known ratings K and assigns a score to the items in U_u. Then the N items with the highest score are returned, if the strategy can score N different items, otherwise a smaller number of items is returned. It is important to note that the user may have not experienced the items whose ratings are requested; in this case the system will not increase the number of known ratings. Some strategies may collect more ratings, some strategies may be better in collecting useful ratings. These two properties play a fundamental role in a rating elicitation strategy.

2.1 Individual Strategies

We considered two types of strategies: *pure* and *partially randomized*. The first ones implement a unique heuristic, whereas in the second type of strategies a pure one is hybridized by adding some random rating requests that are still unclear to the system. As we mentioned in the introduction these strategies add some diversity to the system requests and, as we will show later, can cope with an observed problem of pure strategies: they may in some cases increase the system error.

The pure strategies that we considered are:

- *Popularity:* for all the users the score for item i is the number of not null ratings for i contained in K. These are the known ratings for the item i. More popular items are more likely to be known by the user, and hence it is more likely that a request for such a rating will really increase the size of the rating database.
- *Binary Prediction:* the matrix K is transformed in a matrix B with the same number of rows and columns, by mapping null entries in K to 0, and not null entries to 1. A factor model is built using the matrix B as training data, and then a prediction for each entry in B is computed. Finally, the score for the item i in U_u is the predicted value for the entry in position (u, i) in B. This strategy tries to predict what items the user has experienced, in order to maximize the probability that the user know the requested rating (similarly to the popularity strategy).
- *Highest Predicted:* a prediction is computed for all the items in U_u and the scores are set equal to these predicted values. The idea is that the best recommendations could also be more likely to have been experienced by the user and their ratings could also reveal more information on what the user likes. Moreover, this is the default strategy for RSs, i.e., enabling the user to rate the recommendations.

- *Lowest Predicted:* for all the items in U_u a prediction $\hat{r}_{ui}$ is computed. Then the score for i is $Maxr - \hat{r}_{ui}$, where $Maxr$ is the maximum rating value (e.g., 5). Lowest predicted items are likely to reveal what the user does not like, but are likely to collect a few ratings, since the user is unlikely to have experienced all the items that he does not like.
- *Highest and Lowest Predicted:* for all the items in U_u a prediction $\hat{r}_{ui}$ is computed. The score for an item is $|\frac{Maxr-Minr}{2} + Minr - \hat{r}_{ui}|$, where $Minr$ is the minimum rating value (e.g., 1). This strategy tries to ask ratings for items that the user may like and not like as well.
- *Random:* the score for an item is a random integer number from 1 to 10. This is just a baseline strategy, used for comparison.
- *Variance:* the score for the item i is equal to the variance of its ratings in the dataset K. This is a representative of the strategies that try to collect more useful ratings, assuming that the opinion of the user on items with more diverse ratings are more useful to the generation of correct recommendations.

2.2 Partially Randomized Strategies

In a partially randomized strategy we modify the list of items returned by a pure strategy introducing some random items. As we mentioned in the introduction, these strategies have been introduced to cope with some problems of the pure ones (see section 4). Precisely, the randomized version Ran of the strategy S with randomness $p \in [0, 1]$ is a function $Ran(S(u, N, K, U_u), p)$ returning a new list of items L' computed as follow:

1. $L = S(u, N, K, U_u)$ is obtained
2. if L is an empty list, i.e., the strategy S for some reason could not generate the elicitation list, then L' is computed by taking N random items from U_u.
3. if $|L| < N$, $L' = L \cup \{i_1, \ldots, i_{N-|L|}\}$, where i_j is a random item in U_u.
4. if $|L| = N$, $L' = \{l_1, \ldots, l_M, i_{M+1}, \ldots, i_N\}$, where l_j is a random item in L, $M = \lceil N * (1 - p) \rceil$, and i_j is a random item in U_u.

We note that if S is the highest predicted strategy, there are cases where no rating predictions can be computed by the RS for the user u, and hence S is not able to sort the items to request. This happens for instance when u is a new user and none of his ratings is known. In this case the randomized version of this strategy generates purely random items for the user to rate.

3 Evaluation Approach

In order to study the effect of the considered elicitation strategies we set up a simulation procedure. The goal was to simulate the evolution of a RS's performance exploiting these strategies. In order to run such simulations we partition all the available (not null) ratings in R into three different matrices with the same number of rows and columns as R:

- K: contains the ratings that are considered to be known by the system at a certain point in time.
- X: contains the ratings that are considered to be known by the users but not by the system. These ratings are incrementally elicited, i.e., they are transferred into K if the system asks them to the (simulated) users.
- T: contains the ratings that are never elicited and are used only to test the strategy, i.e., to estimate the evaluation measures (defined later).

We also note that U_u is the set of items whose ratings, at a certain point in time, are worth acquiring because "unclear" to the system. That means that k_{ui} has a null value and the system has not yet asked it to u. That request may end up with a new (not null) rating k_{ui} inserted in K, if the user has experienced the item i, i.e., if x_{ui} is not null, or in a no action, if x_{ui} has a null value in the matrix X. The system, in any case will remove the item i from U_u, to not ask twice the same rating.

We will discuss later how the simulation is initialized, i.e., how the matrices K, X and T are built from the full rating dataset R. In any case, these three matrices partition the full dataset R; if r_{ui} has a not null value then either k_{ui} or x_{ui} or t_{ui} has that value, and only one of them is not null. The testing of a strategy S proceeds in the following way:

1. The not null ratings in R are partitioned into the three matrices K, X, T.
2. MAE, Precision and NDCG are measured on T, training the rating prediction model on K.
3. For each user u:
   (a) Only the first time that this step is executed, U_u, the unclear set of user u is initialized to all the items i with a null value k_{ui} in K.
   (b) Using strategy S (pure or randomized) a set of items $L = S(u, N, K, U_u)$ is computed.
   (c) The set L_e, containing only the items in L that have a not null rating in X, is created.
   (d) Assign to the corresponding entries in K the ratings for the items in L_e as found in X.
   (e) Remove the items in L from U_u: $U_u = U_u \setminus L$.
4. MAE, Precision and NDCG are measured on T, and the prediction model is re-trained on the new set of ratings contained in K.
5. Repeat steps 3-4 (Iteration) for I times.

The MovieLens [Miller *et al.*, 2003] and Netflix rating databases were used for our experiments. Movielens consists of 100,000 ratings from 943 users on 1682 movies. From the full Netflix data set, which contains 1,000,000 ratings, we extracted the first 100,000 ratings entered into the system. They come from 1491 users on 2380 items, so this sample of Netflix data is 2.24 times sparser than Movielens data.

We also performed some experiments with the larger versions of of both Movielens and Netflix datasets (1,000,000 ratings) and obtained very similar results. However, using the full set of Netflix data required much longer times to perform our experiments since we train and test a rating prediction

model at each iteration: every time we add to K new ratings elicited from the simulated users. After having observed a very similar performance on some initial experiments we focussed on the smaller data sets to be able to run more experiments.

When deciding how to split the available data into the three matrices K, X and T an obvious choice is to respect the time evolution of the dataset, i.e., to insert in K the first ratings acquired by the system, then to use a second temporal segment to populate X and finally use the remaining ratings for T. Actually, it is not significant to test the performance of the proposed strategies for a *particular* evolution of the rating dataset. Since we want to study the evolution of a rating data set under the application of a new strategy we cannot test it only against the temporal distribution of the data that was generated by a particular (unknown) previously used elicitation strategy. Hence we followed the approach used in [Harpale and Yang, 2008] to random split the rating data, but we generated several random splits of the ratings into K, X and T. Besides, in this way we could generate ratings configurations where there are users and items that have no (not null) ratings initially in the known dataset K. We believe this approach provided us with a very realistic and hard experimental setup, letting us to address the new user and new item problems [Ricci *et al.*, 2011a].

Finally, we observe that for both data sets the experiments were conducted partitioning (randomly) the 100,000 not null ratings of R in the following way: 2000 in K (i.e., very limited knowledge at the beginning), 68,000 in X, and 30,000 in T. Moreover, $|L| = 10$, which means that the system at each iteration asks to a user his opinion on 10 items. The number of iterations was $I = 170$, and the number of factors in the SVD prediction model was set to 16. All the experiments were performed 5 times and results presented in the following section are obtained as averages of these five repetitions.

We considered four evaluation measures: mean absolute error (MAE), precision, coverage and normalized discounted cumulative gain (NDCG) [Herlocker *et al.*, 2004; Manning, 2008]. For computing precision we extracted, for each user, the top 10 recommended items (whose ratings appear in T) and considered as relevant the items with true ratings equal to 4 or 5. The coverage of a recommender system is measured as the proportion of the full set of items over which the system can form predictions [Herlocker *et al.*, 2004].

Discounted cumulative gain (DCG) is a measure originally used to evaluate effectiveness of information retrieval systems [Järvelin and Kekäläinen, 2002], but also collaborative filtering RSs [Weimer *et al.*, 2008] [Liu and Yang, 2008]. In RSs the relevance is measured by the rating value of the item in the predicted recommendation list. Assume that the recommendations for u are sorted according to the predicted rating values, then DCG_u is defined as:

$$DCG_u = \sum_{i=1}^{N} \frac{r_u^i}{log_2(i+1)} \quad (1)$$

where r_u^i is the true rating (as found in T) for the item ranked in position i for user u, and N is the length of the recommendation list. Normalized discounted cumulative gain for user u is then calculated in the following way:

$$NDCG_u = \frac{DCG_u}{IDCG_u} \quad (2)$$

where $IDCG_u$ stands for the maximum possible value of DCG_u, that could be obtained if the recommended items are ordered by decreasing value of their true ratings. We measured also the overall average discounted cumulative gain $NDCG$ by averaging $NDCG_u$ over the full population of users.

4 Evaluation of Pure Strategies

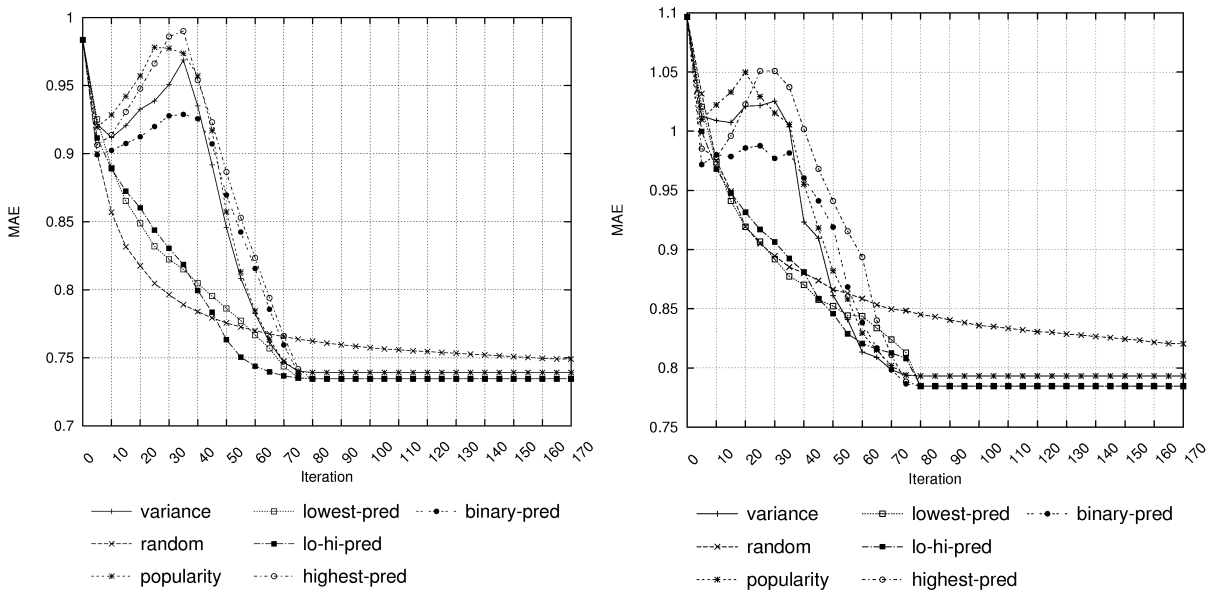
4.1 MAE

The MAE computed on the test matrix T at successive iterations of the application of the elicitation strategies is depicted in Figure 1. First of all we must observe that the behavior of the considered strategies in the two data sets are very similar. Moreover, there are two clearly distinct groups:

1. Monotone error decreasing strategies: lowest-highest predicted, lowest predicted and random.
2. Non-monotone error decreasing strategies: binary predicted, highest predicted, popularity, variance.

Strategies of the first group show an overall better performance (MAE) for all the duration of the test except at the beginning and the end. During the iterations 1-5 the best performing strategy is binary-predicted, the second best being highest predicted, both being non-monotone. During iterations 6-45 random strategy has the lowest MAE value in the Movielens data set, and it is overtaken by the lowest-highest-predicted strategy at iteration 46. This is not observed in the Netflix data set. At the iteration 80 the MAE obtained using the variance, popularity and all the prediction-based strategies stop changing: as K reaches the largest possible size for those strategies. The MAE obtained using the random strategy keeps decreasing until all the ratings in X are moved to K. It is important to note that the prediction based strategies (e.g., highest predicted) cannot elicit ratings for which the prediction can not be made, i.e., for all those movies and users that don't have (not null) ratings in K. This is reflected by the behavior of the coverage The coverage graph is not shown here for lack of space, but we can summarize these results noting that the coverage of the prediction-based strategies is stable, with value 0.74 (Movielens) throughout the experiment, because the set of users or items with not null ratings in K is not increasing. The system coverage produced by the random strategy is slowly increasing and reaches the full coverage, because ratings for new users and new items are randomly added to K. The coverage obtained by the variance and popularity strategies increases and stabilizes to 0.84 on iteration 10, but it does not reach the full coverage. This is because those strategies are able to elicit ratings from new users, but are not able to elicit ratings for new items. Very similar results are observed in the Netflix data.

The non-monotone strategies' behaviors can be divided into three stages: they decrease MAE at the beginning (approximately iterations 1-5), then they slowly increase it, when MAE reaches a peek (approximately iterations 6-35), and



(a) Movielens (b) Netflix

Figure 1: MAE of the pure strategies

Table 1: The percentage of the ratings elicited by the *Highest Predicted* strategy at different iterations

	Percentage of elicited ratings (r)				
Iterations	r=1	r=2	r=3	r=4	r=5
1 to 5	2.06%	4.48%	16.98%	36.56%	39.90%
35 to 40	6.01%	13.04%	29.33%	34.06%	17.53%

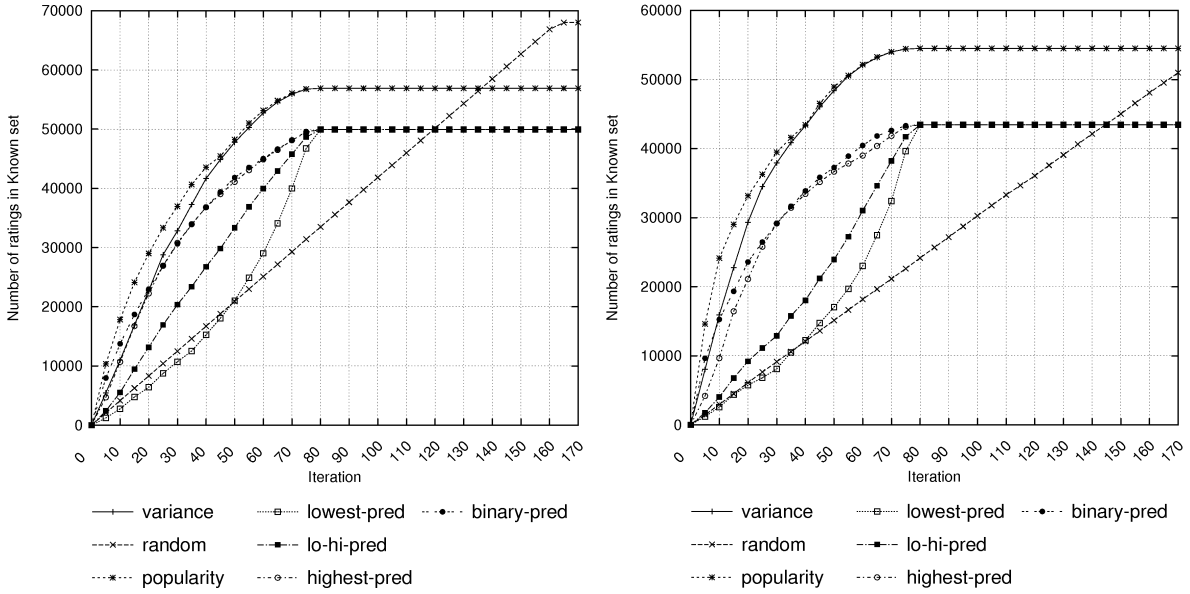
then they slowly decrease MAE till the end of the experiment (approximately iterations 36-80). The explanation of such a behavior is that the strategies belonging to this second group have a selection bias which can negatively affects MAE. For instance, the highest predicted strategy at the first iterations elicits many more high ratings compared to those elicited later on (Table 1). As a result it ends up with adding more high (than low) ratings to the known matrix (K), which biases the rating prediction.

In fact, low rated movies are selected for elicitation by the highest predicted strategy in two cases: 1) when a low rated item is predicted to have a high rating 2) when all the highest predicted ratings have been already elicited or marked as "not available" (they are not present in X and removed from U_u). Looking into the data we discovered that at iteration 36 the highest-predicted strategy has already elicited most of the highest ratings. Then the next ratings that are elicited are actually average or low ratings, which reduces the bias in K and also the prediction error. The random and lowest-highest predicted strategies do not introduce such a bias, and this results in a constant decrease of MAE.

4.2 Number of Acquired Ratings

It is important to measure how many ratings are added by the considered strategies. In fact, certain strategies can acquire more ratings by better guessing what items the user actually experienced. This occurs in our simulation if a strategy asks to the simulated user more ratings that are present in the matrix X. Conversely, a strategy may not be able to acquire many ratings but those actually acquired are very useful to generate better recommendations.

Figure 2 shows the size of the system known ratings in K, as the strategies elicit new ratings from the simulated users. It is worth noting, even in this case, the strong similarity of the behavior of the considered strategies in the two data sets. The only strategy that differs substantially in the two data sets is random. This is clearly dependent on the larger number of users and items that are present in our sample of the Netflix data. In fact, here there are 100,000 ratings as in the Movielens data set but the sparsity is higher: there are only 2.8% of the possible ratings (1491*2380) vs. 6.3% of the possible ratings (943*1682) contained in the Movielens data set. This larger sparsity makes more difficult for a pure random selection to pick up items that are known to the user. In general this is a major limitation of any random strategy, i.e., the very slow rate of addition of new ratings. Hence for relatively small problems (items and users) the random strategy may be applicable, but for larger ones this is impractical. In fact, observing Figure 2, one can see that in the Movielens simulations after 70 iterations, which means 70*10*943=660.100 ratings' requests (iterations * number-of-rating-requests * users) the system has acquired on average only 28.000 new ratings (30.000 is the new size but 2000 were already present at the beginning of the process). This means that only one out of 23 random rating requests could be provided by a user. In the Netflix data set this is even worse. It is interesting to note that even the popularity strategy has a



(a) Movielens (b) Netflix

Figure 2: Number of elicited ratings

poor performance in term of number of elicited ratings; it can elicit the first 28.000 ratings at a speed equal to one rating for each 6.7 rating requests. We also observe that according to our results, quite surprisingly the larger sparsity of the Netflix sample has produced a substantially different impact only on the random strategy.

Figure 3 illustrates a related aspect, i.e., how much the acquired ratings are useful for the effectiveness of the system, i.e., how the same number of ratings, acquired by different strategies can reduce MAE. It is clear that in the first stage of the process, i.e., when a small number of ratings are present in the known set K, the random and lowest-predicted strategies collect the more useful ratings for reducing MAE. Successively, the lowest-highest-predicted strategy bring more useful ratings. This is an interesting result, showing that the items with the lowest predicted ratings and random items are bringing more useful information even if it is difficult to acquire these ratings. It is also clear that certain strategies are not able to acquire all the ratings in X. For instance lowest-highest-predicted, lowest-predicted and highest-predicted stop acquiring new ratings when they have collected 50.000 ratings (Movielens). This is due to the fact that these strategies need rating predictions that in some cases (e.g., for new users) cannot be made by the system.

4.3 NDCG

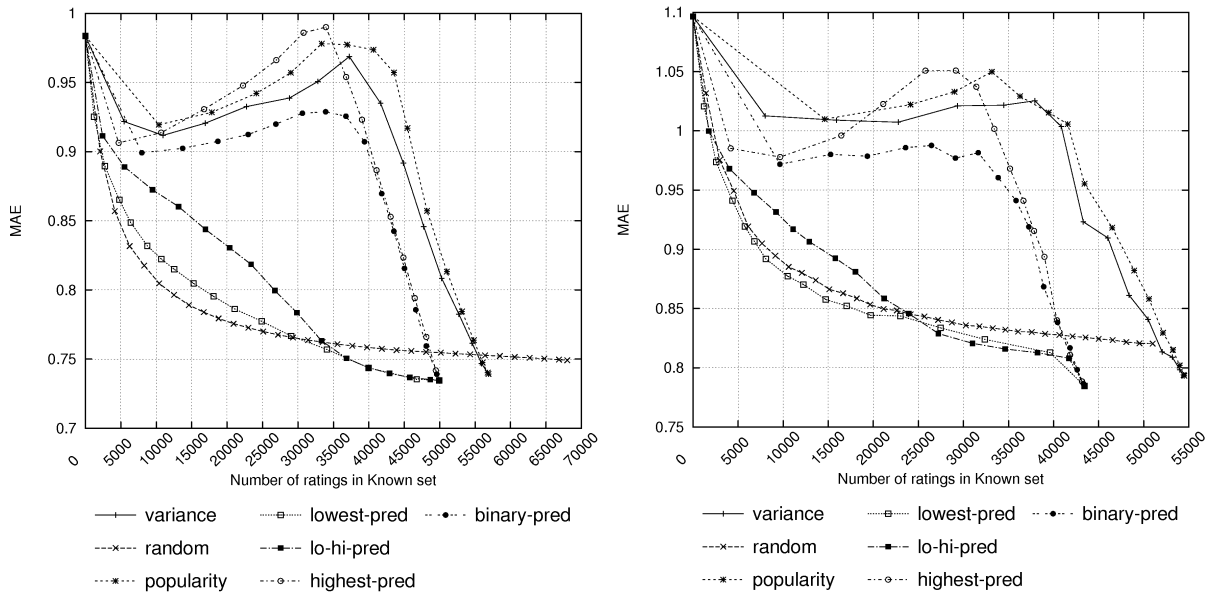
We measured NDCG on the first top 10 recommendations with not null values in T (for each user) (Figure 4). Note that sometime we computed NDCG on a smaller set, i.e., only on the ratings in the user test set whose value can be actually predicted.

Popularity is the best strategy at the beginning of the experiment. But at iteration 4 (Movielens) and 15 (Netflix) the random strategy passes the popularity strategy and then remains the best one. Excluding the random strategy, popularity and variance are the best in both data sets. Lowest predicted is by far the worst, and this is quite surprising considering how effective it is in reducing MAE. Moreover, another striking difference from the MAE results, is that all the strategies improve NDCG monotonically. Analyzing the experiment data we discovered that lowest predicted is not effective for NDCG because it is eliciting more ratings for the lowest ranked items and this is useless to predict the ranking of the top items. It is also important to note that here the random strategy is by far the best. This is again different from the MAE behavior.

4.4 Precision

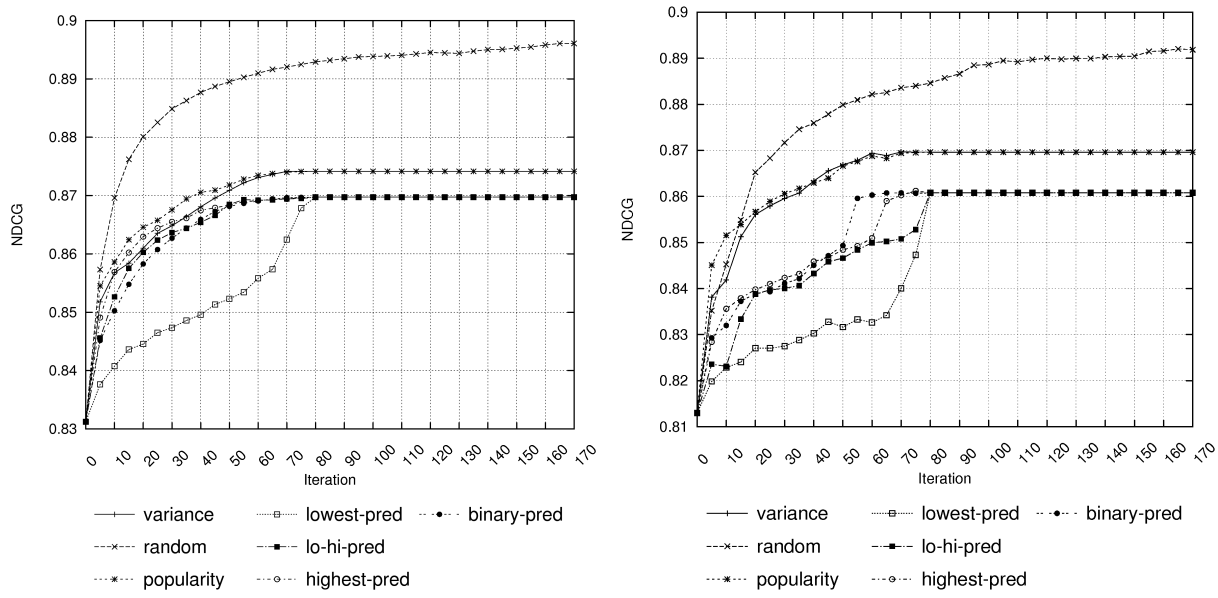
In the rest of this paper we focus on the Movielens data. In fact, as we have already observed apropos of MAE and NDCG, very similar results were observed for the system precision using the Netflix data, so for lack of space we omit them.

Precision, as it was described in section 3, measures the proportion of items rated 4 and 5 that are found in the recommendation list. Figure 5 depicts the evolution of the system precision when the proposed strategies are applied. Here, highest predicted is the best performing strategy for the largest part of the test. Starting from iteration 50 it is equally good as the binary predicted and the lowest-highest-predicted strategies. It is also interesting to note that all the strategies monotonically increase the precision. Moreover, the random strategy, differently from NDCG, does not perform so well, if compared with the highest predicted strategy. This is again related to the fact that the random strategy increases substantially the coverage and this produces a lower overall precision because precision is significantly smaller for new users.



(a) Movielens (b) Netflix

Figure 3: MAE vs number of ratings elicited



(a) Movielens (b) Netflix

Figure 4: NDCG of the pure strategies

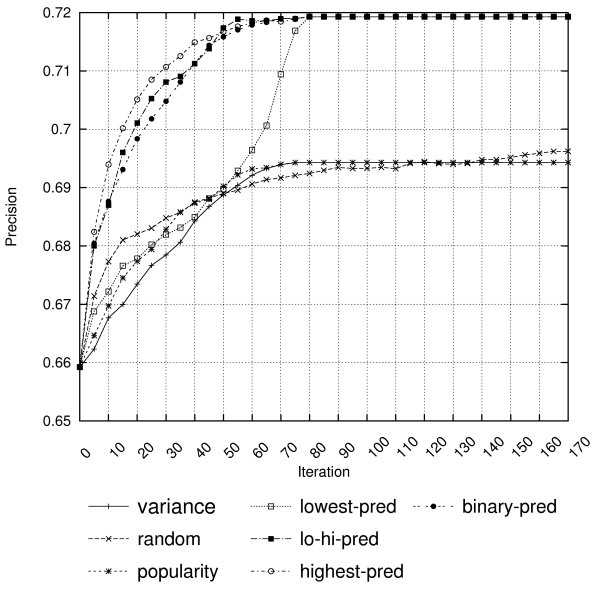


Figure 5: Precision of pure strategies (Movielens)

In conclusion from these experiments one can conclude that *there is no single best strategy*, among those that we evaluated, that dominates the others for all the evaluation measures. The *random* strategy is the best for NDCG, whereas for MAE and precision we would suggest using *lo-high predicted*, performing quite well for both measures.

5 Evaluation of the Partially Randomized Strategies

Among the pure strategies only the random one is able to elicit ratings for items that have not been evaluated by any user already represented in K. Partially randomized strategies address this problem by asking new users to rate random items (see Section 2). In this section we have used partially randomized strategies where $p = 0.2$, i.e., at least 2 of the 10 rating values elicited from the simulated users are chosen at random.

Figure 6 depicts the system MAE evolution during the experimental process. Now all the curves are monotone, i.e., it is sufficient to add some randomly selected ratings to the elicitation lists to reduce the bias of the pure, prediction-based, strategies. The best performing partially randomized strategies, with respect to MAE, are, at the beginning of the process, the partially randomized binary-predicted, and subsequently the low-high-predicted (similarly to the pure strategies case).

For lack of space, the behaviors of the system precision and NDCG are not shown here, we just describe them. These two measures behave similarly. The partially randomized highest predicted strategy has the best results for the largest part of the test (for both measures). Interestingly, the worst strategy is the lowest-predicted, i.e., it seems that for improving the recommender precision it does not pay off to ask the user his opinion on items that the system believes are irrelevant (which is not the case if the goal is to improve MAE). It is important to note that the strategies that show good performance at the beginning (partially randomized highest and binary predicted strategies) are those tuned for finding items that a user may know and be able to provide a rating for. Therefore, they are very effective in the beginning when there are many users with very little items in the known dataset K.

6 Related Work

The rating elicitation problem can be considered as an active learning problem. Active learning aims at actively acquiring training data to improve the output of the recommender system [Rubens *et al.*, 2011]. [Rashid *et al.*, 2002] proposes six techniques that collaborative filtering recommender systems can use to learn about new users in the sign up process. They considered: pure entropy, i.e., items with the largest entropy are preferred; random selection; popularity, i.e., the items that have the largest number of ratings; items with the largest $log(popularity) * entropy$, i.e., items that are both popular and have diverse rating values; and finally in "item-item personalized" the items are proposed randomly until one rating is acquired, then a recommender is used to predict the items that the user is likely to have seen.

They studied the behavior of an item-based CF only with respect to MAE, and designed an offline experimental study that simulates the sign up process. Each strategy was used to select a certain number of items (30, 45, 60 and 90) for a user without knowing if they were experienced by that user, i.e., their ratings were present in the dataset or not. Then MAE was measured for the user on the remaining ratings while using the elicited ratings, and the ratings of other training users, as training data. The process was repeated and averaged for all the test users. In this scenario the $log(popularity) * entropy$ strategy is the best. However, popularity and item-item personalized strategies outperformed $log(popularity)*entropy$ with respect to the user effort. User effort was computed as a fraction of items elicited over total number of items presented. It is worth noting that these results are not comparable with ours as they measured how a varying set of ratings elicited from one user are useful in predicting the ratings of the same user. In our experiments we simulate the simultaneous acquisition of ratings from all the users, by asking in turn to each user 10 ratings, and repeating this process several times. This simulates the long term usage of a recommender system where users come again and again to get new recommendations and the rating provided by a user is exploited to generate better recommendations to others (system performance).

[Harpale and Yang, 2008] remarks that the Bayesian active learning approach introduced in [Jin and Si, 2004] makes an implicit and unrealistic assumption that a user can provide rating for any queried item. Hence, they propose a revised Bayesian selection approach, which does not make such an assumption, and introduces an estimation of the probability that a user has consumed an item in the past and is able to provide a rating.

Their results show that the personalized Bayesian selection outperforms Bayesian selection and the random strategy

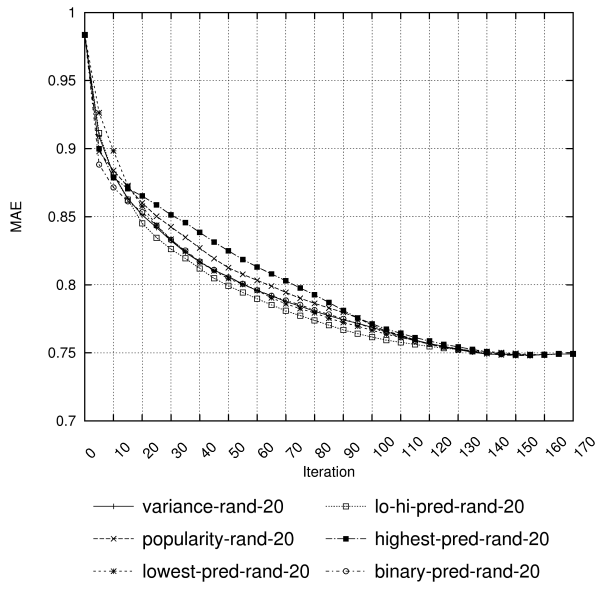


Figure 6: MAE of partially randomized strategies (Movielens)

with respect to MAE. Their simulation setting is similar to that used in [Rashid *et al.*, 2002], hence for the same reason their results are not directly comparable with ours. There are other important differences between their experiment and ours: their strategies elicit only one rating per request; they compare the proposed approach only with the random strategy; they do not consider the new user problem, since in their simulations all the users have 3 ratings at the beginning of the experiment, whereas in our experiments, there might be users that have no ratings at all in the initial stage of the experiment; they use a completely different rating prediction algorithm (Bayesian vs. Matrix Factorization). All these differences make the two set of experiments hard to compare. Moreover, their simulations starts from a known ratings data set that is larger than ours. In fact, the MAE they measured initially on Movielens is around 0.83, whereas in our experiments the MAE is almost 1.

In [Carenini *et al.*, 2003] again a user-focussed approach is considered. They propose a set of techniques to intelligently select ratings when the user is particularly motivated to provide such information. They present a conversational and collaborative interaction model which elicits ratings so that the benefit of doing that is clear to the user, thus increasing the motivation to provide a rating. Item-focused techniques that elicit ratings to improve the rating prediction for a specific item are proposed. Popularity, entropy and their combination are tested, as well as their item focused modifications. The item focused techniques are different from the classical ones in that popularity and entropy are not computed on the whole rating matrix, but only on the matrix of user's neighbors that have rated an item for which the prediction accuracy is aimed at being improved. Results have shown that item focused strategies are constantly better than unfocused ones. Also in this case, their results are complementary to our findings, since the elicitation process and the evaluation metrics are different.

7 Conclusions and Future Work

In this work we have addressed the problem of selecting items to present to the users for acquiring their ratings; that is also defined as the ratings elicitation problem. We have proposed and evaluated a set of ratings elicitation strategies. Some of them have been proposed in a previous work [Rashid *et al.*, 2002] (popularity, random, variance), and some, which we define as prediction-based strategies, are new: binary-prediction, highest-predicted, lowest-predicted, highest-lowest-predicted. Moreover, we have studied the behavior of other novel strategies, partially randomized, which insert some random ratings in the elicitation lists computed by the aforementioned strategies. We have evaluated these strategies for their system-wide effectiveness implementing a simulation loop that models the day-by-day process of rating elicitation and rating database growth. We have taken into account the limited knowledge of the users, i.e., the fact that the users will not know all the possible ratings. During the simulation we have measured several metrics at different phases of the rating database growth. The metrics include: MAE to measure the improvements in prediction accuracy, precision to measure the relevance of recommendations, normalized discounted cumulative gain (NDCG) to measure the quality of produced ranking and coverage to measure the proportion of items over which the system can form predictions.

The evaluation has shown that different strategies can improve different aspects of the recommendation quality and in different stages of the rating database development. Moreover, we have discovered that some pure strategies may incur in the risk of increasing the system MAE if they keep adding only ratings with a certain value, e.g., the largest ones, as for the highest-predicted strategy that is an approach often adopted in real RSs. In addition, prediction-based strategies neither address the problem of new users, nor of new items. Popularity and variance strategies are able to select items for new users, but can not select items that have no ratings.

Partially randomized strategies, have less problems because they add random items to rate that have no ratings at all. In this case the lowest-highest (highest) predicted is a good alternative if MAE (precision) is the target effectiveness measure. These strategies simulate that some items to rate are deliberately selected by the user, but one can also implement this in a pure elicitation strategy because of the benefits it produces.

This research opened a number of new problems that would definitely deserve some more study. First of all, it would be useful to repeat the same experiments using even more diverse datasets to study how the data distribution influences the strategies' behavior. In fact, we have already observed that the performance of some strategies (random) depends on the sparsity of the rating data. The MovieLens data and the Netflix sample that we used, still have a considerably low sparsity compared to other larger datasets. For example, if the data sparsity was higher, there would be only a very low

probability for random strategy to select an item that a user has consumed in the past and can provide a rating for. So the partially randomized strategies may perform worse in reality or it could be needed a different degree of randomness.

Furthermore, there remain many unexplored possibilities for combining strategies that use different approaches depending on the state of the target user. For instance, asking users to rate popular items when a user does not have any ratings yet and using another strategy at a latter stage. Moreover, it is important to consider the noise and inconsistency of the data when designing strategies that search for items that optimally combine the probability that a user has experienced them, and thus can really provide a rating value for them, with the usefulness of obtaining that information.

References

[Carenini *et al.*, 2003] Giuseppe Carenini, Jocelyin Smith, and David Poole. Towards more conversational and collaborative recommender systems. In *Proceedings of the 2003 International Conference on Intelligent User Interfaces, January 12-15, 2003, Miami, FL, USA*, pages 12–18, 2003.

[Harpale and Yang, 2008] Abhay S. Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98, New York, NY, USA, 2008. ACM.

[Herlocker *et al.*, 2004] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[Järvelin and Kekäläinen, 2002] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[Jin and Si, 2004] Rong Jin and Luo Si. A bayesian approach toward active learning for collaborative filtering. In *UAI '04, Proceedings of the 20th Conference in Uncertainty in Artificial Intelligence, July 7-11 2004, Banff, Canada*, pages 278–285, 2004.

[Koren and Bell, 2011] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer Verlag, 2011.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[Li *et al.*, 2008] Ying Li, Bing Liu, and Sunita Sarawagi, editors. *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008.

[Liu and Yang, 2008] Nathan N. Liu and Qiang Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90, New York, NY, USA, 2008. ACM.

[Manning, 2008] Christopher Manning. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.

[Miller *et al.*, 2003] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 263–266, New York, NY, USA, 2003. ACM.

[Rashid *et al.*, 2002] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. Mcnee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces, IUI 2002*, pages 127–134. ACM Press, 2002.

[Resnick and Varian, 1997] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[Ricci *et al.*, 2011a] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer Verlag, 2011.

[Ricci *et al.*, 2011b] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer, 2011.

[Rubens *et al.*, 2011] Neil Rubens, Dain Kaplan, and Masashi Sugiyama. Active learning in recommender systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul Kantor, editors, *Recommender Systems Handbook*, pages 735–767. Springer Verlag, 2011.

[Sean M. McNee and Riedl, 2003] Joseph A. Konstan Sean M. McNee, Shyong K. Lam and John Riedl. Interfaces for eliciting new user preferences in recommender systems. In *User Modeling 2003*, 2003.

[Timely Development, 2008] LLC Timely Development. Netflix prize, 2008. http://www.timelydevelopment.com/demos/NetflixPrize.aspx.

[Weimer *et al.*, 2008] Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Adaptive collaborative filtering. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 275–282, New York, NY, USA, 2008. ACM.

Social-behavior Transfer Learning for Recommendation Systems

Qian Xu, Evan Wei Xiang and Qiang Yang
Hong Kong University of Science and Technology
Hong Kong, China
fleurxq@cse.ust.hk, wxiang@cse.ust.hk, qyang@cse.ust.hk

Abstract

Online recommendation systems are becoming more and more popular with the development of web. Data sparseness is a major problem for collaborative filtering (CF) techniques in recommender systems, especially for new users and items. In this paper, we try to reduce the data sparseness in the collaborative filtering problem by involving Wikipedia as an auxiliary information source. In this paper, we attempt to improve the recommendation accuracy by extracting collaborative social behavior information embedded in Wikipedia co-editing history, and use this knowledge to help alleviate the data-sparsity problem in CF. In addition, we introduce a parallel computing algorithm to help scale up the transfer learning process. Our experimental results on two real world recommendation datasets show that the social co-editing knowledge in Wikipedia can be effectively transferred for CF problems.

1 Introduction

Machine learning and data mining technologies have already achieved significant success in many knowledge engineering areas including Web search, computational advertising, recommender systems, etc. A major challenge in machine learning is the data sparseness problem. In the domain of online recommender systems, we attempt to recommend information items (e.g., movies, TV, books, news, images,web pages, etc.) that are likely to be of interest to the user. However, in many CF problems, the number of user preference values is small and the data are sparse. This can be caused by the large item spaces or new services. In such cases, overfitting can easily happen when we learn a model, causing significant performance degradation. To address such data sparseness problem, some service providers turn to explicitly ask the newly registered customers to rate some selected items, such as some most sparsely rated jokes in a joke recommender system [Nathanson *et al.*, 2007]. However, this approach may degrade the customers' experience and satisfaction with the system. Alternative methods, making use of the implicit user feedbacks [Hu *et al.*, 2008], may achieve good results, However, such tracking data also suffer from the sparseness for newly launched systems.

More recently, researchers have introduced transfer learning techniques to solve the data sparseness problem [Li *et al.*, 2009]. Transfer learning methods aim at making use of the data from other information sources, e.g., some other recommender systems, to help with our prediction tasks in the target domain. However, they require user preferences expressed in auxiliary and target domains to be homogeneous such as a common rating scale of 1-5. In practice, such homogeneous data from the auxiliary domains are also very hard to obtain.

Fortunately, the development of Web 2.0 provides us an opportunity to access and share information on Internet. In order to cope with the huge needs of information from hundreds of millions of web users, Web 2.0 based social applications become more and more popular, such as sharing sites (e.g., Flickr, Picassa, YouTube), blogs (e.g., Blogger, WordPress, LiveJournal), social networks (e.g., MySpace, Facebook), and social tagging systems (e.g. Delicious, CiteULike, Digg). Among these Web 2.0 based social applications, Wikipedia is the most important and successful example.

Our intuition in this work is to transfer some knowledge from Wikipedia to help solve the data sparseness problem for collaborative filtering tasks. One way is to use the content or concept structure information in Wikipedia as the source data, where we can build the similarity measure based on the content similarity between two articles or or via the Wikipedia hyperlink graph. However, we have found that the content-based similarity does not fully reflect the closeness of users' common interests on two products, which is important in CF problems. Instead, we have found that the co-editing behaviors of Wikipedia articles can better reflect users' interests in an implicit manner. For example, if there is a group of users who favor both Action and Fantasy movies, it is more likely that a group of editors to Wikipedia's action-movie articles are also willing to edit articles on fantasy movies. Such a similarity measure is obtained from the perspective of the closeness of the *social behavior* between two groups of users.

Thus, in this paper, we explore how to directly transfer social behavior knowledge from the knowledge base of Wikipedia to help with the prediction tasks in our target recommendation systems. We introduce a new algorithm, known as 'Collaborative Editing based Domain and Item Transfer' (COEDIT), to alleviate the problem of data sparseness prob-

lems in collaborative filtering.

2 Related Works

2.1 Collaborative Filtering

Generally speaking, two common approaches are usually exploited for collaborative filtering. One is the memory-based approach and the other is the model-based approach. Memory-based approach conducts certain forms of nearest neighbor search in order to predict the rating for a particular user-item pair. Amongst memory-based methods, the user-based model has been widely used to estimate the unknown ratings of a target user based on the ratings by a set of neighboring users that tend to have similar rating behavior to the target user. A crucial component of the user-based model is the user-user similarity for determining the set of neighbors. Popular similarity measures include the Pearson Correlation Coefficient(PCC) [Resnick *et al.*, 1994][Herlocker *et al.*, 2002] and the vector similarity(VS) [Breese *et al.*, 1998]. An alternative form of the memory-based approach is the item-based model [Sarwar *et al.*, 2001][Linden *et al.*, 2003], which compares items based on the ratings they received. When using the memory-based models, it is often difficult to reliably compute user or item similarities especially when the rating matrix is sparse. To alleviate the sparseness problem, different techniques such as dimensionality reduction [Goldberg *et al.*, 2001] and data-smoothing methods[Xue *et al.*, 2005][Ma *et al.*, 2007], have been proposed to fill in the unknown ratings in the matrix.

The model-based approach for collaborative filtering uses the observed user-item ratings to train a compact model that explains the hidden pattern of the given data. Models in this category include matrix factorization [Rennie and Srebro, 2005; Paterek, 2007][Koren *et al.*, 2009], probabilistic mixture models [Hofmann, 2004; Jin *et al.*, 2003], Bayesian networks [Pennock *et al.*, 2000] and restricted boltzman machine [Salakhutdinov *et al.*, 2007]. Previous studies showed that model-based approach such as matrix factorization is one of the best-performing solutions and achieves significant improvement over memory-based methods.

2.2 Transfer Learning

In machine learning community, the problem of utilizing data sets from related but different domains to build model for a target application domain is known as transfer learning [Pan and Yang, 2009]. Although transfer learning algorithms have achieved great success traditional data mining such as classification, regression, there are limited works on transfer learning in the context of collaborative filtering. Bhaskar and Hofmann [Mehta and Hofmann, 2007] considered two systems with shared users and then used manifold alignment methods to jointly build memory-based models for the two systems. Li et al. [Li *et al.*, 2009] designed a regularization function for jointly factorize two rating matrices with neither common users nor common items. However, all these existing works require the participating systems to share their rating matrix completely, which may be infeasible in practice. Moreover, the existing solutions only work with two systems and are difficult to generalize to multi-systems setting. Singh et al. [Singh and Gordon, 2008] proposed a collective matrix factorization model, where they simultaneously factor several matrices, sharing parameters among factors when an entity participates in multiple relations.The newest work done by Pan et al. [Pan *et al.*, 2010] demonstrated how a joint matrix factorization model can be used to transfer knowledge between heterogeneous source and target domains. However, they only focus on the knowledge transfer among close recommendation systems, and our focus is whether we could transfer the social behaviors on Web.

2.3 Data Mining with Wikipedia

In recent years, understanding and utilizing online knowledge repositories to aid real world data mining tasks have become a hot research topic. An example is to use the Wikipedia for feature enrichment. Gabrilovich et al. [Gabrilovich and Markovitch, 2005; 2007b] tried to use the Open Directory Project (ODP) for feature enrichment in the text classification problem. They also showed that using Wikipedia as the external Web knowledge resource for feature enrichment outperforms using ODP [Gabrilovich and Markovitch, 2006]. Gabrilovich et al. [Gabrilovich and Markovitch, 2007a] tries to explicitly represent the meaning of texts in terms of a weighted vector of Wikipedia-based concepts. Their semantic analysis is explicit in the sense that they manipulate manifest concepts grounded in human cognition, rather than latent concepts used by LSA. In [Phan *et al.*, 2008], a general framework, building classifiers with hidden topics discovered from large-scale data collections, was proposed. The framework is mainly based on latent topic analysis models such as PLSA [Hofmann, 1999] and LDA [Blei *et al.*, 2001], and machine learning methods such as maximum entropy and SVMs. The underlying idea of such a framework is that for each classification task, a very large external data, called by "universal dataset", is collected and then a classification model on both a small set of labeled training data and a rich set of hidden topics is built. Currently, a limited approaches employ auxiliary knowledge such as online knowledge database for transfer learning. Wang et al. made an extension [Wang *et al.*, 2008] to the feature-based transfer learning models by incorporating a semantic kernel [Wang *et al.*, 2007] [Wang and Domeniconi, 2008] learned from Wikipedia. Different from these traditional data mining works on Wikipedia, which focus on transferring content or structure information, our work tries to extract some collaborative social behavior information from the Wikipedia co-editing history.

3 Problem Description

Suppose that we have a target recommendation S_{tar} that is associated with m_{tar} users and n_{tar} items, denoted by $\mathcal{U}_{tar}$ and $\mathcal{V}_{tar}$, respectively. In the target system, we observe a sparse rating matrix $\mathbf{X}_{tar} \in \mathbb{R}^{m_{tar} \times n_{tar}}$ with entries $X_{tar,ij}$. Let $\mathcal{R}_{tar} = \{(i, j, r) : r = X_{tar,ij}, \text{where } X_{tar,ij} \neq 0\}$ denote the set of observed ratings in the target system.

In order to predict the unobserved ratings in S_{tar}, a popular method is to employ low-rank factorization to recover missing entries in $\mathbf{X}_{tar}$. We model the users $\mathcal{U}_{tar}$ and the items $\mathcal{V}_{tar}$ by a user factor matrix $\mathbf{U}_{tar} \in \mathbb{R}^{k \times m_{tar}}$ and an item factor matrix $\mathbf{V}_{tar} \in \mathbb{R}^{k \times n_{tar}}$, where the i-th user and j-th item

are represented by $\mathbf{u}_i$ and $\mathbf{v}_j$, corresponding to the i-th and j-th column of $\mathbf{U}_{tar}$ and $\mathbf{V}_{tar}$, respectively. The goal is to approximate the rating matrix $\mathbf{X}_{tar}$, i.e., $\mathbf{X}_{tar} \approx \mathbf{U}_{tar}^T \mathbf{V}_{tar}$. Under the low-rank factorization model, the factor matrices $\mathbf{U}_{tar}$ and $\mathbf{V}_{tar}$ can be learned by minimizing the following loss function:

$$\mathcal{L}_{tar} = \sum_{(i,j)\in\mathcal{R}_{tar}} \left(\mathbf{u}_i^T \mathbf{v}_j - X_{tar,ij}\right) + \lambda(\| \mathbf{U}_{tar} \|_F^2 + \| \mathbf{V}_{tar} \|_F^2), \quad (1)$$

where λ controls the trade-off between the rating matrix approximation errors and model complexity reflected by the Frobenius norm of the factor matrices.

However, for new recommendation services, users may rate few items, causing the matrix $\mathbf{X}_{tar}$ to be extremely sparse. Directly optimizing $\mathcal{L}_{tar}$ will suffer from severe overfitting problem.

As stated earlier, the historical editing logs may carry huge amount of user preference information. For example, if a user favors some products, it is likely that he will edit the product article on Wikipedia. Thus we also model Wikipedia as an information system S_{wiki}, which is associated with a set of m_{wiki} users and n_{wiki} items denoted by $\mathcal{U}_{wiki}$ and $\mathcal{V}_{wiki}$. In Wikipedia, the user editing activities are recorded and represented with a sparse matrix $\mathbf{X}_{wiki} \in \mathbb{R}^{m_{wiki} \times n_{wiki}}$. Let $\mathcal{R}_{wiki} = \{(i,j,r) : r = X_{wiki,ij}, \text{where } X_{wiki,ij} \neq 0\}$ denote the set of editing activities in Wikipedia, i.e., if user i edits article j in Wikipedia, then $x_{wiki,ij} = 1$. In contrast with $\mathbf{X}_{tar}$, $\mathbf{X}_{wiki}$ is much larger so that it may contain more information, which help us explore hidden patterns of user social behavior. Our intuition is to "borrow" user social behavior information from S_{wiki} to learn S_{tar} better.

4 Transfer Learning via Co-Edit Knowledge

4.1 Matrix Factorization in COEDIT

In order to achieve knowledge transfer from S_{wiki} to S_{tar}, we need to solve two related problems. First, we need to align the products in S_{tar} to some articles in S_{wiki} through the Search API in Wikipedia. Once the item correspondence is established, a subsequent problem is to build a compact model to fuse the information in S_{wiki} and S_{tar} together for knowledge transfer.

We can consider Wikipedia and our target recommendation system as two information systems S_{wiki} and S_{tar} with similar formation. The system s is associated with m_s users and n_s items denoted by $\mathcal{U}_s$ and $\mathcal{V}_s$, $s \in \{S_{wiki}, S_{tar}\}$, respectively. For each system s, we observe a sparse rating matrix $\mathbf{X}_s \in \mathbb{R}^{m_s \times n_s}$ with entries $X_{s,ij}$. Let $\mathcal{R}_s = \{(i,j,r) : r = X_{s,ij}, \text{where } X_{s,ij} \neq 0\}$ denote the set of observed records in each system. In the first stage, we have already established some item correspondence between S_{wiki} and S_{tar} to serve as the information bridge for knowledge transfer. More specifically, this implies that $\mathcal{V}_{wiki} \cap \mathcal{V}_{tar} \neq \emptyset$. We refer the set of items as *shared items* denoted by $\widetilde{\mathcal{V}}$. Let $\mathcal{U}^* = \mathcal{U}_{wiki} \bigcup \mathcal{U}_{tar}$ and $\mathcal{V}^* = \mathcal{V}_{wiki} \bigcup \mathcal{V}_{tar}$ denote the union of the collections of users and items, respectively, where $m^* = |\mathcal{U}^*|$ and $n^* = |\mathcal{V}^*|$ denote the total number of unique users and items in the union of two systems.

In order to derive a compact model to capture the user behavior information in both systems, we introduce the Item Bridged Joint Matrix Factorization (COEDIT) model. Under COEDIT, we model the users $\mathcal{U}^*$ and the items $\mathcal{V}^*$ by a user factor matrix $\mathbf{U} \in \mathbb{R}^{k \times m^*}$ and an item factor matrix $\mathbf{V} \in \mathbb{R}^{k \times n^*}$, where the i-th user and j-th item are represented by $\mathbf{u}_i$ and $\mathbf{v}_j$ corresponding to the i-th and j-th column of $\mathbf{U}$ and $\mathbf{V}$, respectively. Let $\mathbf{U}_s \in \mathbb{R}^{k \times m_s}$ denote the matrix formed by the rows in $\mathbf{U}$ that correspond to $\mathcal{U}_s$. Similarly, let $\mathbf{V}_s \in \mathbb{R}^{k \times n_s}$ denote the matrix formed by the rows in $\mathbf{V}$ that correspond to $\mathcal{V}_s$. The goal is to approximate each rating matrix $\mathbf{X}_s$, that is $\mathbf{X}_s \approx \mathbf{U}_s^T \mathbf{V}_s$, $s \in \{S_{wiki}, S_{tar}\}$. Under the COEDIT model, the factor matrices $\mathbf{U}$ and $\mathbf{V}$ can be learned by minimizing the loss function as follows:

$$\mathcal{L} = \sum_{s\in\{S_{wiki},S_{tar}\}} (\alpha_s \sum_{(i,j)\in\mathcal{R}_s} \left(\mathbf{u}_i^T \mathbf{v}_j - X_{s,ij}\right)^2) + \lambda(\| \mathbf{U} \|_F^2 + \| \mathbf{V} \|_F^2), \quad (2)$$

where α_s is the weight of each system, and λ controls the trade-off between the rating matrix approximation errors and model complexity reflected by the Frobenius norm of the factor matrices.

In this way $\mathbf{X}_{wiki}$ and $\mathbf{X}_{tar}$ are jointly factorized and the set of factor matrices $\mathbf{V}_{wiki}, \mathbf{V}_{tar}$ for different systems becomes inter-dependent because the features of a shared item are required to be the same for knowledge sharing. If $\mathbf{X}_{tar}$ is sparse, we can still learn a good $\mathbf{V}$, benefitting from the auxiliary user behavior information carried by $\mathbf{X}_{wiki}$. Moreover, a better estimate of $\mathbf{V}$ can further improve the estimation of $\mathbf{U}_{tar}$.

4.2 COEDIT Learning Algorithm

In order to find the optimal solution for the COEDIT model, we can use the *alternating least squares* (ALS) algorithm[Zhou *et al.*, 2008] (shown in Algorithm 1) to minimize the loss function in Equation (2) with respect to $\mathbf{U}$ and $\mathbf{V}$. When one of the factor matrices is fixed, minimizing $\mathcal{L}$ with respect to the other factor matrix is equivalent to solving a least squares problem. We can easily compute the gradient of the loss function $\mathcal{L}$ with respect to different factors:

$$\nabla_{\mathbf{u}_{s,i}} \mathcal{L} = (\sum_{j\in\mathcal{V}_{s,i}} \mathbf{v}_j \mathbf{v}_j^T + \lambda E_k)\mathbf{u}_i - \sum_{j\in\mathcal{V}_{s,i}} \mathbf{X}_{s,ij} \mathbf{v}_j \quad (3)$$

$$\nabla_{\mathbf{v}_j} \mathcal{L} = (\sum_{s\in\{S_{wiki},S_{tar}\}} \alpha_s \sum_{i\in\mathcal{U}_{s,j}} \mathbf{u}_i \mathbf{u}_i^T + \lambda E_k)\mathbf{v}_j - \sum_{s\in\{S_{wiki},S_{tar}\}} \alpha_s \sum_{i\in\mathcal{U}_{s,j}} \mathbf{X}_{s,ij} \mathbf{u}_i \quad (4)$$

where E_k denotes a $k \times k$ identity matrix and $\mathcal{V}_{s,i}$ denotes the set of items rated or edited by user i in system s. Setting the gradient $\nabla_{\mathbf{u}_{s,i}} \mathcal{L}$ to zero, we obtain the following closed form expression for updating $\mathbf{u}_{s,i}$:

$$\mathbf{u}_{s,i} = \mathbf{A}_{s,i}^{-1} \mathbf{b}_{s,i} \quad (5)$$

where

$$\mathbf{A}_{s,i} = \sum_{j\in\mathcal{V}_{s,i}} \mathbf{v}_j \mathbf{v}_j^T + \lambda E_k \quad (6)$$

is a $k \times k$ matrix and

$$\mathbf{b}_{s,i} = \sum_{j \in \mathcal{V}_{s,i}} X_{s,ij} \mathbf{v}_j \tag{7}$$

is a k dimensional vector. Similarly, to update the item features $\mathbf{v}_j$, we fix the $\mathbf{U}$ and minimize $\mathcal{L}$ with respect to $\mathbf{v}_j$, which yields the following updating formulas:

$$\mathbf{v}_j = \mathbf{A}_j^{-1} \mathbf{b}_j \tag{8}$$

where

$$\mathbf{A}_j = \sum_{s \in \{S_{wiki}, S_{tar}\}} \alpha_s \sum_{i \in \mathcal{U}_{s,j}} \mathbf{u}_i \mathbf{u}_i^T + \lambda E_k \tag{9}$$

is a $k \times k$ matrix and

$$\mathbf{b}_j = \sum_{s \in \{S_{wiki}, S_{tar}\}} \alpha_s \sum_{i \in \mathcal{U}_{s,j}} \mathbf{X}_{s,ij} \mathbf{u}_i \tag{10}$$

is a k dimensional vector.

Algorithm 1 Alternating Least Squares

1: Initialize $\mathbf{U}$ and $\mathbf{V}$ with small random numbers
2: **while** $\mathcal{L}$ has not converged **do**
3: Update $\mathbf{V}$ using Equation (8)
4: Update $\mathbf{U}_{wiki}$ and $\mathbf{U}_{tar}$ using Equation (5)
5: **end while**

5 Parallel Learning for COEDIT

As we know, Wikipedia carries a huge amount of information than newly launched information systems. Thus, how to improve the efficiency of our learning algorithm is also an important issue. Recently, parallel computing algorithms are becoming more and more popular in large-scale data mining. Here we introduce a parallel learning algorithm based on the Peer-to-Peer Message Passing Interface (MPI) platform [Snir *et al.*, 1998], where our transfer learning model is implemented.

We introduce a learning algorithm implementation based on P2P communication protocol. According to the introduction of the ALS algorithm in Section 4, we may find that either of the updating procedure of $\mathbf{U}$ or $\mathbf{V}$ mainly contains three stages: i) calculate some statistics, e.g., $\mathbf{v}_j \mathbf{v}_j$ and $X_{s,ij} \mathbf{v}_j$ for $\mathbf{u}_{s,i}$; ii) collect and aggregate these statistics, e.g., Equations (9) and (10); iii) get the aggregated statistics and calculate the updated parameters, e.g., Equation (8). The stage i) and stage iii) are totally independent for different users or for different items, while only stage ii) is depending on the other entities, e.g., for each item, Equations (9) and (10) need to aggregate the statistics over a set of users.

According to above observation, we first divide the data into l blocks with equal size, and distribute them to the slave nodes. The slave nodes take over the calculation of the statistics or parameters, and the statistics aggregation for a subset of users or items. To achieve this goal, a naive method is to record which slave node takes charge of the aggregation for which user or item via a table. However, such naive method requires each slave node to keep a table, so that it is inefficient when m^* and n^* are increasing. Thus, we employ a hash function $H : n \to l$, which can easily map a user or item ID to its corresponding slave node for aggregation. The algorithm is shown as Algorithm 2.

Algorithm 2 Slave Node l P2P Procedure for COEDIT

1: Get the parameters of $\mathbf{U}_l$ and $\mathbf{V}_l$
2: **while** not instructed to terminate **do**
3: Compute the statistics of $\mathcal{M}_{l,i}^A$ and $\mathcal{M}_{l,i}^b$ using Equation (11) and (12)
4: $\mathbf{Sync_1}$: Send $\mathcal{M}_{l,i}^A$ and $\mathcal{M}_{l,i}^b$ to node $l' = H(i)$
5: $\mathbf{Sync_2}$: Receive $\mathcal{M}_{l'',i}^A$ and $\mathcal{M}_{l'',i}^b$ from other slave nodes and aggregate statistics using Equation (13) and (14)
6: $\mathbf{Sync_3}$: Send each slave node l'' the aggregated statistics
7: $\mathbf{Sync_4}$: Receive the aggregated statistics and update $\mathbf{u}_i$ using Equation (5)
8: Compute $\mathcal{M}_{l,j}^A$ and $\mathcal{M}_{l,j}^b$
9: $\mathbf{Sync_5}$: Send $\mathcal{M}_{l,j}^A$ and $\mathcal{M}_{l,j}^b$ to node $l' = H(j)$
10: $\mathbf{Sync_6}$: Receive $\mathcal{M}_{l'',j}^A$ and $\mathcal{M}_{l'',j}^b$ from other slave nodes and aggregate the statistics
11: $\mathbf{Sync_7}$: Send each slave node l'' the aggregated statistics
12: $\mathbf{Sync_8}$: Receive the aggregated statistics and update $\mathbf{v}_j$ using Equation (8)
13: **end while**

$$\mathcal{M}_{l,i}^A = \sum_{j \in \mathcal{U}_{l,i}} \mathbf{v}_j \mathbf{v}_j^T \tag{11}$$

$$\mathcal{M}_{l,i}^b = \sum_{j \in \mathcal{U}_{l,i}} X_{L,ij} \mathbf{v}_j \tag{12}$$

$$\mathbf{A}_i = \sum_l \mathcal{M}_{l,i}^A + \lambda E_k \tag{13}$$

$$\mathbf{b}_i = \sum_l \mathcal{M}_{l,i}^b \tag{14}$$

Then Equation (5) can be updated via $\mathbf{A}_i$ and $\mathbf{b}_i$

6 Experimental Results

6.1 Data Description

Movie Recommendation Datasets We conduct the experiments using datasets from two real-world recommender systems: Netflix and MovieLens. The MovieLens dataset consists of around 10 million ratings for 10,681 movies by around 71,000 users. We successfully align about 9,600 movies to the articles in Wikipedia by movie title matching. We randomly sample users' ratings for 10 times, resulting in 50,000 ratings in each time. The Netflix dataset contains over 100 million ratings from over 480 thousand users on around 17,770 movies. We also align about 11,000 movies to the articles in Wikipedia. We randomly split the 480k users into 10 folds.

Wikipedia Datasets In this paper, we incorporate Wikipedia as our external data source. Wikipedia is currently the largest knowledge repository on the Web, and the quality of its article content is remarkable due to the open editing strategy. In our experiments, we use the mirror of Wikipedia on Aug. 10, 2009. Over 10 million users register Wikipedia. There are over 20 million pages and about 3 million of them are content articles and there are over 300 million editing record, which implies each article is edited 15 times on average. Note that in cases where the different systems use different rating scales, an additional normalization step can be conducted to convert them into a common scale.

Evaluation Metrics We use Root Mean Square Error (RMSE) to evaluate the prediction quality of different models. The metric RMSE is defined as:

$$RMSE = \sqrt{\frac{\sum_{i,j}(X_{i,j} - \widehat{X}_{i,j})^2}{N}}, \qquad (15)$$

where $X_{i,j}$ and $\widehat{X}_{i,j}$ are the observed ratings and predicted ratings, respectively; and N is the total number of ratings included in the test set. Our objective is to determine, to what extent our methods can transfer the social behavior information from S_{wiki} to S_{tar}. Therefore, we evaluate the performance using the RMSEs computed on the test set of S_{tar}.

6.2 Effectiveness Test

In this section, we will evaluate the effectiveness of transferring social behaviors for collaborative filtering tasks in the target domain. For each of the two target domains, Netflix and MovieLens, We have prepared 10 folds of data. In each fold of data, we hold out 30% as the test data for evaluation and the remaining as the training set. In addition, to examine the effect of sparsity, we randomly sample a proportion training data to simulate the scenarios of different levels of sparsity. We define the density of a matrix as the ratio of known values over all matrix elements; sparsity is high when the density is low. The density varies from 0.1% to 0.9%. For parameter settings, we tune $k \in \{3, 5, 10, 15, 20\}$ and $\lambda \in \{1, 2, 5, 10, 20\}$ over the 10 folds, and report the results with best mean RMSEs. In the real-world scenario, we can set them via cross validation. For the domain weight α_s, because we already prepare the item correspondence between S_{tar} and S_{wiki}, we will not investigate the effect of domain difference in this work. For simplicity, we set $\lambda_{tar} = \lambda_{tar} = 1$. In the future work, we will further consider using some techniques for setting domain weights according to domain differences [Zhang *et al.*, 2010] .

Effectiveness on COEDIT for Knowledge Transfer In the first set of experiments, we evaluate our methods with several baseline collaborative filtering models (shown in Table 1). The first two baselines are the average filling method (AF) and latent factorization model (LFM) [Bell and Koren, 2007], which directly learn a model based on the training data in the target domain. We also include two baseline models: $T_{Content}$ considers the description articles of the movies in Wikipedia and uses the movie-word matrix to serve as $\mathbf{X}_{wiki}$ in COEDIT; T_{Link} considers the hyperlink structure of the movies in Wikepdia and uses the movie-neighborhood matrix to serve as $\mathbf{X}_{wiki}$ in COEDIT. Although these two models also transfer some knowledge from Wikipedia, either the content or the hyperlink can only reflect the topical relatedness between two items. In contrast, our model COEDIT$_{Edit}$ tries to transfer knowledge from Wikipedia in terms of social behaviors. The experimental results show that although the content based transfer learning methods $T_{Content}$ and T_{Link} may take some effect when the target domain is severely sparse, the improvements over non-transfer learning approaches, especially comparing with AF, are not so obvious. However, our method COEDIT$_{Edit}$ achieves a significant improvement on all of the target datasets, even when the target domain's density is larger than 0.7%.

In the second set of experiments, we try to answer the following question: will data sparsity in S_{wiki} affect knowledge transfer performance? As we mentioned above, in the first stage, we establish the item correspondence between S_{tar} and S_{wiki}. After that, we collected the users' editing activities on the movie articles. For Netflix, we align 11,000 movies edited by around 56,000 users, and for Movielens, we align 9,600 movies edited by about 54,000 users. Then we sort the users in a descending order according to the numbers of their edits. By cutting down different proportion of the tailed users, we can simulate different levels of sparsity for S_{wiki}. We vary the density level of S_{wiki} from 0.3% to 1.1%. The prediction performance is shown in Table 2. We can observe that when the density of S_{wiki} is low ($\leq$ 0.5%), which means S_{wiki} is very sparse, the effect of knowledge transfer is not so good. However, once the density increases to larger than 0.5%, the results of transfer learning become more and more stable.

Effectiveness on Wikipedia Data Selection In the above section, we only use the aligned movies articles together with their editing histories for knowledge transfer. In this section, we try to investigate the question: can we benefit more from an extended scope of social behaviors in Wikipedia? That means, besides the co-editing behavior on movie articles, we are interested in whether the user co-editing behavior on other related items in Wikipedia can help solve the data sparseness problem in the target recommendation systems. We try to extend the set of related items with three approaches.

The first approach is to extend the item set via co-editing bipartite graph. Our intuition is that, if two users belong to the same interest group, it is likely that they may co-edit other related items as well, e.g., books, music, etc. Thus, we first collect the set of users who edited the movie article in Wikipedia, and then we collect all the articles they have edited in Wikipedia before and add them to the item set $\mathcal{V}_{wiki}$. The editing records of the extended item set are used to form $\mathbf{X}_{wiki}$. In total, we retrieve about 100,000 items with around 20 million editing records for both Netflix and MovieLens, and the density of $\mathbf{X}_{wiki}$ is 0.31%. The second approach is to extend the item set via hyperlink graph in Wikipedia. The reason for choosing hyperlink is that, when users are editing an article of their interest, they might follow some hyper-

Table 1: Prediction performance of average filling (AF), latent factorization model (LFM), collective matrix factorization(CMF) and COEDIT. Numbers in boldface (e.g., **0.993**) are the best results among all methods

Mean and Std of RMSEs on Netflix					
Methods	Without Transfer		With Transfer		
Target density	AF	LMF	$T_{Content}$	T_{Link}	COEDIT$_{Edit}$
0.1%	1.005±0.003	1.021±0.006	1.019±0.007	1.011±0.005	**0.993**±0.003
0.3%	0.968±0.003	0.981±0.002	0.978±0.005	0.972±0.004	**0.945**±0.002
0.5%	0.930±0.002	0.957±0.002	0.951±0.004	0.945±0.004	**0.921**±0.002
0.7%	0.921±0.001	0.932±0.001	0.929±0.003	0.920±0.003	**0.894**±0.001
0.9%	0.918±0.001	0.900±0.001	0.899±0.002	0.891±0.002	**0.869**±0.001
Mean and Std of RMSEs on MovieLens					
Methods	Without Transfer		With Transfer		
Target density	AF	LMF	$T_{Content}$	T_{Link}	COEDIT$_{Edit}$
0.1%	1.041±0.004	1.057±0.009	1.051±0.009	1.045±0.007	**1.030**±0.005
0.3%	0.988±0.003	1.012±0.004	1.005±0.008	1.001±0.005	**0.963**±0.004
0.5%	0.956±0.002	0.983±0.003	0.973±0.005	0.968±0.004	**0.920**±0.003
0.7%	0.920±0.002	0.945±0.003	0.941±0.004	0.938±0.003	**0.885**±0.002
0.9%	0.912±0.002	0.894±0.002	0.890±0.003	0.888±0.003	**0.858**±0.001

Table 2: Prediction performance of COEDIT with different density of S_{wiki}. Numbers in boldface (e.g., **0.993**) are the best results among all the levels of density

Mean and Std of RMSEs on Netflix					
	Density of S_{wiki}				
Target density	0.3%	0.5%	0.7%	0.9%	1.1%
0.1%	1.007±0.005	1.001±0.005	0.998±0.004	0.995±0.004	**0.993**±0.003
0.3%	0.965±0.003	0.959±0.003	0.950±0.003	0.948±0.002	**0.945**±0.002
0.5%	0.940±0.002	0.937±0.002	0.930±0.002	0.925±0.002	**0.921**±0.002
0.7%	0.914±0.002	0.905±0.002	0.901±0.002	0.898±0.002	**0.894**±0.001
0.9%	0.896±0.001	0.886±0.001	0.873±0.002	0.870±0.001	**0.869**±0.001
Mean and Std of RMSEs on MovieLens					
	Density of S_{wiki}				
Target density	0.3%	0.5%	0.7%	0.9%	1.1%
0.1%	1.045±0.005	1.040±0.005	1.035±0.005	1.031±0.005	**1.030**±0.005
0.3%	0.976±0.004	0.971±0.004	0.967±0.004	0.965±0.004	**0.963**±0.004
0.5%	0.940±0.004	0.936±0.004	0.930±0.004	0.923±0.003	**0.920**±0.003
0.7%	0.935±0.004	0.923±0.004	0.910±0.003	0.896±0.002	**0.885**±0.002
0.9%	0.898±0.002	0.884±0.002	0.872±0.002	0.865±0.001	**0.858**±0.001

Table 3: Prediction performance of COEDIT with different sizes of extended items by user co-editing history from the Wikipedia. Numbers in boldface (e.g., **0.982**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	25,000	50,000	75,000	100,000
0.1%	0.993±0.003	0.989±0.003	0.985±0.003	0.983±0.003	**0.982**±0.003
0.3%	0.945±0.002	0.939±0.002	0.935±0.002	0.934±0.002	**0.933**±0.002
0.5%	0.921±0.002	0.917±0.002	0.913±0.002	0.910±0.002	**0.909**±0.002
0.7%	0.894±0.001	0.891±0.001	0.888±0.001	0.887±0.001	**0.886**±0.001
0.9%	0.869±0.001	0.865±0.001	0.863±0.002	0.862±0.001	**0.861**±0.001
Mean and Std of RMSEs on MovieLens					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	25,000	50,000	75,000	100,000
0.1%	1.030±0.005	1.018±0.005	1.015±0.005	1.010±0.005	**1.006**±0.005
0.3%	0.963±0.004	0.955±0.004	0.947±0.004	0.946±0.005	**0.944**±0.004
0.5%	0.920±0.003	0.916±0.003	0.912±0.003	0.911±0.003	**0.909**±0.003
0.7%	0.885±0.002	0.883±0.002	0.878±0.002	0.876±0.002	**0.875**±0.002
0.9%	0.858±0.001	0.849±0.001	0.847±0.001	0.845±0.001	**0.844**±0.001

Table 4: Prediction performance of COEDIT with different sizes of extended items by hyperlink graph from the Wikipedia. Numbers in boldface (e.g., **0.983**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	20,000	30,000	40,000	50,000
0.1%	0.993±0.003	0.988±0.003	0.985±0.003	0.984±0.003	**0.983**±0.003
0.3%	0.945±0.002	0.942±0.002	0.936±0.002	0.935±0.002	**0.934**±0.002
0.5%	0.921±0.002	0.919±0.002	0.916±0.002	0.913±0.002	**0.912**±0.002
0.7%	0.894±0.001	0.893±0.001	0.890±0.001	0.889±0.001	**0.888**±0.001
0.9%	0.869±0.001	0.864±0.001	0.863±0.002	0.863±0.001	**0.862**±0.001

Mean and Std of RMSEs on MovieLens					
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$				
Target density	0	10,000	20,000	30,000	40,000
0.1%	1.030±0.005	1.020±0.005	1.012±0.005	1.010±0.005	**1.009**±0.005
0.3%	0.963±0.004	0.956±0.004	0.947±0.004	0.945±0.004	**0.944**±0.004
0.5%	0.920±0.003	0.915±0.003	0.913±0.003	0.912±0.003	**0.909**±0.003
0.7%	0.885±0.002	0.883±0.002	0.881±0.002	0.880±0.002	**0.879**±0.002
0.9%	0.858±0.001	0.848±0.001	0.847±0.001	0.846±0.001	**0.845**±0.001

links through the article. If some hyperlinks point to some interesting articles, they may click and edit these articles as well. Thus, we collect the Tier-1 neighborhood of the aligned movie pages for extending the item set. In total, for Netflix, we retrieved 55,000 items with 6 million editing records, and the density of $\mathbf{X}_{wiki}$ is 0.19%. For MovieLens, we retrieved 42,000 items with 5 million editing records, and the density of $\mathbf{X}_{wiki}$ is 0.22%.

The last approach is to extend the item set via categorical structure. The intuition is that, it is likely that users will edit the articles belonging to the same category. Thus, we first collect the category set of the aligned movie articles, i.e., 450,000 for Netflix and 380,000 for MovieLens. Then we filter out the categories including only one movie article. Finally, we add the articles under these remaining categories to the extended item set. To summarize, for Netflix, we retrieve about 42,000 items with around 4,300,000 editing records, and the density of $\mathbf{X}_{wiki}$ is 0.18%; for MovieLens, we retrieve about 30,000 items with around 2,000,000 editing records, and the density of $\mathbf{X}_{wiki}$ is 0.14%.

The experimental results for these three methods are shown in Tables 3, 4 and 5, respectively. We can observe that when the number of related items increases, the effect of transfer learning is further improved, although slowly. Comparing with the three extending methods, the co-editing approach is a bit better. That may imply that the related items that we are interested in mainly derive their similarity more from social behaviors rather than topical closeness.

6.3 Efficiency Test

In this section, we will test the efficiency of our parallel learning algorithm. We examine the run time of the Peer-to-Peer mode algorithm that runs on a single machine with all the data from S_{tar} and S_{wiki}. For the test bed, we used a 1Gb/s LAN based cluster of 8 servers with Intel 8-core 2.93GHz CPU and 24GB memory. Each of the master and slave processes is performed using one individual core. For testing data, we use one fold of Netflix with density 0.9% as S_{tar} together its co-editing graph extended S_{wiki}. Thus, in total, we have 10 million ratings from S_{tar} and 20 million editing records in S_{wiki}. We plot the speed up achieved along with the ideal case of linear speedup in Figure 1. We can see that the parallel algorithm based on the Peer-to-Peer communication protocol achieves nearly linear speedup.

7 Conclusions and Future Work

In this work, we proposed the algorithm COEDIT to transfer coediting knowledge in Wikipedia to solve the data sparseness problem in collaborative filtering tasks. Different from traditional data mining works on Wikipedia, which focus on transferring content or structure information, our work tries to extract some collaborative social behavior in the form of co-editing history. Our experimental studies clearly demonstrate that these social knowledge can effectively help solve the data sparseness problem in other target domains. In order to improve the efficiency of our learning algorithm, we also introduced a parallel algorithm implementation for model learning.

In the future work, we will study how other social knowledge can be used to help with the tasks in other domains. We would also investigate how to analyze the domain differences for source data selection.

References

[Bell and Koren, 2007] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM*, pages 43–52, 2007.

[Blei *et al.*, 2001] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Proceedings of the 14st Annual Conference on Neural Information Processing Systems, NIPS 2001, Vancouver, British Columbia, Canada, December 3-8*, pages 601–608, 2001.

[Breese *et al.*, 1998] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of UAI 1998*, pages 43–52, 1998.

Table 5: Prediction performance of COEDIT with different sizes of extended items by category graph from the Wikipedia. Numbers in boldface (e.g., **0.982**) are the best results among all the sizes

Mean and Std of RMSEs on Netflix				
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$			
Target density	0	20,000	30,000	40,000
0.1%	0.993±0.003	0.987±0.003	0.982±0.003	**0.982**±0.002
0.3%	0.945±0.002	0.942±0.002	0.937±0.002	**0.936**±0.002
0.5%	0.921±0.002	0.918±0.002	0.915±0.002	**0.913**±0.002
0.7%	0.894±0.001	0.893±0.001	0.890±0.001	**0.889**±0.001
0.9%	0.869±0.001	0.864±0.001	0.863±0.001	**0.862**±0.001

Mean and Std of RMSEs on MovieLens				
	Size of Extended Items $\lvert\mathcal{V}_{wiki} \setminus \mathcal{V}_{tar}\rvert$			
Target density	0	10,000	20,000	30,000
0.1%	1.030±0.005	1.019±0.005	1.015±0.005	**1.012**±0.005
0.3%	0.963±0.004	0.953±0.004	0.950±0.004	**0.949**±0.004
0.5%	0.920±0.003	0.916±0.003	0.914±0.003	**0.912**±0.003
0.7%	0.885±0.002	0.884±0.002	0.883±0.002	**0.881**±0.002
0.9%	0.858±0.001	0.848±0.001	0.847±0.001	**0.846**±0.001

[Gabrilovich and Markovitch, 2005] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI 2005, Edinburgh, Scotland, UK, July 30-August 5*, pages 1048–1053, 2005.

[Gabrilovich and Markovitch, 2006] Evgeniy Gabrilovich and Shaul Markovitch. Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of The 21th National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference, AAAI 2006, Boston, Massachusetts, USA, July 16-20*, 2006.

[Gabrilovich and Markovitch, 2007a] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, Hyderabad, India, January 6-12*, pages 1606–1611, 2007.

[Gabrilovich and Markovitch, 2007b] Evgeniy Gabrilovich and Shaul Markovitch. Harnessing the expertise of 70,000 human editors: Knowledge-based feature generation for text categorization. *J. Mach. Learn. Res.*, 8:2297–2345, 2007.

[Goldberg *et al.*, 2001] Kenneth Y. Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[Herlocker *et al.*, 2002] Jon Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5(4):287–310, 2002.

[Hofmann, 1999] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, UAI 1999 Stockholm, Sweden, July 30-August 1*, pages 289–296, 1999.

[Hofmann, 2004] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[Jin *et al.*, 2003] Rong Jin, Luo Si, Chengxiang Zhai, and Jamie Callan. Collaborative filtering with decoupled models for preferences and ratings. In *Proceedings of CIKM 2003*, pages 309–106, 2003.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[Li *et al.*, 2009] Bin Li, Qiang Yang, and Xiangyang Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, pages 2052–2057, 2009.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[Ma *et al.*, 2007] Hao Ma, Irwin King, and Michael R. Lyu. Effective missing data prediction for collaborative filtering. In *Proc. of SIGIR 2007*, pages 39–46, 2007.

[Mehta and Hofmann, 2007] Bhaskar Mehta and Thomas Hofmann. Cross system personalization and collaborative filtering by learning manifold alignments. pages 244–259. 2007.

[Nathanson *et al.*, 2007] Tavi Nathanson, Ephrat Bitton, and Ken Goldberg. Eigentaste 5.0: constant-time adaptability in a recommender system using item clustering. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 149–152, New York, NY, USA, 2007. ACM.

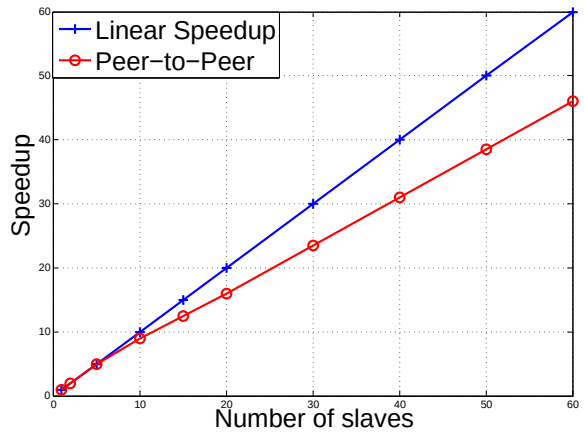


Figure 1: Speedups of the parallel learning algorithm

[Pan and Yang, 2009] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 99(1), 2009.

[Pan *et al.*, 2010] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. Transfer learning in collaborative filtering for sparsity reduction. In *AAAI*, 2010.

[Paterek, 2007] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

[Pennock *et al.*, 2000] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proc. of UAI*, pages 473–480, 2000.

[Phan *et al.*, 2008] Xuan Hieu Phan, Minh Le Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25*, pages 91–100, 2008.

[Rennie and Srebro, 2005] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719, New York, NY, USA, 2005. ACM.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.

[Sarwar *et al.*, 2001] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[Singh and Gordon, 2008] Ajit Paul Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658, 2008.

[Snir *et al.*, 1998] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI-The Complete Reference, Volume 1: The MPI Core*. MIT Press, Cambridge, MA, USA, 1998.

[Wang and Domeniconi, 2008] Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2008, Las Vegas, Nevada, USA, August 24-27*, pages 713–721, 2008.

[Wang *et al.*, 2007] Pu Wang, Jian Hu, Hua-Jun Zeng, Lijun Chen, and Zheng Chen. Improving text classification by using encyclopedia knowledge. In *Proceedings of the 7th IEEE International Conference on Data Mining, ICDM 2007, Omaha, Nebraska, USA, October 28-31*, 2007.

[Wang *et al.*, 2008] Pu Wang, Carlotta Domeniconi, and Jian Hu. Using wikipedia for co-clustering based cross-domain text classification. In *Proceedings of the 8th IEEE International Conference on Data Mining, ICDM 2008, Pisa, Italy, December 15-19*, pages 1085–1090, 2008.

[Xue *et al.*, 2005] Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proc. of SIGIR 2005*, pages 114–121, 2005.

[Zhang *et al.*, 2010] Yu Zhang, Bin Cao, and Dit-Yan Yeung. Multi-domain collaborative filtering. In *UAI*, pages 725–732, 2010.

[Zhou *et al.*, 2008] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *AAIM '08: Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.

Classification of Social Network Sites based on Network Indexes and Communication Patterns

F.Toriumi[1], **I.Okada**[2], **H.Yamamoto**[3], **H.Suwa**[4], **K.Izumi**[5] and **Y.Hashimoto**[5]
Nagoya University, Aichi, Japan[1]
Soka University, Tokyo, Japan[2]
Rissho University, Tokyo, Japan[4]
University of Electro-Communications, Tokyo, Japan[4]
The University of Tokyo, Tokyo, Japan[5]

Abstract

We analyzed data for a large number of small social network services (SNSs) and classified them in terms of their structures and communication patterns. Using this classification, we analyzed their features and found that most of them have small world, scale free, and negative assortativity characteristics. We also classified them on the basis of calculated network indexes and compared the four types. Finally, we classified their communication patterns and identified four types of friend networks: partial, parity, inclusive, and independent.

1 Introduction

As part of the steady growth of new network communication tools, the expansion of social network services (SNSs) such as Facebook and orkut is greatly affecting societies worldwide.

There have been many previous studies of online social networks. Adamic et al. [Adamic *et al.*, 2003], for example, studied a university SNS called Nexus and analyzed its structure and the attributes and personalities of its users. Yuta et al. [Yuta *et al.*, 2007] investigated the network structure of the mixi, and discovered a gap in the community-size distribution that is not observed in real social networks.

Moreover, they developed a simple model to account for this feature. Ahn et al.[Ahn *et al.*, 2007] compared the structures of three online SNSs, (Cyworld, MySpace, and orkut), each with more than 10 million users. They also analyzed the historical evolution of the topological characteristics of Cyworld. These studies mainly focused on large-scale SNSs for general users. In addition to such SNSs, many examples of user-limited SNSs can also be found, such as campus, company, and regional SNSs, that provide specialized services for a limited number of users and thereby effectively stimulate user communication on the Web. These user-limited SNSs are now receiving more attention due to their business potential.

However, SNS studies have been mostly on particular large-scale SNSs, so we cannot say whether their results apply to general features or to special characteristics of SNSs. From the point of view of comparison analysis, a comparison of only a few types of SNS may not produce statistically significant results. We have analyzed a wide variety of SNSs with the aim of classifying them using several approaches.

In this paper, we describe our classification of a large number of small-scale SNSs and our analysis of their features from the viewpoints of network structure and communication pattern.

2 Social Network Data

We analyzed data for 615 SNSs, each with more than 50 users. The data were provided by So-net Entertainment Corporation, which provides SNS support. Using the user relationship data provided, we constructed a friend-network model for each SNS and used it to analyze their network structures.

So-net 's SNS support has three features in particular.

- Anyone can create a social network service.
- The SNS administrator can choose if the SNS permits "registration" to participate.
- Anyone who registers automatically becomes a friend of the administrator.

The data analyzed included four parameters of particular interest.

1. User(user ID, date on registered)
2. Link(link ID, user id, user ID, date to created)
3. Blog entry(blog ID, user ID, date on entered)
4. Blog comment(comment ID, blog ID, comment user ID, comment date)

3 Network Structure Analysis

3.1 Distribution of network indexes

Previous analysis of the structures of large-scale SNSs, e.g., Cyworld[Ahn *et al.*, 2007] and mixi[Yuta *et al.*, 2007], has shown that SNSs can have both " small world " and " scale free " characteristics. However, a question remained as to whether these characteristics are commonly found in various sized networks. We thus statistically analyzed data for a large number of SNSs to clarify the characteristics of SNSs.

Average Path Length and Cluster Coefficients
We investigated whether the SNSs we focused on have the small world characteristic by using their average path lengths and cluster coefficients[Watts and Strogatz, 1998].

First, we determined the distribution of average path length L. The average L and standard deviation for the SNSs were respectively 2.13 and 0.339. The mean average path length was approximately 2.1, and about 56% of the average path lengths were between 1.9 and 2.1. That is, SNSs tend to have very short average path lengths.

The average cluster coefficient C and the standard deviation were respectively 0.377 and 0.206. The cluster coefficients had a wide range, $0.1 \leq C \leq 0.8$. However, about 74% of the cluster coefficients were greater than 0.2. That is, SNSs tend to have high cluster coefficients.

These findings indicate that most SNSs have a small world characteristic.

Degree Distribution
We then investigated whether the SNSs have a scale free characteristic by using their degree distributions [Barabási and Albert, 1999].

To determine whether their degree distributions followed a power law, we calculated their determination coefficients, R^2, by using

$$R^2 = 1 - \frac{\sum_i (\log(y_i) - \log(f_i))^2}{\sum_i (\log(y_i) - \overline{\log(y)})^2}, \quad (1)$$

where $\log(y_i)$ is the logarithmically transformed observed values, $\log(f_i)$ is the logarithmically transformed estimated values of the power law obtained by regression analysis of the degree distribution, and $\overline{\log(y)}$ is the logarithmically transformed average of the observed values. This equation shows that the closer the value of R^2 to 1, the closer the distribution follows a power law. The average determination coefficient was 0.631, and the standard deviation was 0.176. This indicates that the degree distributions of SNSs tend to approximately follow a power law.

Next, we calculated the power indexes for the SNSs with a degree distribution that followed the power law, that is, those with R^2 greater than 0.6. There were 409 SNSs that met this condition. The average power index (γ) for these SNSs was -0.908, and the standard deviation was 0.155. The power index for mixi was about -2.4, which is smaller than the SNSs in So-net SNS. Therefore, the ratio of users with many friends was higher than that of mixi.

Assortativity
Finally, we investigated the distribution of the assortativity of the SNSs. Assortativity, r, is defined as an index showing the degree of correlation between connected nodes. Its value range is $-1 \leq r \leq 1$. When the value r is greater, two nodes that both have high degrees tend to be connected. On the other hand, when the value r is smaller a node with high degree and a node with low degree tend to be connected. The calculated r for mixi was 0.121[Yuta *et al.*, 2007], and that for Cyworld was -0.13[Ahn *et al.*, 2007].

The assortativity of our SNSs was -0.471 on average with a standard deviation of 0.207. Interestingly, all of the SNSs except three (0.5% of the number) had a negative assortativity. This indicates that the users with higher degrees tended to connect with users with lower degrees. We partially attribute this to the existence of a core group of members. These core members actively recruit friends to join the network, so they have many links to other members. Moreover, although these friends tend to accept the invitation, only a few become active users. As a result, the active users have higher degrees, and the neighbors have smaller degrees. The assortativities thus become negative.

The results of our network structure analysis are summarized in Table 2.

These network indexes show that the SNSs we investigated have small world, scale free, and negative assortativity characteristics.

Table 1: Network Indexes

	L	C	R^2	γ	r
Average	2.13	0.377	0.631	-0.908	-0.471
Std. Dev.	0.339	0.206	0.176	0.155	0.207

3.2 Comparison with the Other SNSs

We compared the index values we obtained for the So-net SNSs with those for Flickr [Mislove *et al.*, 2007], orkut[Mislove *et al.*, 2007], Cyworld[Ahn *et al.*, 2007], and mixi[Yuta *et al.*, 2007]. As shown in Table 2, these other SNSs respectively had 1,846,198, 3,072,441, 12,048,186, and 363,819 users at the time the data was collected. (The fault data has been omitted.)

As shown in Table 2, average path length, L, power index γ, and assortativity r had various values. The reason for the big difference between the average path length for the So-net SNSs and the other SNSs is attributed to the great difference in the number of nodes. The assortativity is discriminative because the So-net SNSs are leaning a negative direction widely while that of the other SNSs is non-negative. As mentioned, the general network indexes of the So-net SNSs had different values than those of the other SNSs. This suggests that the So-net SNSs have a vastly different friend network structure to other SNSs, while their sites have similar system feature.

Table 2: Comparison of Network Indexes

	No. of Users	L	C	γ	r
Flickr	1846198	5.67	0.313	-1.74	-
orkut	3072441	5.88	0.171	-1.50	-
Cyworld	12048186	-	0.16	-	-0.13
mixi	363819	5.53	0.328	-2.4	0.121
So-net SNSs	257.6	2.13	0.377	-0.908	-0.471

3.3 Classification of SNSs by Clustering approach

We classified the So-net SNSs by using a clustering approach from view point of the network structure, which based on cal-

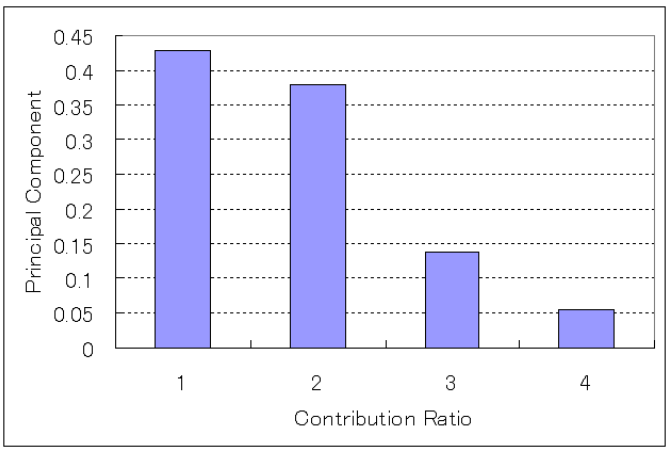


Figure 1: The contribution Ratios of Principal Components

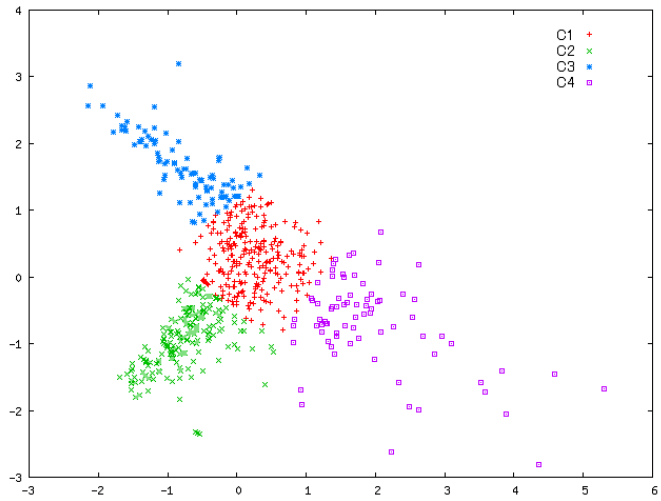


Figure 2: Two-Dimensional Mapping of Network Structures

culated L, C, R^2, and r. Note that γ was not used because some So-net SNSs do not comply with power distribution.

We formulated these four indexes as a character vector v_i,

$$v_i = \left[\frac{L_i}{\sigma_L}, \frac{C_i}{\sigma_C}, \frac{R_i^2}{\sigma_{R^2}}, \frac{r_i}{\sigma_r}\right] \quad (2)$$

where $\sigma_L, \sigma_C, \sigma_{R^2}, \sigma_r$ show standard variations of average path lengths of all SNSs, clustering coefficient, determination coefficients of power low, assortativity, respectively.

In order to observe them easily, we performed principal component analysis, and then clustered the SNSs into four types on the basis of the primary and secondary components because their contribution ratios were notably high. The contribution ratios of principal components are shown in Fig.fig:ContributionRatio A two-dimensional mapping of character vector of each SNSs are shown in Fig.2. We used the k-means clustering method with the number of partitions k equal to 4. To take into account the errors in the initial values produced by this method, we used the case in which the variance ratio among classes was the highest of the multiple cases with different initial values. The calculated average values for the four indexes and the number of SNSs are shown in Table 3 by type. We call each types of SNSs as C1 to C4.

3.4 Characteristics of Clusters

Roughly 40% of the SNSs were classified as C1. These SNSs had both small world and scale free characteristics, so these characteristics should be commonly found in SNSs.

The C2 type SNSs had a substantially smaller average cluster coefficient than the C1 SNSs, meaning that they had a smaller degree of cohesion. Their average assortativity was also substantially smaller, meaning that some of the users in the C2 SNSs exerted traction on the SNS. The average number of users was similar between the two types, but the average number of links in the C2 SNSs was only about 30% that in the C1 SNSs. Moreover, one user in particular had an average of about 78% of the links in the C2 SNSs, meaning that one side of each node pair was almost always the same user. These SNSs thus had an extreme star topology in which one user was connected with all the other users, and all the other users were connected with only that one user.

The C3 SNSs had a higher average cluster coefficient and a shorter average path length L, so they had a higher degree of cohesion. These SNSs had about 88 users on average, which is quite low, but the average degree (number of links) was 16.1, which is extremely high compared to the average for all 615 SNSs (4.94). Therefore, they are close to a complete graph in which members in that SNS are intense relationship.

The C4 SNSs were similar to the C1 SNSs but have longer average path lengths. The average path length for the C4 SNSs was 2.85, significantly higher (0.1% of significant level) compared with that for all the SNSs. Since many of the average path lengths were close to 2.0 because all members tend to connect with the administrator in the So-net SNSs, an average path length greater than 2 means that there are many pairs of users which do not includes administrator. In addition, the C4 SNSs had about 424 friends on average, which is relatively high. This suggests that the C4 SNSs are growing out of the administrator's hands.

The four types of SNSs are diagrammed in Fig.3.

4 Features of SNSs and Analyses of Activation on Users Behaviors

We analyzed the relationship between a user 's friend network and the user 's communication behavior. While a friend network, which consists of links directly, is explicit, communication behaviors is implicit. To determine the correlation between network and behavior, we focused on several features of the communication behavior and identified various patterns of user behavior activation.

We used the 309 Sonet-SNSs sites that satisfied two conditions. 1) The number of users was at least 100 because the analysis would have been meaningless if the number of users actively communicating was small. 2) There were entries of blogs and comments because we needed for our analysis not

Table 3: Average Values of Network Indexes by Four SNS Types

	No. of Users	No. of Links	L	C	R^2	r	No. of SNSs
C1	283.6	1001.1	2.095	0.436	0.713	-0.388	263
C2	236.9	287.5	2.025	0.163	0.573	-0.721	184
C3	87.9	743.0	1.833	0.686	0.380	-0.369	92
C4	423.7	1454.1	2.851	0.313	0.783	-0.280	76

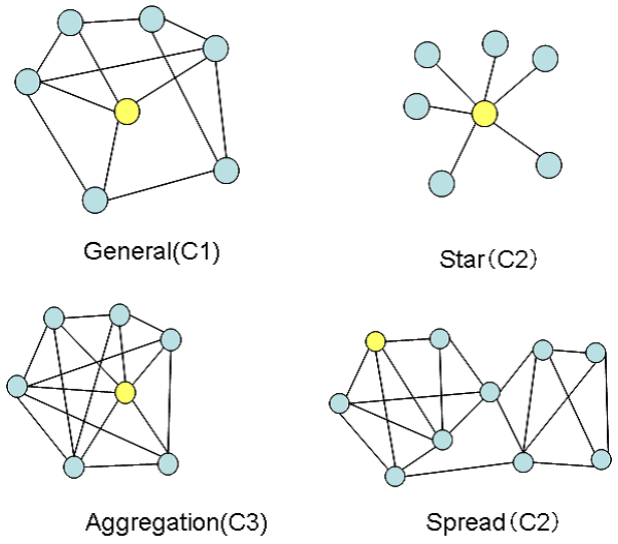


Figure 3: Four Types of SNSs

only the friend network but also blogs and comments to analyze a network of communication behaviors.

We also need appropriate indexes showing how a posted comment on a blog entry is related to the friend network structure, and we need to know what type of behavior patterns the comment has. Can a comment have a "behavior pattern"? To obtain this information, we define an aggregation ratio for friends and a coverage ratio for friends, respectively.

4.1 Index Formulation

The aggregation ratio for friends A is the rato of comments to friends in all the comments. The higher the ratio, the more the comments for blog entries are restricted to friends. The coverage ratio for friends is the ratio of friends who post comments in all the friends who post blog entries. The higher the ratio, the more the actual communications are take place on friend relationships.

$$\text{Aggregation ratio (A)} = \frac{\text{no. of comments for friends}}{\text{no. of all comments}} \quad (3)$$

$$\text{Coverage ratio (C)} = \frac{\text{no. of friends who post comments}}{\text{no. of friends of blog entried user}} \quad (4)$$

4.2 Aggregation and Coverage Ratios

We classified the communication patterns of the SNSs on the basis of the median values of these two indexes (0.737 and 0.610, respectively), as shown in Fig. 4.

- Partial friend network type
  SNSs with a high aggregation ratio and a low coverage ratio. Members communicate with only a limited group of friends.
- Parity friend network type
  SNSs with both high aggregation and coverage ratios. Members communicate within their friend network cyclopaedically, but few communicate with people outside their friend network.
- Inclusive friend network type
  SNSs with high coverage ratio and low aggregation ratio. Members communicate within their friend network cyclopaedically and many communicate with people outside their friend network.
- Independent friend network type
  SNSs with both low aggregation and coverage ratios. Members communications independently of their friend network.

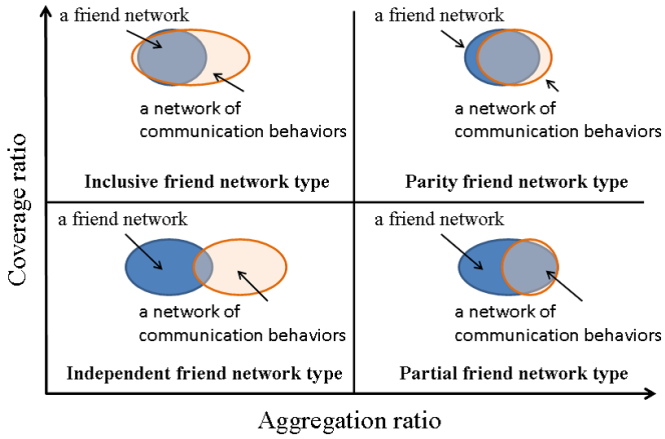


Figure 4: Four Types of Communication Patterns based on Aggregation and Coverage Ratios

4.3 Structural Traits of SNSs based on Communication Pattern

We analyzed several structural traits of SNSs on the basis of their communication patterns: number of users, average degree of cohesion, duration of existence, average path length, cluster coefficient, assortativity, and power index, as shown in

Table 4: SNS Types based on Communication Patterns

communi-cation patterns	N	No. Users	Av. Degree	Duration of Est.	Av. Path Length	Cluster Coef.	Assort-avity	Power Index
Partial	81	131.562	199.599	432.173	2.146	.436	-.360	-.826
Parity	73	137.342	175.342	524.699	2.137	.369	-.403	-.869
Inclusive	81	168.815	128.370	457.728	2.182	.259	-.444	-.940
Independent	74	182.953	115.264	439.014	2.092	.267	-.479	-.935
Chi-Square or F-Value		17.607(x)	45.801(x)	1.795(f)	.946(f)	20.111(f)	4.846(f)	5.525(f)
Significance Prob.		.001***	.000***	.148	.418	.000***	.003**	.001**

$***p < .001$, $**p < .01$
(x) means chi-square value and (f) means F-value.

Table 4. The N in the figure represents the number of SNSs of that type.

The inclusive and independent types have a larger number of users. These types have a low aggregation ratio for friends. This suggests that SNSs with many members who frequently communicate with people outside their friend network tend to be large. The partial and parity types, on the other hand, have high average degrees of cohesion. This suggests that SNSs with many members who limit their communication to within their friend network are thick. The suggestion is supported by the high cluster coefficients of these type SNSs.

4.4 Effect of Communication Pattern on User Behavior Activation

We investigated how the communication pattern affects the activation of user behavior in an SNS. We used the average number of comments posted by a user per day, the number of posting blog entries, the number of user who browsing from PC, and the number of user who browsing from mobile phones as indexes of activation. We tested its differences by using the Kruskal-Wallis test.

As shown in Table. 5, the SNSs with a higher coverage ratio for friends were more active. This makes sense because an SNS is a communication space based on a friend network. A chi-square test showed that the aggregation ratio had no effect on activation. Nevertheless, the communication traits of SNSs do depend on their aggregation ratio. In a parity friend network, communication is only among friends, suggesting that such networks are used to sustain friendships and as a communication tool for everyday matters. In an inclusive friend network, communication is frequently with people outside the friend network, suggesting that they are used mainly to communicate on specific themes or topics.

4.5 Features of Contributive Members and Relationship to Activation

We analyzed the relationship between the activation of user behavior and the contribution of the core users for each type of SNS in order to clarify the role of the core users in the activation. A core user is defined here as a user who plays a central role in network activities such as the administrator. Does the activation pattern when the members on postings are core users different from that when they are edge (not core) users?

To answer this question, we define an index of degree contribution. This index is defined to find whether high-degree user often posts comments or not. The index of degree contribution D_c is calucurated as follows:

$$D_c = \frac{1}{N} \sum_i c_i \cdot d_i \tag{5}$$

where c_i is number of user$_i$'s comments, and d_i is degree of user$_i$. As shown in Table 6, you can observe Kendall's rank correlation coefficients of the index of degree contribution, their significant probabilities, and the indexes of activation for every communicatio pattern.

In the case of the parity friend network, we verify a negative correlation in the contributing degree on links and many indexes on activation and significant tendencies for the number of posting comments and that of posting blog entries. This suggests that there is a negative correlation between the postings of users with a relatively high degree of cohesiveness and activation of behavior in parity type SNSs. The communications in such networks is restricted to within the friend network, and the communication are derived as an extendsion of their daily lives. Therefore, such communication may not need the existence of core members or their involvement.

In the case of the inclusive friend network type, on the other hand, we found a positive correlation relationship between the contributing degree on links and many of the activation indexes. The active involvement of the core members may be needed to activate behavior. Communications tends to go beyond the friend network, so the communications may be for a specific interest or topic rather than for daily matters. These type networks include those for a specific topic such as a disease and those for a specific person, such as a musician. Therefore, the administrator and/or core users play a key role in activating behavior because they work as a traffic controller.

5 Conclusion

We analyzed data for a large number of small SNSs and classified them on the basis of their network structure and their communication pattern. Using the results of this classification, we analyzed several of their features. We found that

Table 5: Effect of Communication Pattern on Activation of User Behavior

Communication pattern	N	Comments No. Postings	Blog Entries No. Postings	From PC No. Browsing	From mobile No. Browsing
Partial Type	81	135.556	145.111	165.654	142.105
Parity Type	73	186.178	170.918	165.849	158.925
Inclusive Type	81	182.296	178.926	160.327	180.432
Independent Type	74	115.649	123.932	126.804	137.405
Chi-square value		34.642	18.066	9.886	11.261
Significant Prob.		.000***	.000***	.020*	.010*

$***p < .001, *p < .05$

Table 6: Relationship between Activation of User Behavior and Contribution of Core Users for Each Type of SNS†

Communication pattern	Comments No. Postings	Blog Entries No Postings	From PC No. Browsing	From mobile No. Browsing
Partial Type	.046	-.057	-.012	-.015
N=81	.541	.448	.870	.845
Parity Type	-.138	-.154	-.129	-.096
N=73	.085+	.054+	.107	.230
Inclusive Type	.180	.137	.174	.052
N=81	.018*	.070+	.021*	.491
Independent Type	.051	.057	.031	-.050
N=74	.517	.469	.692	.526

†Upper values are Kendall's rank correlation coefficients and the lower values are their significant probabilities

$*p < .05$

$+p < .10$

most of them had small world, scale free, and negative assortativity characteristics. We also classified SNSs them on the basis of network indexes and used the results to analyze several other features. A third classification based on communication pattern revealed four types of friend network: partial, parity, inclusive, and independent.

Future work includes clarifying the trajectory of those growing processes by using time analysis. We also plan to analyze the activation and inactivation of behavior by SNS type. The results of the work reported here and of future analysis will enable more effective SNS management.

Acknowledgment

We thank So-net Entertainment Corporation for providing the data of So-net SNS β Version.

References

[Adamic *et al.*, 2003] L.A. Adamic, O. Buyukkokten, and E. Adar. A social network caught in the Web. *First Monday*, 8(6):29, 2003.

[Ahn *et al.*, 2007] Y.Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. *Proceedings of the 16th international conference on World Wide Web*, pages 835–844, 2007.

[Barabási and Albert, 1999] A.L. Barabási and R. Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509, 1999.

[Mislove *et al.*, 2007] A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42. ACM, 2007.

[Watts and Strogatz, 1998] DJ Watts and SH Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.

[Yuta *et al.*, 2007] K. Yuta, N. Ono, and Y. Fujiwara. A Gap in the Community-Size Distribution of a Large-Scale Social Networking Site. *Arxiv preprint physics/0701168*, 2007.

Measuring Semantic Similarity using a Multi-Tree Model

Behnam Hajian and **Tony White**
School of Computer Science Carleton University, Ottawa, Canada
{bhajian,arpwhite}@scs.carleton.ca

Abstract

Recommender systems and search engines are examples of systems that have used techniques such as Pearson's product-momentum correlation coefficient or Cosine similarity for measuring semantic similarity between two entities. These methods relinquish semantic relations between pairs of features in the vector representation of an entity. This paper describes a new technique for calculating semantic similarity between two entities. The proposed method is based upon structured knowledge extracted from an ontology or a taxonomy. A multi-tree concept is defined and a technique described that uses a multi-tree similarity algorithm to measure similarity of two multi-trees constructed from taxonomic relations among entities in an ontology. Unlike conventional linear methods for calculating similarity based on commonality of attributes of two entities, this method is a non-linear technique for measuring similarity based on hierarchical relations which exist between attributes of entities in an ontology. The utility of the proposed model is evaluated by using Wikipedia as a collaborative source of knowledge.

1 Introduction

Similarity refers to psychological nearness between two concepts. Similarity has roots in psychology, social sciences, mathematics, physics and computer science [Larkey and Markman, 2005]. In social psychology, similarity points to how closely attitudes, values, interests and personality match between people which can lead to interpersonal attraction. This can be explained by the fact that similar people tend to place themselves in similar settings and this consequently decreases potential conflicts between them. Furthermore, finding a person with similar tastes helps to validate values or views held in common. With a mental representation, the similarity between two concepts is defined as a function of the distance between two concepts represented as different points in the mental space [Tversky and Shafir, 2004].

Semantic similarity is used to refer to the nearness of two documents or two terms based on likeness of their meaning or their semantic contents [Tversky and Shafir, 2004]. Conventionally, statistical means (e.g., a vector space model) can estimate the distance between two entities by comparing features representing entities [Salton *et al.*, 1975]. For example, in order to compare two documents, the frequency of co-occurrence of words in the text corpus represents the similarity between the two documents. Semantic relatedness is a broader term than semantic similarity, with the former including other concepts such as antonymy and meronymy. However, in certain literature these two terms are used interchangeably .

We can compare the similarity of two concepts by measuring the commonality of their features. Since each concept is represented by the features describing its properties, a similarity comparison involves comparing the feature lists representing that concept. Simply put, concepts which are near to each other are more similar than points which are conceptually distant. There are several mathematical techniques for estimating this distance, such as latent semantic analysis (LSA) [Landauer *et al.*, 1998]. Measuring similarity among entities has applications in many areas such as: recommendation systems, e-commerce, search engines, biomedical informatics and in natural language processing tasks such as word sense disambiguation.

For instance, in user-based collaborative filtering, the system tries to find people with similar tastes and recommend items highly ranked by the people which might be interesting to their peers. Finding people with similar tastes involves processing of their historical transactions (i.e., items viewed and ranked by them in their previous transactions) and calculating similarity between them using one of the methods described above. On the other hand, in content-based recommender systems and search engines, the system finds items which are more similar to show to a user based on his/her query and the similarity of the items (i.e., products). This category of system calculates similarity between products based on the commonality of the features of different products.

In information retrieval (IR) and search engines, words are considered as features in a document or a query. In IR systems, it is conventional to represent a document by a bag-of-words (BOW). A Vector Space Model (VSM) is generally used to estimate the similarity between two documents in classification/clustering tasks or to estimate similarity between a query and documents in keyword-based search engines.

1.1 Contribution, Motivations and Paper Structure

In the Vector Space Model (VSM), a document or a query is represented as a vector of identifiers such as index terms. However, in many cases, conventional methods such as Dices coefficient, Pearson's correlation coefficient, Jaccards index or cosine similarity, which use VSM to represent a document, do not perform well. This is due to a document being represented in a linear form (i.e., a vector of features) in which semantic relations among features are ignored. Examples of such problems are: ignoring polysemy (terms having different sense with a same spelling such as apple as a fruit and apple as a company) and synonymy (terms with different spelling having a same sense such as big and large). An example of the latter problem can be found in recommender systems which find people with similar tastes according to their previous transactions. An example of this problem is demonstrated in the following example in which similarity of tastes for two people are estimated based on their previous transactions:

- T1 (Clothes, Boxspring, Mp3Player, Mattress, LCD TV)
- T2 (Dress, Bed, Mattress, iPod Touch, LED TV)

Using the VSM-based method for computing similarity between the above transactions, these transactions are no longer similar at all. However, intuitively we have a feeling that LED TV and LCD TV are related to each other since both are subclasses of TV. This observation is also true when comparing iPod Touch and Mp3 Player and for the relationship between Clothes and Dress.

This paper proposes replacing the VSM with a non-linear representation for an entity. The proposed representation models an entity by its features in a hierarchical format using an ontology called a semantic multi-tree. Multi-tree similarity considers semantic relations among the features of entities in a hierarchical structure using the ontology classification. This method enhances conventional information retrieval techniques by computing similarity regarding commonality of not only the features but also commonality of semantic relations among features and their parents.

The rest of the paper is organized as follows: The next section provides background information on the VSM including analysis of its limitations as well as related work. In Section 3, we define our model for representing entities called the semantic multi-tree. Section 4 concentrates on the definition of the proposed method for measuring similarity by use of a semantic multi-tree model. In the following section, the technique is validated against human judgment in WordSimilarity-353 and Rubenstein and Goodenough using Wikipedia categories as a taxonomy. A discussion follows, with conclusions and future work provided in the final section.

2 Background and Related Work

The Vector Space Model is defined as an algebraic model in which a document or an entity is represented as a vector of its features; for example, a document which is represented as a vector of index terms [Salton and McGill, 1983]: $d_j = (w_{1j}, ..., w_{nj})$. In these vectors, w_{ij} represents the number of occurrences of the i^{th} term in the j^{th} document. There are two model representation schemes. The superior scheme for representation of vectors in this model is term frequency-inverse document frequency (tf-idf). In the other scheme, w_{ij} is a binary representation of the occurrence of a corresponding term. In order to retrieve a document among a collection of documents, we have to calculate the similarity between our query and all of the documents in the collection and choose the most relevant documents among them. A frequently used method for estimating the similarity is calculating the cosine of the angle between the vectors representing query and a document. The higher the cosine of the angle between two vectors the more similar the vectors and, therefore, the more similar the entities represented by the vectors.

$$cos\theta = sim(d_i, q) = \frac{d_i.q}{|d_i||q|} \tag{1}$$

Another method for calculating similarity is Pearson product-moment correlation coefficient (PMCC). This method calculates the correlation (linear dependence) between two vectors. The PMMC of two vectors is defined as:

$$sim(d_i, q) = \frac{cov(d_i.q)}{\sigma d_i \times \sigma q} \tag{2}$$

Calculation of similarity is straightforward with these methods but a disadvantage is that neither of them considers semantic relations among features.

A conventional method for computing semantic relatedness is the corpus-based technique that relies on the tendency for related words to appear in similar texts called Latent Semantic Analysis (LSA). Unfortunately, LSA is only able to provide accurate results when the corpus is very large.

In recent years, several techniques have been developed – such as the work proposed by Ted Pedersen et. al – for estimating similarity by measuring semantic distance between two words in WordNet [Pedersen *et al.*, 2005]. A limitation of using lexical databases such as WordNet or similar resources is that they have been created by one or a group of linguists rather than experts in different subjects. Furthermore, WordNet is rarely revised when compared to collaborative knowledge sources such as Wikipedia. As a result, WordNet does not include some special proper nouns in different areas of expertise (e.g., Obama, Skyneedle). Recently, Wikipedia has compensated for this lack of knowledge by providing a mechanism for collaboratively creating knowledge. Wikipedia includes a wide range of articles about almost every entity in the world by using human expertise in different areas. In addition, as of May 2004, Wikipedia articles have been categorized by providing a taxonomy; namely, categories. This facility provides hierarchical categorization with multiple parents for a node by means of a multi-tree structure. This observation motivates us to use Wikipedia as a resource of knowledge in this paper.

There are several approaches for measuring semantic relatedness using resources such as WordNet or Wikipedia categories as a graph or network by considering the number or length of paths between concepts. In the WikiRelate project,

Ponzetto and Strube used three measures for computing semantic relatedness: First, a path-based measure using the length of the path between two concepts; second, an information content-based measure and third, the overlap-based measure which applies the Lesk algorithm that defines the relatedness between two words as a function of the overlap between two contexts defining the corresponding words. In WikiRelate [Strube and Ponzetto, 2006], a pair of Wikipedia pages is first retrieved, then categories they refer to are extracted and finally, the relatedness between two concepts is computed regarding the paths found between two concepts in the Wikipedia categories. In the last step, Ponzetto and Strube calculate relatedness by selecting the shortest path and the paths which maximize the information content-based measure.

In contrast with statistical methods for computing relatedness such as LSA, Gabrilovich and Markovitch proposed Explicit Semantic Analysis (ESA) using meaning in natural concepts derived from Wikipedia [Gabrilovich and Markovitch, 2007]. In this work, they used Wikipedia articles for augmenting the text representation and constructing a weighted list of concepts. They finally used tf-idf and conventional machine learning methods to calculate relatedness between the weighted vectors constructed in the previous steps.

Milne and Witten used cross referencing in the Wikipedia link database in order to obtain semantic relatedness between two concepts called Wikipedia Link-based Measure (WLM) [Witten and Milne, 2008]. In WLM, they used tf-idf using link counts weighted by the probability of occurrence of a term in an article. Almost all of the previous research has used tf-idf and VSM to calculate the relatedness between two sets of features.

3 The Semantic Multi-Tree Model

Semantic tree is a term with several different meanings in computer science. The term is regularly used as an alternative term for a semantic tableaux which is a very well known method in logic (i.e., a resolution method for mechanized reasoning) [Annates, 2005]. Semantic tree in this paper is interpreted as the taxonomy of entities in an ontology.

Taxonomy in the literature is defined as the practice and science of classification. Almost all objects, places, concepts, events, properties and relations can be classified into a taxonomic hierarchy. In the ontological literature, taxonomy refers to a set of concepts with *is-a* (i.e., SubClassOf/InstanceOf) relations between them. Therefore, we consider taxonomy as a narrower concept than ontology since ontology includes broader relations such as *part-of*, *has-a*, *rules*, *axioms* and *events* as well as *classes*, *hierarchical relations* and *attributes*.

Definition 1: A taxonomy, $\mathcal{O}$, is defined as a set of classes, and *is-a* relations between them, $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$. In formal logic, the *is-a* relation is defined as a *subclass/instance-of* relation in an ontology:

Subclass/Instance-of: $\forall x : c_i(x) \rightarrow c_j(x)$. Such that $\forall c_i, c_j \in \mathcal{C}_\mathcal{O}, \textit{is-a}(c_i, c_j) \in \mathcal{R}_\mathcal{O}$.

Definition 2: A Multi-Tree is defined as a tree data structure in which each node may have more than one parent. A multi-tree is often used to describe a partially ordered set. In this paper, a taxonomy of concepts is modelled by a multi-tree structure in which each concept may refer to multiple super-concepts. It should be noted that in taxonomies such as Wikipedia Categories and WordNet cycles do exist; however, we have avoided capturing them by breaking edges creating directed cycles and capturing the rest of the graph by using a multi-tree structure. It should also be noted that the definition used here is more general in that the algorithms used allow for the existence of multiple paths between a leaf and root node; i.e., the diamond-free poset requirement is relaxed.

Formally, in this paper, a multi-tree is a directed acyclic graph, $T = (V, E, C, L, M, W, P)$, with hierarchical categorization of its nodes in different levels such that:

- *V* is a set of vertices (nodes), $V = \{v_1, ..., v_n\}$. Each vertex corresponds to a concept in the taxonomy.
- *E* is a set of edges, $E = \{e_1, ..., e_n\}$, (in which $e = \langle v_i, v_j \rangle$ is an ordered set representing an edge from node v_i to node v_j. Each edge represents an *is-a* relation between two concepts c_i, c_j which means (c_i *is-a* c_j). The direction in this digraph is always from a concept (subclass) to its parent (super-class).
- *C* is a set of terms representing concepts which are used as nodes labels.
- *L* is a function mapping V to $\mathbb{R}$ $L : V \rightarrow \mathbb{R}$ assigning a real number to each node. This function is recursively defined as being 1 plus the average value of L for the children of the node. Initially, this function assigns 0 to the leaf nodes.
- *M* is a bijective mapping function mapping V to C ($M : V \rightarrow C$) assigning a label (representing a concept) to a node.
- *W* is a function mapping V to $\mathbb{R}$ ($W : V \rightarrow \mathbb{R}$) which assigns a real number as a weight to each node. This weight is utilized to calculate the similarity between two entities, which will be discussed in the next section.
- *P* is a function mapping E to $\mathbb{R}$ ($P : E \rightarrow \mathbb{R}$) which assigns a real number to each edge as a propagation ratio of each edge. In this paper P was set to 1.

The following functions, properties and operators are defined for a Multi-Tree:

- *leaf(v)* is a function mapping V to $\{true, false\}$ that returns a Boolean value indicating whether a node is a leaf node or not. A leaf node in Multi-Tree does not have any children. A multi-tree may have several leaves.
- *root(v)* is a function mapping V to $\{true, false\}$ that returns a Boolean value indicating whether a node is a root node or not. A Multi-Tree node is a root if it does not have any parents. A multi-tree has only one root node.
- *children(v)* is a function mapping V to P(V) (the power set of V) that returns the set of all the direct children of the node.
- *parents(v)* is a function mapping V to P(V) (the power set of V) that returns the set of all the direct parents of the node.

- $\beta_v = |children(v)|$ is defined as the cardinality of the child set of node v. (count of children of the node v)
- $\gamma_v = |parents(v)|$ is defined as the cardinality of the parent set of node v. (count of parents of the node v)
- The combination operator with the symbol $\uplus$ is defined between two multi-trees T_1, T_2 and returns a multi-tree T_u containing all the vertices and edges that exist in both T_1 and T_2. In other words, this operator returns the combination of two multi-trees. $T_u = T_1 \uplus T_2 \Rightarrow$

$$T_u = \begin{cases} E_u = E_1 \cup E_2 \\ V_u = V_1 \cup V_2 \\ C_u = C_1 \cup C_2 \end{cases} \begin{cases} L^{T_u} \\ M^{T_u} \\ P^{T_u} \\ W^{T_u} \end{cases} \quad (3)$$

- The weights of the vertices in the tree T_u are calculated by a recursive function $W^{T_u} : V \times \mathbb{R} \rightarrow \mathbb{R}$ as defined in equation 4. In the proposed algorithm, weight is propagated from the leaves to the root of the multi-tree combined from two multi-trees. In this equation, α is a damping factor (degradation ratio). The damping factor causes the nodes at lower levels of a multi-tree (i.e., nodes near to the leaves) to contribute more to the weight than nodes in higher levels.

$$W^{T_u}(v_i, \alpha) = \begin{cases} \Delta^{T_u}(v_i) & \textit{leaf}(v_i)\textit{=true} \\ \rho^{T_u}(v_i, \alpha) & \textit{root}(v_i)\textit{=true} \\ \Phi^{T_u}(v_i, \alpha) & \text{Otherwise} \end{cases} \quad (4)$$

  This function considers nodes in a multi-tree in three categories: leaves, root and nodes situated between leaves and the root whose weights are calculated by functions Δ, ρ and Φ respectively.
- Δ is a function mapping $V \rightarrow \{0, 1\}$. This function determines whether a specific node, v_i, in a combined tree T_u exists in both of the trees from which it is constituted (T_1, T_2).

$$\Delta^{T_u}(v_i) = \begin{cases} 1 & \text{if } v_i \in V^{T_1}, v_i \in V^{T_2} \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

- ρ is a function mapping $V \times \mathbb{R} \rightarrow \mathbb{R}$. This function is used to calculate the weights of the nodes in a multi-tree.

$$\rho^{T_u}(v_i, \alpha) = (\frac{1}{\beta_{v_i}})(\sum_{\forall v_x \in children(v_i)} P(v_i, v_x) W^{T_u}(v_x, \alpha)) \quad (6)$$

- Φ is a function mapping $V \times \mathbb{R} \rightarrow \mathbb{R}$. This function returns the weight of a node if the node is neither a leaf node nor the root of the multi-tree.

$$\Phi^{T_u}(v_i, \alpha) = (1 - \frac{1}{\alpha^{L(v_i)+1}}) \rho^{T_u}(v_i, \alpha) \quad (7)$$
$$+ (\frac{1}{\alpha^{L(v_i)+1}}) \Delta^{T_u}(v_i)$$

  The Φ function calculates the weight of a node by $1 - \frac{1}{\alpha^{L(v_i)}}$ share of the average of the weight of its children which calls the function to calculate the weight of each child recursively and $\frac{1}{\alpha^{L(v_i)}}$ share for the commonality of a node between the multi-trees of two concepts.

The above description is best illustrated by the following example. The example, shown in Figure 1,2 and the calculation using equations 5-7 illustrated in Figure 3, demonstrates how the above functions work for two entities represented by their features (i.e., products appeared in the profiles of two users).
d_1=(Web Cam, Digital Camera, LCD TV, Blender, Mattress)
d_2=(Keyboard, DSLR Camera, LED TV, Mattress, Drawer)

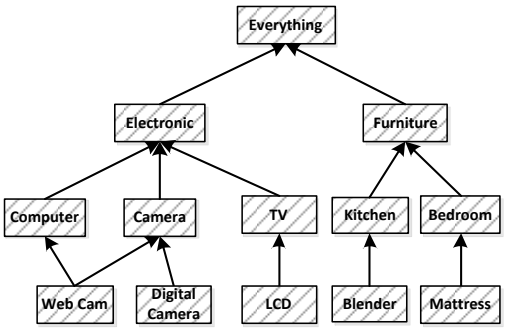


Figure 1: First multi-tree representing transaction d_1

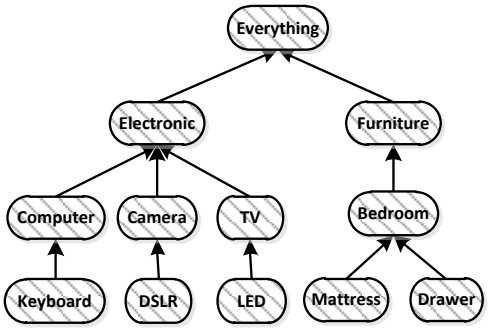


Figure 2: Second multi-tree representing transaction d_2

Using a VSM, the similarity between d_1, d_2 is equal to 0.2. However, using the proposed method, although LED and LCD are not equal they have a parent in common which makes the weight non-zero. Considering $\alpha = e = 2.71$ (also used in experiments reported later), the similarity between d_1, d_2 is: 0.444.
W(Keyboard)=0, W(Web Cam)=0, W(Digital Camera)=0, W(DSLR)=0, W(LED)=0, W(LCD)=0, W(Blender)=0, W(Drawer)=0, W(Mattress)=1, W(Kitchen)=0
W(Computer)=W(TV)=W(Camera)=$(1 - \frac{1}{e})(0) + (\frac{1}{e}) = 0.369$
W(Bed room)=$(\frac{1}{2}) \times (1 - \frac{1}{e}) + (\frac{1}{e}) = 0.684$
W(Electronic)=$(\frac{1}{e}) \times (1 - \frac{1}{e^2}) + (\frac{1}{e^2}) = 0.457$

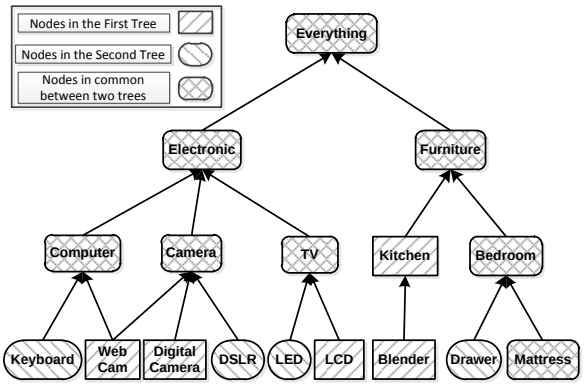


Figure 3: A multi-tree combined from previous two multi-trees

W(Furniture)=$(\frac{0+0.684}{2}) \times (1-\frac{1}{e^2}) + (\frac{1}{e^2}) = 0.431$
W(Everything)=0.444

4 Similarity using a Semantic Multi-Tree

In order to calculate the similarity between two entities, we construct two multi-trees each of which represents features of the corresponding entity in a hierarchical format according to a specific ontology or taxonomy. In this method, each entity is represented by its features as leaves of a multi-tree. The rest of each multi-tree is constructed according to the domain taxonomy (e.g., WordNet or Wikipedia Categories). Hence, the multi-tree corresponding to each entity is a sub multi-tree of the taxonomy with which the sub-multi-tree is constructed. A multi-tree T_x is said to be a sub multi-tree of $T_\mathcal{O}$ if:

$$T_x \subseteq T_\mathcal{O} \Leftrightarrow \begin{cases} V_x \subseteq V_\mathcal{O} \\ E_x \subseteq E_\mathcal{O} \\ C_x \subseteq C_\mathcal{O} \end{cases} \quad (8)$$

Assume that, $T_\mathcal{O} = (V_\mathcal{O}, E_\mathcal{O}, C_\mathcal{O}, L_\mathcal{O}, M_\mathcal{O}, W_\mathcal{O}, P_\mathcal{O})$, is a multi-tree representing the domain taxonomy (e.g., Wikipedia Categories), $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$, in which $\mathcal{C}_\mathcal{O}$ represents set of concepts and $\mathcal{R}_\mathcal{O}$ represents set of relations among concepts in the taxonomy. The transformation function $\mathcal{T}$ is defined as a bijective function $\mathcal{T} : \mathcal{R}_\mathcal{O} \rightarrow E$ which maps each relation in the taxonomy $\mathcal{O}$ to an edge in the multi-tree $T_\mathcal{O}$. So, $E_x = \{e_i = \mathcal{T}(R_i) \mid R_i \in \mathcal{R}_\mathcal{O}\}$. ($C_\mathcal{O} \equiv \mathcal{C}_\mathcal{O}$ and $E_\mathcal{O} \equiv \mathcal{R}_\mathcal{O}$).

The multi-tree, $T_x = (V_x, E_x, C_x, L_x, M_x, W_x, P_x) \subseteq T_\mathcal{O}$, corresponds to entity $d_x = (c_1, ..., c_n)$ in which c_i is a term representing a feature of this entity as well as a concept in the taxonomy. We define $C_x \subseteq C_\mathcal{O}$ in multi-tree T_x as a set of terms representing features of the entity d_x.

Hence, $C_x = \{c_1, ..., c_n\} \cup \{c_j \mid \forall c_i \in C_x, \forall c_j \in \mathcal{C}_\mathcal{O}, \textit{is-a}(c_i, c_j) \in \mathcal{R}_\mathcal{O}\}$, $V_x = \{v_i = M(t_i) \mid t_i \in C_x\}$ and $E_x = \{e_i = \mathcal{T}(R_i) \mid \forall c_k, c_l \in C_x, R_i(c_k, c_l) \in R_\mathcal{O}\}$ such that $C_x \subseteq \mathcal{C}_\mathcal{O}$ and $\mathcal{O} = (\mathcal{C}_\mathcal{O}, \mathcal{R}_\mathcal{O})$ is the taxonomy which is used to construct the tree.

In the next step, the combination operator is applied to the two trees whose similarity is being computed. Applying the combination operator to the two trees, the weight of the root of the combined tree represents the similarity of the two trees. The weight of the root is recursively calculated by application of equations 5-7. The following steps demonstrate the process of calculating the similarity between two entities d_1, d_2 represented by sets of features C_1, C_2 respectively:

1. Construct multi-trees T_1 and T_2 from sets of features C_1 and C_2 respectively.
2. Construct $T_{sim} = T_1 \uplus T_2$ as a combination of two multi-trees $T_1, T_2 \subseteq T_\mathcal{O}$
3. Update the weights for the nodes in the combined multi-tree T_{sim} using the recursive equations 5-7.
4. The weight of the root of T_{sim} is the value which represents the similarity of two entities represented by C_1, C_2; i.e., $Sim(d_1, d_2) = W(root(T_{sim}))$.

Algorithm 1 and 2 describes the process by which a multi-tree is constructed from a feature set representing an entity.

Algorithm 1 Constructing a multi-tree.

Proc ConstructMulti-Tree(ConceptSet C_x)
$T_x \leftarrow null$
for all c in C_x **do**
  FindPaths($T_\mathcal{O}.M^{-1}(c), T_\mathcal{O}, T_x$)
end for
return T_x

Algorithm 2 Finding a path in a multi-tree from a leaf node to root.

Proc FindPaths(Node v,Multi-Tree $T_\mathcal{O}$,Multi-Tree T_x)
if root(v) **then**
  $T_x.root \leftarrow v$
  return
end if
for all parent in $T_\mathcal{O}$.Parents(v) **do**
  if parent not in $T_x.V$ **then**
    $T_x.V \leftarrow T_x.V \cup parent$
    $T_x.E \leftarrow T_x.E \cup < parent, v >$
    FindPaths(parent,$T_\mathcal{O}, T_x$)
  end if{avoid cycles}
end for

Algorithm 3 describes the calculation of similarity using the proposed non-linear method.

Algorithm 3 Calculation of similarity using multi-trees.

$T_1 \leftarrow$**ConstructMulti-Tree**(ConceptSet C_1)
$T_2 \leftarrow$**ConstructMulti-Tree**(ConceptSet C_2)
$T_{Sim} \leftarrow T_1 \uplus T_2$
$similarity \leftarrow W^{T_{Sim}}(root, \alpha)$

5 Experimental Results

The proposed model is not only useful for measuring similarity between pairs of words but is also useful for information retrieval and recommender systems. In this paper, we evaluated the semantic multi-tree model for the application of measuring similarity between pairs of words, but the domain of the proposed model is not limited just to the application of measuring similarity between pairs of words. Our rationale for doing this is that the concept domain is potentially larger as we are not limited to (say) movies, music or books, domains often used in recommender datasets.

One of the methods for evaluating psycholinguistic systems is comparing the results with human judgement. Among three standard datasets that exist in the domain of measuring semantic relatedness between pairs of words, WordSimilarity-353 is the most comprehensive dataset since this dataset includes all of the 30 nouns of the Miller and Charles dataset and most of the 65 pairs of Rubenstein and Goodenough testset [Rubenstein and Goodenough, 1965]. The WordSimilarity-353 dataset contains 353 pairs of words compared by human agents in terms of similarity [Finkelstein *et al.*, 2002]. This system has been evaluated by both WordSimilarity-353 and Rubenstein and Goodenough testsets. While the Charles and Miller testset is not available on the web and is already included in WordSimilarity-353, we are not able to give statistics about the performance of proposed method on this dataset.

Wikipedia is a resource of concepts linked to each other forming a network, which is collaboratively constructed by human agents around the world. Each page in Wikipedia is linked to a set of categories classifying concepts and Wikipedia pages. Each page in Wikipedia describes one of the concepts associated to a word. A Wikipedia page describes a concept using other concepts described in other pages. In order to evaluate the proposed model to estimate the similarity between two entities, we used Wikipedia as a resource of knowledge and Wikipedia Categories as a taxonomy of concepts with which Wikipedia pages are annotated. The existing links in a Wikipedia page are considered as features describing the associated concept. Figure 4 illustrates the Wikipedia link structure data model. In this figure, each circle represents a wikipedia category and each square represents a Wikipedia page corresponding to a concept. Solid lines represent links between pages and a dotted line represents a link between a page and categories it belongs to.

In this experiment, The VSM is compared to the multi-tree model described previously against human judgement in terms of accuracy and correlation. For this purpose, each word is mapped to a Wikipedia page, and then a vector containing the links of the pages to which the corresponding page is linked is constructed and is referred to as a *link vector*. In Figure 4, the page L is linked to $\{N, O, P\}$ which are considered as features of the link vector $C_L = (N, O, P)$. Each link in the link vector represents another page in Wikipedia which is linked to Wikipedia categories. In the next step, another vector is constructed according to the categories of a link vector's elements (i.e., leaf nodes in Categories) called a first order category vector. In Figure 4, $\{N, O, P\}$ are

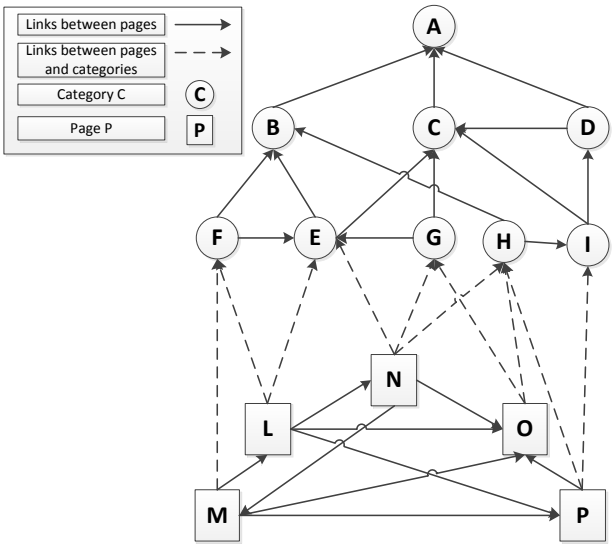


Figure 4: The Wikipedia link structure data model

linked to categories $\{E, G, H, I\}$. Then, the categories of the main page, L, (i.e., $\{E, F\}$) are added to the first order category vector and then the multi-tree representing the concept L is constructed from the first order category vector ($C_L^{1st} = (E, F, G, H, I)$). The rest of the process for construction of the multi-tree model is the same as described in Sections 3 and 4. This process is pictorially represented in Figure 5.

In VSM, the link vectors are compared using cosine similarity as described in equation 1. Both VSM schemes have been evaluated against human judgement with the same dataset and the results are compared in Table 1. Since, in this paper, we are not using a corpus-based approach, and in each experiment only two vectors representing two concepts are compared, the inverse document frequency of each link can not be calculated. Therefore, tf was used instead of tf-idf to implement the second VSM scheme.

The average accuracy in Table 1 is measured regarding the difference between the value of similarity measured by the techniques tested above and that of human judgement.

$$Accuracy = 1 - Average(error_i) \tag{9}$$

$$error_i = Sim_{Human}(w_{i1}, w_{i2}) - Sim_{Computer}(w_{i1}, w_{i2}) \tag{10}$$

The correlation was estimated by Spearman's rank correlation coefficient. The correlation between human judgement and the multi-tree model is demonstrated in Figure 6. Table 1 demonstrates that the proposed model achieved better results than both VSM schemes in terms of accuracy and correlation with human judgment in estimating similarity between entities.

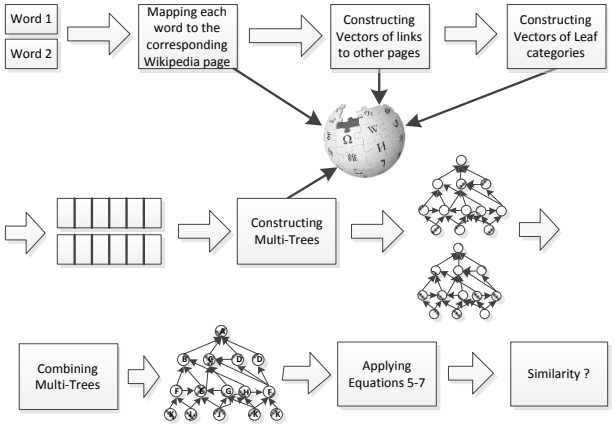


Figure 5: The Architecture of the proposed system

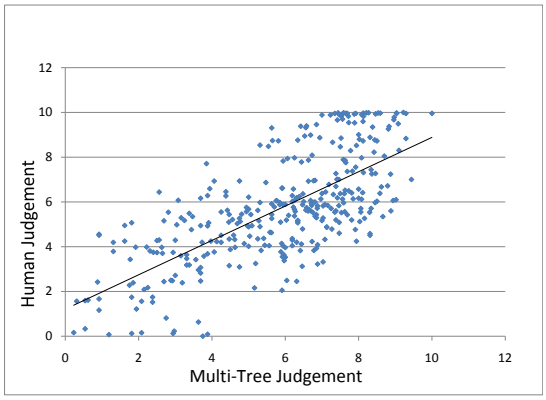


Figure 6: Human Judgement vs. Multi-Tree similarity algorithm on WordSimilarity-353 dataset

6 Discussion, Conclusions and Future Work

In this paper we observed that techniques such as linear VSM ignore the semantic relationships among features. VSM calculates the similarity between two documents regarding the commonality of their features. However, in some cases, two documents may not be equal but may refer to the same entity. This limits the capability of a VSM to retrieve related documents. The multi-tree model compensates for the lack of semantic relatedness among features using taxonomic relations that exist among the features of two entities. In this model the similarity weight is propagated from leaf nodes to the root of the multi-tree. The multi-tree model was evaluated by using the WordSimilarity-353 and Rubenstein and Goodenough datasets against human judgement and the results show that the multi-tree method outperforms VSM in terms of correlation and the average accuracy against human judgement for similarity of pairs of words. The results for WikiRelate and WLM, two previous systems which used Wikipedia Categories or WordNet to perform the task of similarity between two words using the same dataset, are shown in Table 2. WikiRelate and WLM were briefly described in Section 2.

	Average Accuracy compared to human judgement	Correlation with Human judgement
WordSim-353		
VSM Boolean	57.1	54
VSM Frequency of terms	59.2	56.9
Multi-Tree	84.7	70.6
Rubenstein and Goodenough		
VSM Boolean	61.9	57.1
VSM Frequency of terms	62.2	60
Multi-Tree	80	74

Table 1: The comparison between VSM and Multi-Tree model using two testsets.

The results in Table 2 show that the method proposed in this paper outperforms two of the other competitors namely WikiRelate and WLM by more than 20% and 3% respectively. However, ESA is the first ranked system using machine learning techniques as the basis of a semantic interpreter that is part of a system that maps fragments of natural language text into a weighted sequence of Wikipedia concepts ordered by their relevance to the input. Regarding ESA, it is clear that augmenting the text representation and constructing a weighted list of concepts provides benefits that link analysis alone does not completely replace.

Dataset	WikiRelate	ESA	WLM	Multi-Tree
Goodenough	52	82	64	74
WordSim-353	49	75	69	71
W-average	49	76	68	71

Table 2: Performance of semantic relatedness measures for two standard datasets for three popular systems vs. Multi-tree model.

Therefore, while the multi-tree model shows promise as a means by which semantically similar entities can be found, there are various ways that we can extend this approach. For instance, in this model we ignored the number of occurrences of features in each multi-tree as initial weights for leaves and used a binary scheme to calculate node weight.

Another potential model extension is in using a more sophisticated function such as Pearsons product-moment correlation or cosine similarity instead of the simple average function in equation 6.

A potential application of this model is in recommender systems, which concentrate on similarity between two products or two people. Referring once again to the example described in Section 3 and shown graphically in Figures 1, 2 and 3, the similarity of buying patterns can be established

using a semantic multi-tree approach. For the evaluation of such systems, we need to construct a handcrafted taxonomy of products plus annotation of the product dataset to the taxonomy of products. Keyword search engines are also another potential application of such systems instead of linear VSM.

References

[Annates, 2005] U. Annates. Semantic tree method-historical perspective and applications Izabela Bondecka-Krzykowska. *Annales Universitatis Mariae Curie-Skłodowska: Informatica*, page 15, 2005.

[Finkelstein *et al.*, 2002] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM Transactions on Information Systems (TOIS)*, 20(1):116–131, 2002.

[Gabrilovich and Markovitch, 2007] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 6–12, 2007.

[Landauer *et al.*, 1998] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2):259–284, 1998.

[Larkey and Markman, 2005] L.B. Larkey and A.B. Markman. Processes of similarity judgment. *Cognitive Science: A Multidisciplinary Journal*, 29(6):1061–1076, 2005.

[Pedersen *et al.*, 2005] S.P.T. Pedersen, S. Banerjee, and S. Patwardhan. Maximizing semantic relatedness to perform word sense disambiguation, 2005. *University of Minnesota Supercomputing Institute Research Report UMSI*, 25, 2005.

[Rubenstein and Goodenough, 1965] H. Rubenstein and J.B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, 1965.

[Salton and McGill, 1983] G. Salton and M.J. McGill. Introduction to modern information retrieval. *New York*, 1983.

[Salton *et al.*, 1975] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[Strube and Ponzetto, 2006] M. Strube and S.P. Ponzetto. WikiRelate! Computing semantic relatedness using Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1419. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[Tversky and Shafir, 2004] A. Tversky and E. Shafir. *Preference, belief, and similarity: selected writings*. The MIT Press, 2004.

[Witten and Milne, 2008] I.H. Witten and D. Milne. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30, 2008.

Recommending Information Sources to Information Seekers in Twitter

Marcelo G. Armentano, Daniela Godoy, and Analía Amandi
ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA
Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{marmenta, dgodoy, amandi}@exa.unicen.edu.ar

Abstract

Finding high-quality sources in the expanding micro-blogging community using Twitter becomes essential for information seekers in order to cope with information overload. In this paper, we present a recommendation algorithm aiming to identify potentially interesting users to follow in the Twitter network. This algorithm first explores the graph of connections starting at the target user (the user to whom we wish to recommend previously unknown followees) in order to select a set of candidate users to recommend, according to an heuristic procedure. The set of candidate users is then ranked according to the similarity between the content of tweets that they publish and the target user interests. Experimental evaluation was conducted to determine the impact of different profiling strategies.

1 Introduction

Micro-blogging activity taking place in sites such as Twitter is becoming every day more important as real-time information source and news spreading medium. In the followers/followees social structure defined in Twitter a follower will receive all the micro-blogs from the users he follows, known as followees, even though they do not necessarily follow him back. In turn, re-tweeting allows users to spread information beyond the followers of the user that post the tweet in the first place.

Recent research efforts on understanding micro-blogging as a novel form of communication [Java *et al.*, 2007; Krishnamurthy *et al.*, 2008] revealed that few users in Twitter maintain reciprocal relationships with other users. This fact differentiates Twitter from other online social networks, such as Facebook, Hi5, or Orkut, in which people mainly make connections to keep in touch with people they consider as friends or acquaintances.

Although posts in Twitter or *tweets* are allowed to have any textual content within the limit of 140 characters, many users only publish information about a particular subject, such as sports, movies, music or about a particular rock band. These users can be considered as information sources or broadcasters. In contrast, many people use twitter to get information on particular subject, as a form of RSS reader, registering themselves as followers of their favorite artists, celebrities, bloggers, or TV programs. Users acting as information sources are characterized by having a larger number of followers than followees, as they are actually posting useful information or news. Information seekers, on the other hand, subscribe to this kind of users but rarely post tweets and, finally, friends are users exhibiting reciprocal relationships. With information seekers being an important portion of registered users in the system, finding relevant and reliable sources in the constantly increasing Twitter community[1] becomes a challenging issue.

To address this problem, we propose a followee recommender system that, according to an heuristic procedure, explores the topology of followers/followees network of Twitter to find candidate users to recommend and then these candidate users are ranked according to their similarity with the target user's interests. Three profiling strategies are analyzed and evaluated for modeling users' interests in Twitter based on two general approaches. The first approach models a user by analyzing the content of his/her own tweets whereas the second approach represents users by the tweets of their followees. For the second approach, two different types of profiles were considered: modeling a target user by the set of profiles of his/her followees, and by a set categories that can be discovered by clustering his/her followees according to the content of their tweets.

Unlike other works that focus on ranking users according to their influence in the entire network [Weng *et al.*, 2010; Yamaguchi *et al.*, 2010], the algorithm we propose explores the follower/followee relationships of the user up to a certain level, so that only the neighborhood of the target user is explored in the search of candidate recommendations. The influence rankings presented by studies on the complete Twittersphere have no direct utility for followee recommendation since people who are popular in Twitter would not necessarily match a particular user's interests. For example, if a user follows accounts talking about technology, he/she would not be interest in Ashton Kutcher, one of the most influential Twitter accounts according to [Kwak *et al.*, 2010].

[1] In 2010 Twitter grew by more than 100 million registered accounts (`http://yearinreview.twitter.com/whosnew/`. Accessed on March 2011)

In this article we study Twitter from a user modeling perspective. Our goal is to provide recommendations to information seekers about users who publish tweets that might be of their interest. Unlike traditional recommendation systems, we do not have any explicit information available about the user's interests in the form of ratings on items he/she likes or dislikes. The only information available for profiling a Twitter user is the structure of the followers/followees network and the tweets published in this network. Both of these elements are considered in this paper as a mean to recommend people who share the same content-related interests with the user who will receive the recommendations.

The rest of this work is organized as follows. Section 2 discusses how related work is related to our research. Section 3 describes the content-based approach to the problem of followee recommendation for helping information-seeking users in Twitter. In Section 4 experiments carried out to validate the approach using a Twitter dataset are reported. Finally, Section 5 discusses the results obtained and presents our conclusions and future work avenues.

2 Related Work

The problem of helping users to find and to connect with people on-line to take advantage of their friend relationships has been studied in the context of traditional social networks. For example, SONAR [Guy *et al.*, 2009] recommends related people in the context of enterprises by aggregating information about relationships as reflected in different sources within a organization, such as organizational chart relationships, co-authorship of papers, patents, projects and others. Liben-Nowell et al. [Liben-Nowell and Kleinberg, 2003] presented different methods for link prediction based on node neighborhoods and on the ensemble of all paths. These methods were evaluated using co-authorship networks obtained from the author lists of papers at five sections of the physics e-Print arXiv[2]. Authors found that there is indeed useful information contained in the network topology alone. Chen et al. [Chen *et al.*, 2009] compared relationship-based and content-based algorithms in making people recommendations, finding that the first ones are better at finding known contacts whereas the second ones are stronger at discovering new friends. Weighted minimum-message ratio (WMR) [Lo and Lin, 2006] is a graph-based algorithm which generates a personalized list of friends in a social network built according to the observed interaction among members. Unlike these algorithms that gathered social networks in enclosed domains from structured data (such as interactions, co-authorship relations, etc.), we face the problem of taking advantage of the massive, unstructured, dynamic and inherently noisy user-generated content from Twitter for recommendation.

Several studies dedicated to understand micro-blogging as a novel form of communication and news spreading medium have been recently published. Some of these research efforts have been dedicated to study the structure of Twitter network and its community structure. Java et al. [Java *et al.*, 2007] presented a characterization of Twitter users identifying three kinds of users:

- "Information Sources" are users who are characterized by having a much larger number of followers than they themselves are following.
- "Friends" are users who trend to use Twitter as a typical online social network and are characterized by reciprocity in their relationships.
- "Information Seekers" are users who rarely post a tweet authored by himself, but that regularly follows other users

In a posterior study presented by Krishnamurthy et al. [Krishnamurthy *et al.*, 2008] also three categories of users were identified. The first category is called "Broadcasters of tweets" and corresponds to the "Information Sources" category above. The second category, "Acquaintances", is equivalent to "Friends" category identified by Java et al. However, a different interpretation is given to the third category of users. Krishnamurthy et al. call users who follow lots of other users but that are followed by few users "Miscreant / Evangelists". They consider that users in this category are usually spammers or stalkers that contact lots of users expecting to be followed by them.

Kwak et al. [Kwak *et al.*, 2010] quantified these findings indicating that 77.9% of Twitter connections are unidirectional and only 22.1% of the relations are reciprocate. Moreover, 67.6% of users are not followed by any of their followees, indicating that these users probably use Twitter as a source of information rather than as a social networking site.

Other line of research has been devoted to measure the influence of users in Twitter. In [Kwak *et al.*, 2010] it was shown that ranking users by the number of followers and by their PageRank give similar results. However, ranking users by the number of re-tweets indicates a gap between influence inferred from the number of followers and that from the popularity of user tweets. Coincidentally, a comparison between in-degree, re-tweets and mentions as influence indicators [Cha *et al.*, 2010] concluded that the first is more related to user popularity. Analyzing spawning re-tweets and mentions, it was found that most influential users hold significant influence over a variety of topics but this influence is gained only through a concentrated effort (such as limiting tweets to a single topic). TwitterRank [Weng *et al.*, 2010], an extension of PageRank algorithm, tries to find influential twitterers by taking into account the topical similarity between users as well as the link structure. Garcia et al. [Garcia and Amatriain, 2010] propose a method to weigh popularity and activity of links for ranking users. User recommendation, however, can not be based exclusively on general influence rankings since people get connected for multiple reasons.

While the studies mentioned above focused on the analysis micro-blogging usage, other works try to capitalize the massive amount of user-generated content as a novel source of preference and profiling information for recommendation. Chen et al. [Chen *et al.*, 2010] proposed an approach to recommend interesting URLs coming from information streams such as tweets based on two topic interest models of the target user and a social voting mechanism. For each user two models are used: a Self-profile built with the words of the user tweets and a Followee-profile built by combining the

[2] http://www.arxiv.org

self-profiles of the user followees. Thus, a set of candidate pages posted by a user followees and followees of followees is filtered according to these models. In the social scheme filtering is based on a voting system within a user followee-of-followees neighborhood so that the most popular URLs within the group are recommended. *Buzzer* [Phelan *et al.*, 2009] indexes tweets and recent news appearing in user specified feeds, which are considered as examples of user preferences, to be matched against tweets from the public timeline or from the user Twitter friends for story ranking and recommendation. Esparza et al. [Esparza *et al.*, 2010] address the problem of using real-time opinions of movie fans expressed through the Twitter-like short textual reviews for recommendation. The work by Esparza et al. assumes that tweets contain preference-like information that can be used in content-based and collaborative filtering recommendation. Opinion mining and sentiment analysis applied to tweets are starting to be considered to replace explicit ratings required by traditional recommendation technologies [Pak and Paroubek, 2010; Davidov *et al.*, 2010].

Continuing in this direction, Naaman et al. [Naaman *et al.*, 2010] classify users into "informers" and "meformers". According to this work, "informers" users publish tweets containing mainly non-personal information while"meformers" users mainly post status updates about themselves and their daily routines. Ramage et al. [Ramage *et al.*, 2010] go a step forward using a partially supervised learning model that maps the content of tweets into different dimensions that correspond to substance, style, status and social characteristics of posts. "Substance" tweets contain information about events, ideas, things or people; "social" tweets relate to some socially communicative end; "status" tweets refer to personal updates; finally "style" tweets are those indicative of broader trends of language use. Perez-Tellez et al. [Perez-Tellez *et al.*, 2010] categorize tweets which contain a company name into two clusters corresponding to those which refer to the company and those which do not. They use a text enrichment technique, called Self-Term Expansion Methodology (S-TEM), aiming at improving the quality of the corpora. Several variations of this technique are presented and compared, such as enhancing S-TEM by considering additional information extracted from Wikipedia.

In contrast to the previous works that address the problem of suggesting potentially relevant content from micro-blogging services, we concentrate in recommending interesting people to follow. In this direction, Sun et al. [Sun *et al.*, 2009] proposes a diffusion-based micro-blogging recommendation framework which identifies a small number of users playing the role of news reporters and recommends them to information seekers during emergency events. Closest to our work are the algorithms for recommending followees in Twitter evaluated and compared in [Hannon *et al.*, 2010] using a subset of users. Multiple profiling strategies were considered according to how users are represented in a content-based approach (by their own tweets, by the tweets of their followees, by the tweets of their followers, by the combination of the three), a collaborative filtering approach (by the IDs of their followees, by the IDs of their followers or a combination of the two) and two hybrid approaches. User profiles are indexed and recommendations generated using a search engine, receiving a ranked-list of relevant Twitter users based on a target user profile or a specific set of query terms. Our work differs from this approach in that we do not require indexing profiles from Twitter users. Instead, a topology-based algorithm is used to explore the follower/followee network in order to find candidate users to recommend and a content-based analysis is then applied to generate the ranked list of recommendations.

3 Followees Recommendations in Twitter

The problem of followee recommendation in Twitter consists in identifying users posting relevant tweets for a target user, so that he/she can subscribe to these users and start receiving real-time information from them. The approach presented in this work can be decomposed into three main parts. First, we create the target user's profile which describes his/her interests or information needs. Second, we search for a suitable group of candidate users to be considered for recommendation and determine whether the information that they publish may be of interest to the target user. Finally, we rank these users and present the top-ranked users as followee recommendations. These parts of our approach are described in the following sections. Section 3.1 describes different strategies for building user profiles in order to describe a user's interests. Next, Section 3.2 describes the search for candidates based on the topology of the Twitter network. Finally, Section 3.3 explains how profiles are compared in order to determine which set of users recommend to the target user.

3.1 Content-based User Profiles

In our approach, the user profile for a target user u_T will model the information he/she likes to read, whereas for a candidate user u_C the user profile will model the information he/she publishes. For any user u, let $tweets\,(u)$ be the set of all his/her posts:

$$tweets\,(u) = \{t_1, \ldots, t_k\} \tag{1}$$

The interests of a target user can then be described using different content sources, such as the text of his/her own tweets or the content of the tweets published by his/her followees. The different strategies we used to create the target user's profiles are described in the following sections.

Profile Strategy T0

The simplest alternative to build a profile for a user in Twitter is to aggregate his/her own tweets under the assumption that users are likely to tweet about things that are of interest to them:

$$Profile^{T0}\,(u_T) = \sum_{i=1}^{k} t_i \tag{2}$$

The profile of a user is then a vector in which terms are weighted according to their frequency of occurrence in the text of the user tweets. Tweets are processed to obtain the vector of a given user posts, $Profile^{T0}\,(u)$, by applying a number of filters in a pipeline. First, tokens only composed of punctuation symbols are assumed to be emoticons and are

then removed. Second, common slang vocabulary and abbreviations are substituted. This kind of words are widely used in Twitter messages to overcome the limitation in the number of characters. The NoSlang on-line dictionary[3], containing 5,227 entries, was used to this end. In this step abbreviations are replaced with the corresponding complete words or phrases, for example "idn" is replaced by "i don't know" or "ntta" by "nothing to talk about". Finally, stop-words are removed and Porter stemming algorithm [Porter, 1980] is applied to the remaining words.

Profile Strategy T1

Information seekers are characterized by posting few tweets themselves, but they follow people who generate content more actively. Hence, as followee recommendation is oriented toward information seekers, an alternative method to model the interests of a user is based on who is he/she following, this is, which information the user wants to read about.

It is assumed that users select their followees expecting that their tweets will be of interest to them. Thus, a second type of profile is built based on the observation that a user has a number of followees:

$$followees\left(u_T\right) = \{f_1, \ldots, f_l\} \tag{3}$$

and information a user is interested in can be obtained from the profiles of his/her followees.

However, users might follow people twitting about different subjects. For example, a user may follow celebrities, politicians, sportsmen and other type of users. As a result, considering all followees as responding to a unique topic of interest is not enough to effectively model multiple user interests in diverse areas. Consequently, rather than creating a single vector representing all of the user's interests, this strategy creates multiple vectors, each of them representing a different followee. This profile strategy allows us to attain fine-grained profiles. The profile of a user is then defined as the set of the profiles of the user followees, each modeling a followee own tweets:

$$Profile^{T1}\left(u_T\right) = \left\{Profile^{T0}\left(f_1\right), \ldots, Profile^{T0}\left(f_l\right)\right\} \tag{4}$$

Profile Strategy T2

In a more realistic view of a user information preferences, it can be assumed that users are likely to follow people in different interest categories. For example, a user can be following some Twitter users because they talk about his/her favorite sport and others according to her/his political opinions. Hence, to assess a more precise description of the user interests a last type of profile tries to group the user followers into meaningful categories.

Coarser-grained profiles are created using a simple clustering algorithm detailed in Algorithm 1. The identification of categories to which a user's followees belong need to be incrementally discovered starting from scratch as the user starts following a new user in Twitter. In this clustering approach, as soon as the user subscribes to a followee it is assigned to the first cluster or category in the user profile. Each subsequent followee is incorporated into either some of the existent categories or to a novel category depending on its similarity with the current categories. Hence, a user's interest categories are extensionally defined in the user profile by highly similar followees that conform clusters. This partition reduces the total number of vectors representing all followees to a relatively smaller number of clusters, which can be further analyzed to discover topicality.

[3] http://www.noslang.com/dictionary

Algorithm 1 Incremental clustering algorithm

Input: The vector profiles, $Profile^{T0}(f)$, of all $f \in followees\left(u\right)$ of user u and a similarity threshold δ

Ouput: The profile of u grouping the followees in a set of followee categories $FC_u = \{fc_1, \ldots, fc_m\}$

INCREMENTALCLUSTERING

1: $FC_u \leftarrow \emptyset$ */*Create an empty profile for u*/*
2: $Q \leftarrow \emptyset$ */*Initialize a set to contain the clusters the new followee is similar to*/*
3: **for all** f_i such that $f_i \in followees\left(u\right)$ **do**
4: **for all** fc_j such that $fc_j \in FC_u$ **do**
5: Let c_j be the centroid of fc_j
6: $sim_j \leftarrow sim\left(c_j, fc_j\right)$
7: **if** $sim_j \geq \delta$ **then**
8: $Q \leftarrow add\left(\langle fc_j, sim_j\rangle\right)$
9: **end if**
10: **end for**
11: **if** $(Q \neq \emptyset)$ **then**
12: Sort instances in Q by decreasing order of sim_j
13: Let fc_k be the first cluster in Q
14: Include the followee f_i into fc_k */*The centroid vector of the cluster is updated*/*
15: **else**
16: Create an empty cluster fc_{new}
17: Include the followee f_i into fc_{new}
18: **end if**
19: **end for**
20: Return FC_u

The clustering algorithm returns a set of categories $FC_u = \{fc_1, \ldots, fc_m\}$ the current followees of the user u can be grouped into. Given the cluster or followee category fc_i , which is composed of the set of followees and their corresponding vector representations, the centroid vector c_{fc_i} is

$$c_{fc_i} = \frac{1}{|fc_i|} \sum_{f \in fc_i} Profile^{T0}\left(f\right) \tag{5}$$

Each time the user starts following another user, the new followee vector is incorporated to the current user profile within the most similar existing cluster. In order to predict which this cluster is, the closest centroid is determined by comparing the vector $Profile^{T0}\left(f_{new}\right)$ of the new followee with all centroids in the existing clusters. This similarity measure determines the degree of resemblance between the vector representations and is calculated by the cosine similarity. As the result of vector comparison, the new followee f_{new} is assigned to the cluster with the closest centroid, i.e.

$$\arg \max_{j=1...k} sim\left(f_{new}, c_{fc_j}\right)$$

provided that the similarity is higher than a minimum similarity threshold δ. Vectors not similar enough to any existent centroid according to this threshold cause the creation of new singleton clusters.

In summary, two general approaches are evaluated in this paper for modeling a user's interests in Twitter according to if the user own tweets or the tweets of their followees are used to glean a profile. For the last approach, two different mechanisms to combine the vectors of the user followees were analyzed. The first consists in modeling a target user using a set of vectors, each of them representing the content of a user followee tweets. The second profile models a target user by a set of vectors corresponding to the centroids obtained after applying a clustering algorithm to the vectors representing the target user followee tweets.

3.2 Topology-Based Candidate Search

In order to recommend Twitter users, a set of viable candidates need to be first identified within the follower/followee network. The method employed to explore the Twitter network with the goal of gathering candidate users for recommending to a target user u_T is based on the following hypothesis: the users followed by the followers of u_T followees are possible candidates to recommend to u_T. In other words, if a user u_F follows a user that is also followed by u_T, then other people followed by u_F can be of interest to u_T.

The rationale behind this hypothesis is that the target user is an information seeker that has already identified some interesting users acting as information sources, which are his/her followees. Other people who also follow some users in this group (i.e. are subscribed to some of the same information sources) have interests in common with the target user and might have discovered other relevant information sources in the same topics, which are in turn their followees. Figure 1 illustrates this approach for candidate selection schematically.

More formally, the search of candidate users for recommendations is performed according to the following steps:

1. Starting with the target user u_T, obtain the list of users he/she follows, let's call this list $S = \bigcup_{\forall x \in followees(u_T)} x$.
2. For each element in S get its followers, let's call the union of all these lists L, i.e. $L = \bigcup_{\forall s \in S} followers\,(s)$.
3. For each element in L obtain its followees, let's call the union of all these lists T, i.e. $T = \bigcup_{\forall l \in L} followees\,(l)$.
4. Exclude from T those users who the target user is already following. Let's call the resulting list of candidates $R = T - S$.

Each element in R is a possible user to recommend to the target user as future followee. To relate to the previous hypothesis the group S will be mostly composed of information sources, L will be other users looking for information in the same way that u_T does and T will be further information sources. Users can appear more than once in R, depending on the number of times that they appear in the lists of followees or followers obtained at steps 2 and 3 above, this is a factor that can be later consider to boost its chances of being recommended.

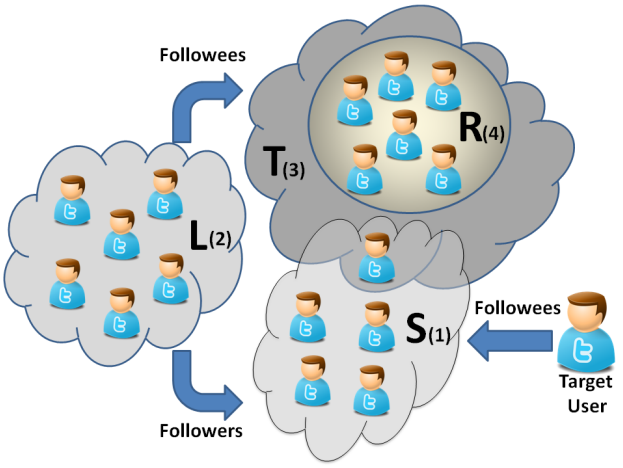


Figure 1: Strategy for exploring the followee/follower network to find candidate users

It is worth noticing that other strategies can be elaborated or combined with the search based on the topology of the network described above to include in the evaluation users who are not in the proximity of the target user. For example, candidate users can be taken from Twitter's *public timeline*. The public timeline is an information stream that contains the collection of the most recently published tweets and it is fed by all accounts that are not configured to be private. The public timeline can be considered as the current flow of information in Twitter and it is a good source to obtain active users in the social network.

3.3 Comparing User Profiles

Once a list of viable candidates R is available, the matching between the information each user $r \in R$ publishes in Twitter and the user interests need to be evaluated in order to obtain a ranked list of followee recommendations.

We determine the similarity between the profiles of a candidate user that need to be evaluated for recommendation u_C and the target user u_T, denoted $sim^{T0}\,(u_C, u_T)$, as the cosine similarity between the two vectors. The cosine of the angle conformed by two vectors in the space is calculated as the normalized dot product [Salton and McGill, 1983].

For strategy T1 and T2, in order to evaluate whether to recommend a candidate user u_C to the target user u_T, the information published by the candidate, $Profile^{T0}\,(u_C)$, needs to be compared with the profile of the target user, $Profile^{T1}\,(u_T)$, which is the information u_T is subscribed to receive in Twitter. The matching is then calculated as shown in equation 6.

Finally, the similarity $sim^{T2}\,(u_C, u_T)$ is evaluated in the same way, as specified in Equation 6.

$$sim^{T1}(u_C, u_T) = \max_{\forall i: f_i \in followees(u_T)} sim^{T0}\left(Profile^{T0}(f_i), Profile^{T0}(u_C)\right) \quad (6)$$

	Average	Maximum	Minimum
#followees	94.77±15.54	119	41
#followers	2.0±1.74	10	1
#tweets	102.44±57.56	199	11

Table 1: Summary of statistics of the users selected for testing the approach

For all strategies, all candidate users are ranked according to their similarity to the profile of the target user and the user is presented with a reduced number of followee recommendations.

4 Experimental Evaluation

4.1 Dataset Description

The Twitter dataset[4] used in this paper is a social graph of 835.541 follower/followee relations between 456.107 users and their corresponding tweets belonging to a time span of 2006 to 2009, reaching a total of 10.467.110 tweets. This dataset was created using a focused crawler based on a snowballing technique over a set of quality users, who post about a diverse range of topics and reasonably frequently. In the assemblage of this dataset, reported in [Choudhury *et al.*, 2010], the crawler was seeded with 500 users comprising politicians, musicians, environmentalists and so on; and next the social graph was expanded from the seeds based on the friend links between users.

From the entire dataset a test set $|U_{test}| = 100$ was created to empirically evaluate the content-based followee recommendation approach. Since the recommendation approach is intended to help information seekers in Twitter rather than users serving as information sources, the 100 users were selected on the basis of having their followees outnumbering their followers. Likewise, users who posted less than 10 tweets were excluded from the social graph so that valuable content-based profiles could be extracted for all users involved in the evaluation. The profiles of these target users were built analyzing the text of their tweets according to the strategies proposed in Section 3.3. Table 1 summarizes the characteristics of the U_{test} in terms of number of followees, followers and published tweets.

4.2 Methodology and Metrics

Experiments were carried out using a holdout strategy in which some the target user followees are hidden from the recommendation algorithm and then it is verified if they were discovered and suggested as future followees. In all experiments, the set of followees of each user were partitioned into a 70% for training, starting from which candidates are located and evaluated, and a 30% for testing, whose existence is verified in the list of top-*N* suggested followees for each user in U_{test}. If followees in the 30% group are suggested to the target user in spite of being concealed, it means that the algorithm was able to locate these users through the 70% non-concealed followees and their relationships. In order to make the results less sensitive to the particular training/testing partitioning of the followees, in all experiments the average and standard deviation of 5 runs for each individual user are reported, each time using a different random partitioning into training and test sets.

The quality of lists of top-*N* followee recommendations generated for the group of users used for testing was evaluated considering the standard precision:

$$precision(RE) = \frac{1}{|U_{test}|} \sum_{u \in U_{test}} \frac{|followees_{test}(u) \cap RE_u|}{|RE_u|} \quad (7)$$

where RE_u is the set of recommendations for a user $u \in U_{test}$, U_{test} is the set of users considered for testing (in this work $U_{test} = 100$ as described in the previous section), $followees_{test}(u)$ is the set of followees that were reserved for testing the top-*N* list of a single user u (not used as seeds for starting candidate search).

In other words, precision measures the average percentage of overlap between a given recommendation list and the user actual list of followees and it can be evaluated at different points in a ranked list of suggested followees. Thus, precision at rank *k* (P@*k*) is defined as the proportion of recommended followees that were relevant, i.e. were in the target user test set. In the reported experiments we evaluate precision for values of *k* equal to 1, 5, 10, 15 and 20, although *k* values of 1 and 5 are the most common sizes for recommendation lists reported in the literature as people tend to pay more attention to the first few results that are presented.

Another measure similar to precision is the number of hits in a recommendation list, this is the number of followees in the test set that were also present in the top-*N* recommended followees for a given test user. If $|U_{test}|$ is the total number of testing users, the hit-rate (HR) of the recommendation algorithm is computed as [Deshpande and Karypis, 2004]:

$$HR = \frac{number\ of\ hits}{|U_{test}|} \quad (8)$$

HR grants high values to an algorithm if it is able to predict the followees in the test sets of the corresponding users, while assign low values of the algorithm was not able to recommend the hidden followees.

One limitation of this measure is that it treats all hits equally regardless of where they appear in the list of the top-*N* recommended items. Average reciprocal hit-rank (ARHR) rewards each hit based on where it occurred in the top-*N* followees that were recommended by a particular strategy. If h is the number of hits that occurred at positions $p_1, p_2 \ldots, p_h$ within the top-*N* lists (i.e., $1 \leq p_i \leq N$), then the average reciprocal hit-rank is equal to:

[4]Originally posted at `http://www.public.asu.edu/~mdechoud/datasets.html`

$$ARHR(RE) = \frac{1}{|U_{test}|} \sum_{i=1}^{h} \frac{1}{p_i} \quad (9)$$

That is, hits that occur earlier in the top-*N* lists are weighted higher than hits that occur later in the list. The highest value of ARHR is equal to the hit-rate and occurs when all the hits occur in the first position, whereas the lowest value of the ARHR is equal to hit-rate/*N* when all the hits occur in the last position in the list of the top-*N* recommendations.

4.3 Experimental Results

Table 2 show the precision and hit-rate results for followee recommendations using the different profiling strategies and the mentioned pre-processing techniques for analyzing tweets. The number of candidates explored was on average 6,692.74±511.25 users reached through the user own followees.

It can be observed in the results presented that the users own tweets are not effective for identifying potentially interesting followees. This is probably due to the fact that information seeking users tend to be more passive in posting messages while behave more actively following other people to keep up with interesting information or news.

In contrast, the strategies profiling users based on the information published by their followees either separately or grouped into categories, were more effective in recognizing people to start follow among the candidates found. In fact, the strategy using a vector for each followee outperforms all others for the various sizes of the recommendation lists. When followee vector representations were aggregated into clusters, precision diminished significantly but also the number of similarity calculations is reduced since profiles are of smaller size.

Interestingly, the ARHR values shown in Figure 2 for the four profiling strategies allow to infer that hits are better positioned in the list generated using clustering of followees than in those produced with separate followee vectors. Therefore, the issue of improving the ranking of relevant recommendations will be then matter of future research, particularly exploiting the number of occurrences of the candidates in the set R as a voting mechanism.

It is worth noticing that in the previous results the effectiveness of the algorithm to identify followees is being underestimated given the testing methodology employed. Users suggested to the target user that are not in the test set are not necessarily irrelevant, although they are considered incorrect recommendations in the calculation of the precision and hit-rate metrics. In fact, the target users might not be in their list of followees either because they are not interested on receiving their tweets or because they have not yet discovered the recommended users in the Twitter network. In the last case, these recommendations are also appropriate and will be valuable for the users.

Figure 3 depicts the mean average similarities between the vectors of the users in the top-*N* lists with the corresponding target user profile. The low similarity of information publish by the recommended users and the target user tweets account for the poor results of the first profiling strategy. On the other hand, the high average similarities of users in the top-*N* lists generated by the two last strategies suggests that even the recommended users deemed as irrelevant publish information highly similar to the user profile and to the remaining recommended users in each list, most of which the user is already following. Hence, they are likely good recommendations in spite of being considered otherwise.

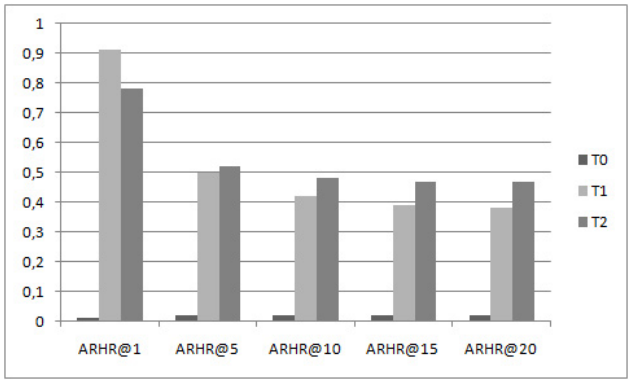


Figure 2: ARHR values of followee recommendations for different profiling strategies

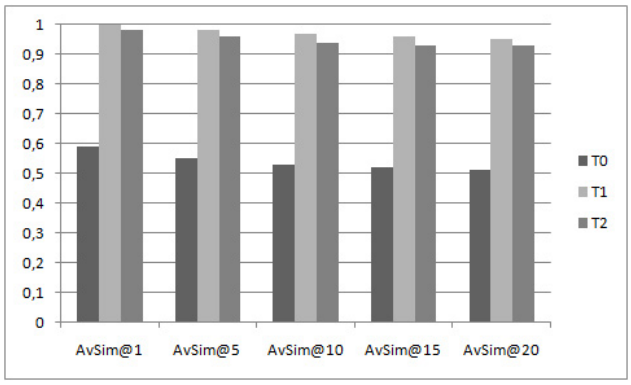


Figure 3: Average similarity of the recommended followees with the target users

5 Conclusions

In this paper an effective algorithm for recommending followees in the Twitter social network dedicated to information-seeking users was presented. This algorithm first explores the social graph in search of candidate recommendations and then ranks these candidates according to the inferred interest of the user that will receive the recommendations on the information the candidates tweet about. The search of suitable candidates was guided by the assumption that the users followed by the followers of a target user followees are potentially interesting and should be further evaluated from a content point of view.

Three different strategies were defined to create content-based profiles of users describing the information they like to received from the people they follow. Using the user's

	T0		T1		T2	
	Average	Std.dev.	Average	Std.dev.	Average	Std.dev.
P@1	0.01	0.01	0.91	0.07	0.78	0.13
P@5	0.01	0.01	0.75	0.08	0.49	0.12
P@10	0.01	0.01	0.61	0.07	0.31	0.09
P@15	0.01	0.01	0.51	0.06	0.22	0.07
P@20	0.00	0.00	0.42	0.06	0.17	0.05
Hits@1	0.01	0.01	0.91	0.07	0.00	0.13
Hits@5	0.03	0.03	3.75	0.38	2.45	0.62
Hits@10	0.06	0.06	6.11	0.70	3.12	0.94
Hits@15	0.08	0.08	7.6	0.96	3.32	1.04
Hits@20	0.09	0.08	8.39	1.2	3.38	1.06

Table 2: Precision and Hit-rate of followee recommendations for different profiling strategies

own tweets, maintaining a term vector for each followee, and grouping followees into categories by means of a clustering algorithm. Thus, candidates are ranked according to the similarity of their tweets with these models of the target user interests in order to recommend a list of top-*N* followees.

Experimental evaluation using a dataset containing a sample of Twitter social graph and the tweets of each user in this graph was carried out in order to validate the approach and compare the performance of the proposed profiling strategies. The achieved results show that the user own tweets are not a good source of profiling knowledge. In contrast, strategies using the posts of the followees of users, either individually or grouped into categories, for modeling their interests reached high levels of precision in recommendation.

Future work will be oriented to obtain further improvements in the performance of the approach by varying the text analysis techniques applied to tweets and the ranking scheme. In the first point, we are currently working on exploiting terms appearing in the URLs linked in tweets as well as words related to hashtags to expand the tweet textual representation. In the second point, the work envisioned consists in measuring the impact that factors such as the number of occurrences in the candidates set or the relation followers/followees that characterize good information sources have on ranking effectiveness.

References

[Cha *et al.*, 2010] M. Cha, H. Haddadi, F. Benevenuto, and K. Gummadi. Measuring user influence in Twitter: The million follower fallacy. In *Proceedings of the 4th International Conference on Weblogs and Social Media (ICWSM'10)*, Washington DC, USA, 2010.

[Chen *et al.*, 2009] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends, but keep the old: recommending people on social networking sites. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, pages 201–210, Boston, MA, USA, 2009.

[Chen *et al.*, 2010] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems (CHI'10)*, pages 1185–1194, Atlanta, Georgia, USA, 2010.

[Choudhury *et al.*, 2010] M. De Choudhury, Y-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A Kelliher. How does the data sampling strategy impact the discovery of information diffusion in social media? In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM'10)*, 2010.

[Davidov *et al.*, 2010] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using Twitter hashtags and smileys. In *Proceeding of the 23rd International Conference on Computational Linguistics (COLING'2010)*, pages 241–249, Beijing, China, 2010.

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

[Esparza *et al.*, 2010] S. Garcia Esparza, M. P. O'Mahony, and B. Smyth. On the real-time web as a source of recommendation knowledge. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*, pages 305–308, Barcelona, Spain, 2010.

[Garcia and Amatriain, 2010] R. Garcia and X. Amatriain. Weighted content based methods for recommending connections in online social networks. In *Workshop on Recommender Systems and the Social Web*, pages 68–71, Barcelona, Spain, 2010.

[Guy *et al.*, 2009] I. Guy, I. Ronen, and E. Wilcox. Do you know?: recommending people to invite into your social network. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI'09)*, pages 77–86, 2009.

[Hannon *et al.*, 2010] J. Hannon, M. Bennett, and B. Smyth. Recommending Twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys'10)*, pages 199–206, 2010.

[Java *et al.*, 2007] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st*

SNA-KDD 2007 Workshop on Web Mining and Social Network Analysis, pages 56–65, 2007.

[Krishnamurthy *et al.*, 2008] B. Krishnamurthy, P. Gill, and M. Arlitt. A few chirps about Twitter. In *Proceedings of the 1st Workshop on Online Social Networks (WOSP'08)*, pages 19–24, Seattle, WA, USA, 2008.

[Kwak *et al.*, 2010] 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 (WWW'10)*, pages 591–600, Raleigh, North Carolina, USA, 2010.

[Liben-Nowell and Kleinberg, 2003] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the12th International Conference on Information and Knowledge Management (CIKM'03)*, pages 556–559, New Orleans, LA, USA, 2003.

[Lo and Lin, 2006] S. Lo and C. Lin. WMR–A graph-based algorithm for friend recommendation. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, pages 121–128, Washington, DC, USA, 2006.

[Naaman *et al.*, 2010] M. Naaman, J. Boase, and C-H. Lai. Is it really about me?: message content in social awareness streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work (CSCW'10)*, pages 189–192, Savannah, Georgia, USA, 2010.

[Pak and Paroubek, 2010] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the 7th Conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010.

[Perez-Tellez *et al.*, 2010] F. Perez-Tellez, D. Pinto, J. Cardiff, and P. Rosso. On the difficulty of clustering company tweets. In *Proceedings of the 2nd International Workshop on Search and Mining User-Generated Contents (SMUC'10)*, pages 95–102, Toronto, ON, Canada, 2010.

[Phelan *et al.*, 2009] O. Phelan, K. McCarthy, and B. Smyth. Using Twitter to recommend real-time topical news. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys'09)*, pages 385–388, New York, NY, USA, 2009.

[Porter, 1980] M. Porter. An algorithm for suffix stripping program. *Program*, 14(3):130–137, 1980.

[Ramage *et al.*, 2010] D. Ramage, S. Dumais, and D. Liebling. Characterizing microblogs with topic models. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM'10)*, Washington, DC, USA, 2010.

[Salton and McGill, 1983] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[Sun *et al.*, 2009] A. R. Sun, J. Cheng, and D. D. Zeng. A novel recommendation framework for micro-blogging based on information diffusion. In *Proceedings of the 19th Workshop on Information Technologies and Systems*, 2009.

[Weng *et al.*, 2010] J. Weng, E-P. Lim, J. Jiang, and Q. He. TwitterRank: finding topic-sensitive influential twitterers. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10)*, pages 261–270, New York, NY, USA, 2010.

[Yamaguchi *et al.*, 2010] Y. Yamaguchi, T. Takahashi, T. Amagasa, and H. Kitagawa. TURank: Twitter user ranking based on user-tweet graph analysis. In *Web Information Systems Engineering*, volume 6488 of *LNCS*, pages 240–253, Hong Kong, China, 2010.

Analyzing the Potential of Microblogs for Spatio-Temporal Popularity Estimation of Music Artists

Markus Schedl
Department of Computational Perception
Johannes Kepler University
Linz, Austria
http://www.cp.jku.at

Abstract

This paper looks into the suitability of microblogs for an important task in music information research, namely popularity estimation of music artists. The research questions addressed are the following: To which extent are microblogs used to communicate music listening behavior? Are there differences between different countries of the world? Is it possible to derive a popularity measure from user's microblogging activities?
We found that microblogging does indeed represent an important communication channel for revealing music listening activities, although the intensity of its use vary considerably from country to country. Motivated by this finding, we took first steps towards a *geo-aware, social popularity measure* for music artists. To this end, we analyzed user posts mined from the microblogging service Twitter over a period of five months. Addressing the problem of determining the popularity of music artists, we employed a gazetteer on extracted posts relevant for particular music artists. The presented approach aims at extracting time- and location-specific artist popularity information. We evaluated the performance of the approach by comparing the popularity rankings derived from Twitter posts against the popularity rankings provided by last.fm, a popular music information system and recommender engine.

1 Motivation and Context

The emergence of microblogging services date back to 2005. However, they gained greater popularity not before the years 2007 and 2008[1]. Today's most popular microblogging service is Twitter[2], where millions of users post what they are currently doing or what is currently important to them [Kazeniac, 2009].

The work at hand tackles a problem from music information research (MIR), a field that is concerned with the extraction, analysis, and usage of information about any kind of music entity (for example, a song or an album) on any representation level (for example, an audio signal, a symbolic MIDI representation, or an artist's name) [Schedl, 2008]. Figuring out which artists/performers of music are popular is an interesting research question, not at last for the music industry, but also for the artists themselves and for the interested music aficionado.

In MIR we can distinguish three broad categories of modeling music items with respect to the underlying data source, namely *music content*-based [Casey *et al.*, 2008], *music context*-based [Schedl, 2011], and *user context*-based [Göker and Myrhaug, 2002] approaches. Feature vectors describing aspects from one or more of these three categories can be constructed, and similarity measures can be applied to the resulting vectors of two pieces of music or two music artists/performers. Elaborating such musical similarity measures that are capable of capturing aspects that relate to perceived similarity is one of the main challenges in MIR. Such measures are a key ingredient of various music applications, for example, automatic playlist generators [Aucouturier and Pachet, 2002; Pohle *et al.*, 2007], music recommender systems [Celma, 2008], music information systems [Schedl, 2008], semantic music search engines [Knees *et al.*, 2007], and intelligent user interfaces [Pampalk and Goto, 2007] to music collections.

For all applications mentioned, popularity information can be of particular value. For example, a music recommender system or a playlist generator will benefit from popularity information in that it will allow adapting its recommendations or playlists to different types of users. It is a well-known fact that different kinds of users (in terms of their music understanding and background) require different music recommendations, cf. [Celma, 2008]. In particular, music experts in certain styles or genres get quickly bored if such a system keeps on recommending popular artists that are already known to this sort of user. Incorporating popularity information – in this case, including "long tail" artists in the recommendations – is likely to yield serendipitous results for such users.

Addressing popularity as an important category of music-related information, the work at hand was driven by two research questions: To which extent are microblogs used to express music preferences and listening activities in different

[1] http://www.sysomos.com/insidetwitter (access: March 2010)

[2] http://www.twitter.com (access: January 2011)

places around the world? Is it possible to derive a popularity measure from user's microblogging activities? The first question will be answered by an analysis of microblogs about music listening in Section 3.1. Prior to that, Section 2 describes the data acquisition and popularity estimation steps. Here, we present first steps towards deriving *location- and time-specific music popularity information* from posts of Twitter users. The second question is addressed in Section 3.2, where we report on the results of quantitative experiments comparing the microblog-based popularity estimates with a reference data set extracted from last.fm[3]. Eventually, Section 4 summarizes the work and points out some directions for future research.

1.1 Microblog Mining

With the advent of microblogging, a huge, albeit noisy data source became available. Since millions of Twitter users tweet around the world, telling everyone who is interested what is important to them, microblogs are an obvious source to derive popularity information. However, it seems that literature dealing with microblogs mostly studies human factors (e.g., [Teevan *et al.*, 2011]) or describes properties of the Twittersphere (e.g., [Java *et al.*, 2007; Kwak *et al.*, 2010]). A general study on the use of Twitter can be found in [Java *et al.*, 2007]. Java et al. report that Twitter is most popular in North America, Europe, and Asia (Japan), and that same language is an important factor for cross-connections ("followers" and "friends") over continents. The authors also distilled certain categories of user intentions to microblog. Employing the *HITS* algorithm [Jon M. Kleinberg, 1999] on the network constructed by "friend"-relations, Java et al. derive user intentions from structural properties. They identified the following categories: information sharing, information seeking, and friendship-wise relationships. Analyzing the content of Twitter posts, the authors distill the following intentions: daily chatter, conversations, sharing information/URLs, and reporting news.

Scientific work related to content mining of microblogs includes the following: Cheng et al. propose a method to localize Twitter users based on cues ("local" words) extracted from their tweets' content [Cheng *et al.*, 2010]. Sakaki et al. propose semantic analysis of tweets to detect earthquakes in Japan in real-time [Sakaki *et al.*, 2010]. A more general approach to automatically detect events and summarize trends by analyzing tweets is presented by Sharifi et al. [Sharifi *et al.*, 2010].

1.2 Popularity Estimation for Music

Determining the popularity of a music artist or song is a relatively new research area. The earliest work in this direction, to the best of our knowledge, is [Grace *et al.*, 2008], where Grace et al. estimate popularity rankings based on user posts mined from myspace[4]. The authors apply different annotators to artist pages in order to detect artist, album, and track names, as well as descriptions of sentiments and spam. A data hypercube (OLAP cube) is then used to project the data to a one-dimensional popularity space. Based on a conducted user study, the authors conclude that the list generated by this method is on average preferred to the Billboard charts[5].

Using search queries raised in the Peer-to-Peer network Gnutella [Ripeanu, 2001],[Koenigstein and Shavitt, 2009] present an approach to predict music charts. The authors demonstrate that a song's popularity in the Gnutella network correlates with its ranking in the Billboard charts. For their analysis Koenigstein and Shavitt only consider the United States of America, because of its predominance in available data.

The work at hand is probably most related to [Schedl *et al.*, 2010], where popularity estimates of music artists are calculated on the country level using different data sources. Schedl et al. use page count estimates returned by Web search engines as result of music artist-related requests, user posts returned by Twitter as result of artist-related queries, information on music files shared by users of the Gnutella file sharing network, and playcount data extracted from last.fm. The approach proposed here is different from [Schedl *et al.*, 2010] in that Schedl et al.'s work only allow for an overall popularity prediction on the country level. It does neither take into account the *popularity on the level of individual cities*, nor the *time-dependence of the popularity estimate*. Both factors are, in our belief, indispensable for a fine-grained analysis of popularity.

Using content-based audio features and manually assigned labels to predict the popularity of a song is addressed in [Pachet and Roy, 2008]. Pachet and Roy's conclusion is, however, that even state-of-the-art machine learning techniques fail to learn factors that determine a song's popularity, irrespective of whether they are trained on signal-based features or on high-level human annotations.

2 Microblog Mining for Popularity Estimation

Since we are interested in the *spatio-temporal popularity distribution* of music artists, we first extracted in May 2010 from World Gazetteer[6] a list of world's largest agglomerations. The data set comprises 790 cities with at least 500,000 inhabitants. We further gathered the corresponding location information (longitude- and latitude-values).

Using these coordinates, we monitored user posts on Twitter that include geographic positioning information for a period of five months, more precisely, from May to September 2010. To this end, the geo-localization methods provided by the Twitter API[7] were used. We searched for the exact longitude- and latitude-coordinates of the agglomerations in our list and added a search radius of 50 kilometers in order to account for surrounding suburbs. Since we focused our analysis on music listening activities, we restricted the extraction of messages to posts including the #nowplaying

[3] http://last.fm (access: January 2011)

[4] http://www.myspace.com (access: November 2010)

[5] http://en.wikipedia.org/wiki/Billboard_Hot_100 (access: May 2009)

[6] http://world-gazetteer.com (access: October 2010)

[7] http://apiwiki.twitter.com/Twitter-API-Documentation (access: January 2011)

hashtag, as this descriptor is commonly used to refer to music currently played by the user. It has to be noted, however, that only a few percentage ($< 5\%$) of tweets come along with geo-local information. Therefore, our results may be biased towards technology-affine users who possess the latest generation of smartphones or other mobile computing equipment.
Having gathered user posts together with spatio-temporal information in this way, we built a word-level index [Zobel and Moffat, 2006] applying casefolding and stopping. We then employed an annotation component, whose knowledge base comprised of $3{,}000$ names of music artists. To this end, we retrieved the overall most popular artists from `last.fm` using their Web API[8]. Since `last.fm`'s data is known to contain a high amount of misspellings or other mistakes due to their collaborative, user-generated knowledge base [Lamere, 2008], we cleaned the data set by first matching each artist name with the database of the expert-based music information system `allmusic.com`[9] and second retaining only those names that were also known by `allmusic.com`. For indexing the tweets we used a modified version of the `lucene`[10] indexer. We adapted the retrieval component for optimized retrieval of ranked artist sets, given a particular day and location. Ranking is simply performed according to the total count of artist occurrences in the respective posts, which corresponds to the *term frequency* $tf_{a,l,t}$ of term a (denoting the artist name) in the posts retrieved for a particular location l at a particular time t.

The list of agglomerations we used can be downloaded from `[omitted due to to double-blind review]`. The set of $3{,}000$ artist names is available at `[omitted due to to double-blind review]`.

3 Statistics and Evaluation

Experimentation was driven by two main questions. On the one hand, we were interested in the *distribution of music-related `Twitter` posts* around the world. Assessing whether the quantity of available information varies considerably for different regions of the world was one subtask. The other was investigating if any differences between the general use of `Twitter` and the posting of music-related information can be detected.
The second set of experiments addressed the question whether music-related posts are capable of revealing information on the *current popularity of music artists*. To this end, we compared the location- and time-specific information derived from the `Twitter` posts with artist charts gathered from the music information system `last.fm`.

3.1 Geographical Analysis of the Data

Analyzing the geographical distribution of `Twitter` users who reveal their current music taste (by including the `#nowplaying` hashtag in their posts) was our first objective. Figures 1 and 2 show the cities whose inhabitants are most active in terms of posting listening-related messages. Figure 1 reveals the top 10% of cities with the highest absolute number of music-related `Twitter` posts, whereas Figure 2 shows the cities with highest relative number of posts, normalized by the respective city's number of inhabitants. Taking a closer look at the most active agglomerations in absolute terms, the dominance of Asian metropolises stands out; six out of the top 10 cities are located in Asia, two in Europe, one in North America, and one in South America. When analyzing the activeness of inhabitants relative to their city's size, Brazil dominates the ranking (four out of the top 10 cities). Four cities are located in Asia, one in Europe, and five in South America (one in Chile, in addition to the four in Brazil). This might reflect the ascribed affinity of South Americans to music and their openness to talk about their habbits and activities.

To obtain an estimate of music-related `Twitter` use on the country level, we aggregated the data obtained for individual cities to the respective countries. The results are shown in Figures 3 and 4 as absolute amount of posts and as number of posts relative to the number of inhabitants, respectively.
These figures are largely in line with a report that explores the use of `Twitter` around the world [Evans, 2010] and shows the top 20 countries in terms of total number of posts contributed. However, some interesting outliers can be found. Comparing the list of top-ranked countries in terms of total tweet contributed [Evans, 2010] with the 20 top-ranked countries in terms of music listening-related posts (cf. Figure 3) allows to approximate in which countries `Twitter` is disproportionately often or seldom used to report on listening activities: Countries whose inhabitants are most frequent users of `Twitter`, but not among the top 20 in terms of sharing current listening activities are Australia (ranked 5^{th} in terms of total tweet contributed according to [Evans, 2010]), Singapore (12^{th}), France (14^{th}), Ireland (15^{th}), New Zealand (17^{th}), Italy (19^{th}), and Iran (20^{th}). On the other hand, a disproportionately high amount of music-related posts in relation to overall `Twitter` usage can be found in China (ranked 6^{th} in Figure 3), South Korea (7^{th}), Venezuela (8^{th}), South Africa (16^{th}), Colombia (18^{th}), Chile (19^{th}), and the Dominican Republic (20^{th}). These countries occur among the top 20 in our list reporting on music-related posts, but are not included in the list of top 20 countries for total tweet contributed.

3.2 Comparison of `Twitter` and `last.fm` Popularities

In a second set of evaluation experiments, we compared the set of popular artists extracted from `Twitter` posts with a reference set. Since traditional music charts, such as the "Billboard Hot 100" released weekly for the United States of America by the `Billboard Magazine`, are neither available on the level of individual cities, nor for all countries in the world, we gathered popularity data from `last.fm` as follows.

`last.fm` provides weekly artist charts for selected "metros". Taking as input the list of 790 locations gathered from `World Gazetteer`, we were able to extract artist charts

[8] `http://last.fm/api` (access: March 2010)
[9] `http://allmusic.com` (access: January 2011)
[10] `http://lucene.apache.org` (access: January 2011)

for 84 of the corresponding metropolises. In order to better understand the quality of the data on different temporal levels, various experimental settings were evaluated. The results are summarized in Table 1. First, we performed an exact *day-to-day* comparison experiment. To this end, for all pairs of days and locations for which information was available from both sources (`Twitter` and `last.fm`), we compared the extracted artist sets. Since `last.fm` provides popularity information only on the level of weeks, we interpolated this weekly information to individual days. The results of this day-to-day comparison experiment, averaged over all locations and days under consideration, are depicted in the second column of Table 1, labeled `D2D`. Taking the `last.fm` artist set as reference set, we calculated the following performance measures:

Precision at a specific day t and location l is defined as the fraction of artists found in `Twitter` messages posted at day t and location l that are also reported by `last.fm`'s chart function for l and t, among the total number of artists found in `Twitter` messages for day t and location l. Formally,

$$prec_{t,l} = \frac{\left|A_{t,l}^{tw} \cap A_{t,l}^{fm}\right|}{\left|A_{t,l}^{tw}\right|}$$

where $A_{t,l}^{tw}$ is the set of popular artists predicted by our approach for time t and location l, and $A_{t,l}^{fm}$ is the set of popular artists reported by `last.fm`.

Recall is defined as the percentage of `last.fm` artists for t and l that are also part of the artist set extracted from `Twitter` messages at t for l:

$$rec_{t,l} = \frac{\left|A_{t,l}^{tw} \cap A_{t,l}^{fm}\right|}{\left|A_{t,l}^{fm}\right|}$$

F_1*-measure* is the weighted harmonic mean of precision and recall [van Rijsbergen, 1979]:

$$F_1 = \frac{2 \cdot prec \cdot rec}{prec + rec}$$

Overlap is defined as the number of artists occurring in both sources divided by the maximum number of artists retrieved by either source (at t for l):

$$overlap_{t,l} = \frac{\left|A_{t,l}^{tw} \cap A_{t,l}^{fm}\right|}{\max\left(\left|A_{t,l}^{tw}\right|, \left|A_{t,l}^{fm}\right|\right)}$$

Alleviating the very strict matching requirement of the day-to-day experiment, we further performed *city-to-city* matching by aggregating all `Twitter` posts retrieved for each city (regardless of the date) and comparing them to the city's aggregated `last.fm` charts for the same period (the five months for which we gathered data). The precision on the city level is calculated based on artist sets A_l^{tw} and A_l^{fm}, irrespective of the date t:

$$prec_l = \frac{\left|A_l^{tw} \cap A_l^{fm}\right|}{|A_l^{tw}|}$$

The definition of recall, F_1-measure, and overlap updates analogously.

The results of this experiment can be found in the third column of Table 1, labeled `C2C`. It can be seen that the average recall increases substantially compared to the day-to-day setting, while the average precision remains almost the same. This can be explained by the disproportionately low number of artists covered by `last.fm` charts, compared to the number extracted from `Twitter`, for this granularity level. In fact, the average number of unique artists in `Twitter` posts exceeds the average number of unique artists covered by `last.fm` charts by a factor of five – cf. first two rows of Table 1. The considerable improvement of the `C2C` setting over the `D2D` setting might also be explained by a temporal lead or lag of the two data sources `last.fm` and `Twitter`, which is smoothed out when temporal aspects are ignored.

Further broadening the scope of matching yields the final experiment conducted. For this overall matching experiment, all extracted `Twitter` posts as well as all retrieved `last.fm` charts were aggregated, and the performance measures were only calculated on the resulting two artist sets. This setting can be thought of as a global popularity prediction. The precision for the overall matching experiment is calculated on artist sets A^{tw} and A^{fm}, irrespective of both date t and location l:

$$prec = \frac{\left|A^{tw} \cap A^{fm}\right|}{|A^{tw}|}$$

The definition of recall, F_1-measure, and overlap updates correspondingly.

The results of this overall matching experiment are given in the fourth column of Table 1. Please note that in the table average performance values are given for day-to-day matching (averaged over all locations and dates) and city-to-city matching (averaged over all locations), whereas total scores are given for the overall comparison experiment. Hence, for day-to-day matching, average precision is calculated as

$$prec = |T|^{-1} \cdot |L|^{-1} \cdot \sum_{l \in L} \sum_{t \in T} \frac{\left|A_{t,l}^{tw} \cap A_{t,l}^{fm}\right|}{\left|A_{t,l}^{tw}\right|},$$

whereas for city-to-city matching average precision is calculated as

$$prec = |L|^{-1} \cdot \sum_{l \in L} \frac{\left|A_l^{tw} \cap A_l^{fm}\right|}{|A_l^{tw}|}.$$

L denotes the set of locations, whereas T denotes the points in time (days) for which information is available. The other performance measures are calculated analogously.

Addressing the question if the quality of the popularity estimates is consistent over different cities, Figure 5 depicts the individual precision and recall values for all agglomerations for which `last.fm` provided corresponding data. The cities are sorted according to the F_1-measure. As it can be seen, the results vary strongly over different cities. The standard deviation of the precision values is $\sigma_{prec} = 0.0678$, that of the recall values equals $\sigma_{rec} = 0.2403$.

4 Conclusions and Outlook

We presented an analysis of music-related microblogging activity around the world and a simple popularity measure based on music artists' term frequencies in `Twitter` posts. Investigating the spatial distribution of music-related tweets revealed a considerable dominance of Asian countries (in terms of absolute number of posts) and of South American countries (in terms of number of posts relative to the number of inhabitants). The *location- and time-specific popularity measure* was evaluated in various experiments on different scales of granularity. On the level of individual days, the approach yielded modest precision and recall values, whereas remarkable recall could be achieved when aggregating the location-specific posts for all days under consideration.

Future work will be centered around exploiting the fine-grained day-level rankings. They could be used, for example, to illustrate changes in popularity around the world. For the application scenario of music chart prediction, the rankings could be used to complement traditional music charts, as they are generally biased towards actual music sales and also neither available on the city level, nor for all countries in the world. We will also experiment with data from domains other than music. For example, we are currently investigating different term weighting approaches to predict popularity of movies. Furthermore, it would also be interesting to analyze if certain popularity patterns can be clustered according to properties such as country, continent, or language group. Another direction for future work will be visualizing the derived popularity information. By applying time-series visualization techniques [Few, 2007], changes in popularity could be appealingly illustrated, for example via popularity "Flow Maps" [Phan *et al.*, 2005]. Reconsidering our main research focus on music information retrieval, artist popularity estimates on different geographical scopes and temporal points can help build personalized models of musical similarity and user preferences, which may ultimately yield to better personalized music services and applications, such as automatic playlist generators and music recommender systems.

Acknowledgments

This research is supported by the Austrian Science Funds (FWF): P22856-N23 and L511-N15.

References

[Aucouturier and Pachet, 2002] Jean-Julien Aucouturier and François Pachet. Scaling Up Music Playlist Generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2002)*, pages 105–108, Lausanne, Switzerland, August 2002.

[Casey *et al.*, 2008] Michael A. Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96:668–696, April 2008.

[Celma, 2008] Òscar Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2008.

[Cheng *et al.*, 2010] Zhiyuan Cheng, James Caverlee, and Kyumin Lee. You Are Where You Tweet: A Content-Based Approach to Geo-Locating Twitter Users. In *Proc. of the 19th ACM Int'l Conference on Information and Knowledge Management (CIKM)*, Oct 2010.

[Evans, 2010] Mark Evans. Exploring the Use of Twitter Around the World. http://blog.sysomos.com/2010/01/14/exploring-the-use-of-twitter-around-the-world (access: October 2010), 2010.

[Few, 2007] Stephen Few. Visualizing Change: An Innovation in Time-Series Analysis. *Visual Business Intelligence Newsletter*, September 2007.

[Göker and Myrhaug, 2002] Ayse Göker and Hans I Myrhaug. User Context and Personalisation. In *Proceedings of the 6th European Conference on Case Based Reasoning (ECCBR 2002): Workshop on Case Based Reasoning and Personalization*, Aberdeen, Scotland, September 2002.

[Grace *et al.*, 2008] Julia Grace, Daniel Gruhl, Kevin Haas, Meenakshi Nagarajan, Christine Robson, and Nachiketa Sahoo. Artist Ranking Through Analysis of On-line Community Comments. In *Proceedings of the 17th ACM International World Wide Web Conference (WWW 2008)*, Bejing, China, April 21–25 2008.

[Java *et al.*, 2007] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why We Twitter: Understanding Microblogging Usage and Communities. In *Proceedings of WebKDD/SNA-KDD*, 2007.

[Jon M. Kleinberg, 1999] Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5), 1999.

[Kazeniac, 2009] Andy Kazeniac. Social Networks: Facebook Takes Over Top Spot, Twitter Climbs. http://blog.compete.com/2009/02/09/facebook-myspace-twitter-social-network (access: March 2010), 2009.

[Knees *et al.*, 2007] Peter Knees, Tim Pohle, Markus Schedl, and Gerhard Widmer. A Music Search Engine Built upon Audio-based and Web-based Similarity Measures. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007)*, Amsterdam, the Netherlands, July 23–27 2007.

[Koenigstein and Shavitt, 2009] Noam Koenigstein and Yuval Shavitt. Song Ranking Based on Piracy in Peer-to-Peer Networks. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, Kobe, Japan, October 2009.

Table 1: Summary of the experiments comparing `Twitter` and `last.fm` popularity rankings.

Property	**D2D**	**C2C**	**Overall**
Avg. number of artists in `Twitter` posts	21.97	410.49	2,490
Avg. number of artists in `last.fm` charts	37.49	79.94	1,534
Avg. precision on `last.fm` charts (%)	11.16	12.70	51.68
Avg. recall on `last.fm` charts (%)	6.36	51.80	83.90
Avg. F_1-measure on `last.fm` charts (%)	8.10	20.39	63.96
Avg. overlap between `Twitter` posts and `last.fm` charts (%)	4.43	11.05	51.68

[Kwak *et al.*, 2010] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, A Social Network or a News Media? In *Proc. of the 19th Int'l Conference on World Wide Web (WWW)*, Apr 2010.

[Lamere, 2008] Paul Lamere. Social Tagging and Music Information Retrieval. *Journal of New Music Research: From Genres to Tags – Music Information Retrieval in the Age of Social Tagging*, 37(2):101–114, 2008.

[Pachet and Roy, 2008] François Pachet and Pierre Roy. Hit Song Science is Not Yet a Science. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR 2008)*, Philadelphia, PA, USA, September 2008.

[Pampalk and Goto, 2007] Elias Pampalk and Masataka Goto. MusicSun: A New Approach to Artist Recommendation. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, September 23–27 2007.

[Phan *et al.*, 2005] Doantam Phan, Ling Xiao, Ron Yeh, Pat Hanrahan, and Terry Winograd. Flow Map Layout. In *Proceedings of IEEE Information Visualization 2005 (InfoVis 2005)*, Minneapolis, Minnesota, USA, October 2005.

[Pohle *et al.*, 2007] Tim Pohle, Peter Knees, Markus Schedl, Elias Pampalk, and Gerhard Widmer. "Reinventing the Wheel": A Novel Approach to Music Player Interfaces. *IEEE Transactions on Multimedia*, 9:567–575, 2007.

[Ripeanu, 2001] Matei Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Proceedings of IEEE Peer-to-Peer Computing*, 2001.

[Sakaki *et al.*, 2010] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake Shakes Twitter Users: Real-Time Event Detection by Social Sensors. In *Proc. of the 19th Int'l Conference on World Wide Web (WWW)*, Apr 2010.

[Schedl *et al.*, 2010] Markus Schedl, Tim Pohle, Noam Koenigstein, and Peter Knees. What's Hot? Estimating Country-Specific Artist Popularity. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, the Netherlands, August 2010.

[Schedl, 2008] Markus Schedl. *Automatically Extracting, Analyzing, and Visualizing Information on Music Artists from the World Wide Web*. PhD thesis, Johannes Kepler University Linz, Linz, Austria, 2008.

[Schedl, 2011] Markus Schedl. *Music Data Mining*, chapter Web- and Community-based Music Information Extraction. Taylor-Frances/CRC, 2011.

[Sharifi *et al.*, 2010] Beaux Sharifi, Mark-Anthony Hutton, and Jugal Kalita. Summarizing Microblogs Automatically. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, Jun 2010.

[Teevan *et al.*, 2011] Jaime Teevan, Daniel Ramage, and Meredith Ringel Morris. #TwitterSearch: A Comparison of Microblog Search and Web Search. In *Proc. of the 4th ACM Int'l Conference on Web Search and Data Mining (WSDM)*, Hong Kong, China, Feb 2011.

[van Rijsbergen, 1979] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, London, UK, 2nd edition, 1979.

[Zobel and Moffat, 2006] Justin Zobel and Alistair Moffat. Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38:1–56, 2006.

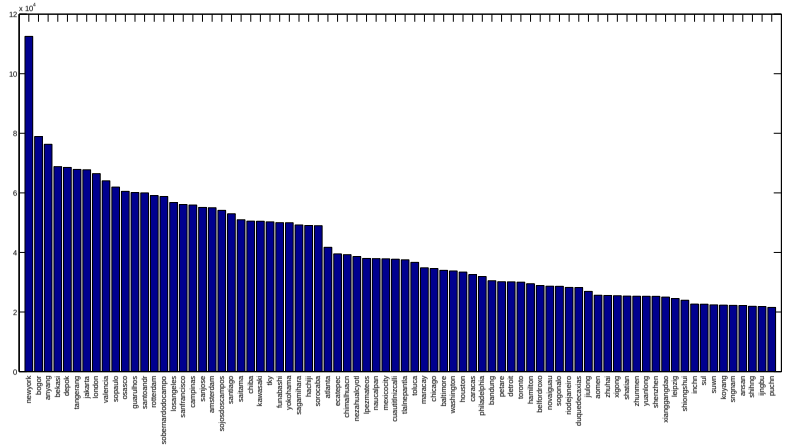


Figure 1: Cities with largest total number of `#nowplaying` posts (> 90th percentile).

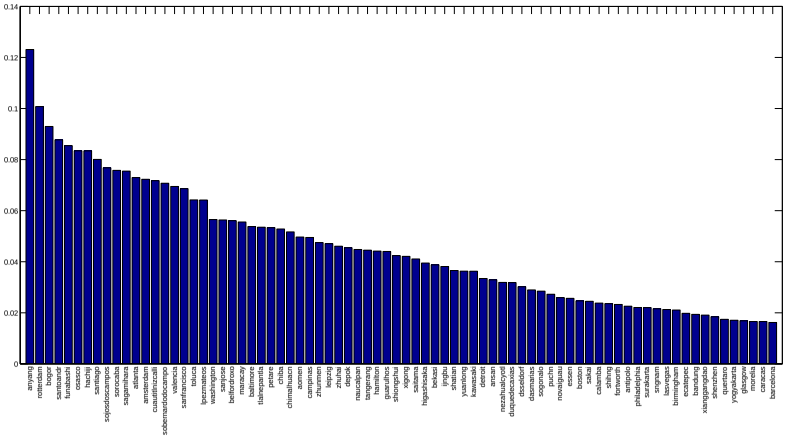


Figure 2: Cities with largest relative number of `#nowplaying` posts (> 90th percentile).

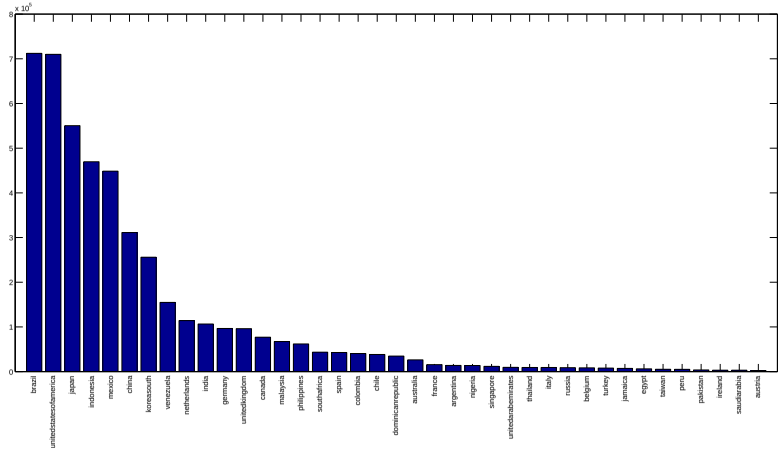


Figure 3: Countries with largest total number of `#nowplaying` posts (> 70th percentile).

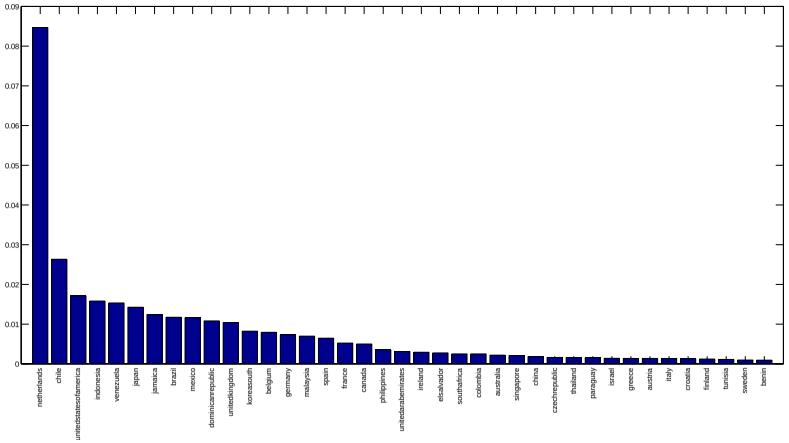


Figure 4: Countries with largest relative number of `#nowplaying` posts (> 70th percentile).

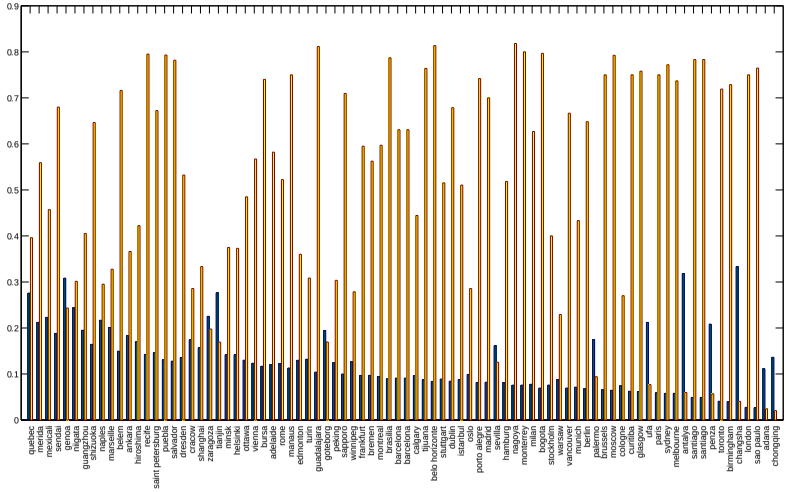


Figure 5: Precision (dark bars) and recall (bright bars) of the `Twitter`-based popularity estimation evaluated on `last.fm`, sorted by F_1-measure.