LocalRank - Neighborhood-based, fast computation of tag recommendations

Marius Kubatz, Fatih Gedikli**, and Dietmar Jannach

Technische Universität Dortmund,
44221 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. On many modern Web platforms users can annotate the available online resources with freely-chosen tags. This Social Tagging data can then be used for information organization or retrieval purposes. Tag recommenders in that context are designed to help the online user in the tagging process and suggest appropriate tags for resources with the purpose to increase the tagging quality. In recent years, different algorithms have been proposed to generate tag recommendations given the ternary relationships between users, resources, and tags. Many of these algorithms however suffer from scalability and performance problems, including the popular *FolkRank* algorithm. In this work, we propose a neighborhood-based tag recommendation algorithm called *LocalRank*, which in contrast to previous graph-based algorithms only considers a small part of the user-resource-tag graph. An analysis of the algorithm on a popular social bookmarking data set reveals that the recommendation accuracy is on a par with or slightly better than FolkRank while at the same time recommendations can be generated instantaneously using a compact in-memory representation.

Key words: recommender systems, collaborative filtering, social tagging

1 Introduction

More and more Social Web platforms such as Delicious or Flickr but also e-commerce sites such as Amazon allow their users to annotate the online resources with freely-chosen tags[1]. In recent years, these community-created *folksonomies* have emerged as a valuable tool for content organization or retrieval in the participatory web. In contrast, for example, to formal Semantic Web ontologies, Social Tagging represents a more light-weight approach, which does not rely on a pre-defined set of concepts and terms that can be used for annotation. The advantage of Social Tagging lies in the fact that no special knowledge is required by the users. On the other hand, the value of the community-provided tags can be limited because no consistent vocabulary may exist as users have their own

** Contact author.

[1] http://delicious.org, http://flickr.com, http://www.amazon.com

style and preferences which tags they use and which aspects of the resource they annotate. A picture of a car could for instance be annotated with tags such diverse as "red", "cool", or "mine"[1]. In [2], for example, Sen et al. reported that only 21% of the tags in the MovieLens system[2] had adequate quality to be displayed to the user.

One way to counteract this effect is to provide the user with a list of tag recommendations to choose from. When the users are provided with a set of tag suggestions, the goal is that the annotation vocabulary as a whole becomes more homogenous across users and that in addition the tagging volume increases, see [3]. In recent years, several approaches to building such *tag recommenders* have been proposed. The state-of-the-art *FolkRank* algorithm [4], for example, represents one early graph-based recommendation approach which was inspired by Google's PageRank [5] and which is still used as a baseline for comparison in the development of new tag recommender approaches today. Later on, different other tag recommendation algorithms have been proposed that rely on techniques such as tensor factorization and latent semantic analysis [6, 7], follow a probabilistic approach [8, 9, 10] or use hybridization strategies [11]. Some approaches also even go beyond recommendation, and try to automatically generate and attach personalized tags for Web pages [12].

Beside improving the predictive accuracy, the question of scalability and the time needed for computing the recommendations is a major issue for the different approaches. Rendle et al. [6, 7] for example conclude that FolkRank does not scale to larger problem sizes and report much shorter running time figures for their own tensor factorization approach. A clustering approach is developed in [13] to allow for "real-time" recommendation.

In this work, we also focus on the issue of scalability of tag recommendation to larger data sets. We therefore propose a graph- and neighborhood-based tag recommendation approach, which is not only capable of generating tag recommendations very quickly also for larger data sets, but which can also be efficiently updated when new data arrives. At the same time, we show that despite its simplicity, the accuracy of our method is comparable to that of FolkRank on the commonly-used Delicious data set.

2 FolkRank and LocalRank

2.1 Folksonomies and FolkRank

The LocalRank algorithm proposed in this paper is based on the ideas of FolkRank, which we will shortly discuss in the following section. Hotho et al. [4] define a folksonomy as a tuple $\mathbb{F} := (U, T, R, Y, \prec)$ where

– U, T , and R are finite sets, whose elements are called users, tags, and resources,

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

– Y is a ternary relation between them, i. e., $Y \subseteq U \times T \times R$, called tag assignments, and
– $\prec$ is a user-specific subtag/supertag-relation, i.e., $\prec \subseteq U \times T \times T$, called subtag/supertag relation. Note that in our work $\prec$ is an empty set[3].

The main idea of Google's PageRank algorithm is that pages are important when linked by other important pages. Therefore, PageRank views the web as a graph and uses a weight spreading algorithm to calculate the importance of the pages. FolkRank adopts this idea and assumes that a resource is important if it is tagged with important tags from important users. As a first step, a given folksonomy $\mathbb{F} = (U, T, R, Y)$ is converted into an undirected tripartite graph $\mathbb{G}_\mathbb{F}$, where the set of nodes $V = U \dot\cup T \dot\cup R$ and the set of edges E and their weights is determined by the elements of Y.

Note that $\mathbb{G}_\mathbb{F}$ is different from the directed unipartite web graph. Hotho et al. therefore propose the Folksonomy-Adapted PageRank (FA-PR) algorithm to compute a ranking of the elements and which also takes the weights of the edges into account[4]. Since $\mathbb{G}_\mathbb{F}$ is undirected, a part of the weight spread over an edge will flow back in each iteration.

Formally, the weight spreading function is $\overrightarrow{w} = dA\overrightarrow{w} + (1-d)\overrightarrow{p}$, where A is the row-stochastic version of the adjacency matrix of $\mathbb{G}_\mathbb{F}$, $\overrightarrow{w}$ is the vector containing the rank values for the elements of V, $\overrightarrow{p}$ a preference vector whose elements sum up to 1 and d a factor determining the influence of $\overrightarrow{p}$. When a non-personalized ranking of the elements of $\mathbb{G}_\mathbb{F}$ is to be computed, d can be set to 1. When the goal is to personalize the ranking (or support topic-specific rankings), more weight can be given to elements in $\overrightarrow{p}$ which correspond to the user preferences or a given topic. Similar to PageRank, Folksonomy-Adapted PageRank works by iteratively computing $\overrightarrow{w}$ until convergence is achieved.

The FolkRank algorithm finally computes $\overrightarrow{w}$ two times – one time including the user preferences and one time without them – and compares the differences between the rankings of the two $\overrightarrow{w}$ vectors. The "winners" of the inclusion of the preference vector therefore get higher rank values. Recommending tags for a given resource or user can be accomplished by taking the n elements with the highest rank values.

Overall, FolkRank has shown to lead to highly accurate results and even the more recent algorithms mentioned above are only slightly more accurate than FolkRank on some evaluation data sets. However, one of the major issues of FolkRank are the steep computational costs involved in the computation of recommendations. Note that while the non-personalized ranks can be computed in an offline phase, this is not manageable for the personalized ranking. For analysis purposes, we use the original Java implementation provided by the developers of FolkRank[5] and evaluated it on three Delicious data sets at different density. Computing a single recommendation list for this data set consisting

[3] For this reason we will simply denote a folksonomy as a quadruple $\mathbb{F} := (U, T, R, Y)$.
[4] Note that FolkRank is not limited to the calculation of weights for the tags but can also be used to compute weights of users and resources.
[5] http://www.kde.cs.uni-kassel.de/code

of about 36,000 thousand users, 70,000 bookmarks, 21,000 tags, and 7,000,000 assignments required about 20 seconds on a typical desktop PC (AMD Athlon II Dual Core, 2.9Ghz, 8GB Ram) when the maximal number of iterations is set to 10. When pre-computing the unbiased ranks, the running time is reduced to about 10 seconds on average. Note that, given that FolkRank always propagates the weights through the whole network, the non-personalized weights have to be re-computed (or at least updated on a regular basis) when new tag assignments are added to the system.

2.2 LocalRank

In order to address the issues of scalability and updates, we propose LocalRank, a new tag recommendation algorithm which in contrast to FolkRank computes the rank weights only based on the local "neighborhood" of a given user and resource. Instead of considering all elements in the folksonomy, LocalRank focuses on the *relevant* ones only. Given a folksonomy $\mathbb{F} = (U, T, R, Y)$, its representation as $\mathbb{G}_{\mathbb{F}}$, a user $u \in U$ and a resource $r \in R$, we first compute the following sets of relevant elements.

- $Y_u \subseteq Y$ is the set of all (u, t, r)-assignments of Y where u is the given user .
- Analogously, $Y_r \subseteq Y$ is the set of all (u, t, r)-assignments of Y where r is the given resource.
- The set of user-relevant tags T_u is defined to be the set of all tags appearing in the (u, t, r)-assignments of Y_u.
- The resource-relevant tags T_r are analogously defined as the set of tags from the assignments in Y_r.
- The overall set of relevant tags to be ranked by the algorithm is $T_u \cup T_r$.

Figure 1 visualizes the local neighborhood of a user and a resource as two subgraphs of $\mathbb{G}_{\mathbb{F}}$, constructed using the sets Y_u and Y_r. The side aspect is that the sets can be represented efficiently as a compact data structure in memory. Note that the two subgraphs can also be connected in $T_u \cap T_r$.

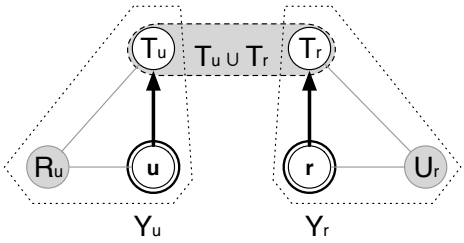

Fig. 1. Neighborhood of relevant tags for a given user-resource query.

The rank computation in LocalRank takes into account how often a certain tag was used by a user and how often a tag was attached to a resource. A similar

approach was presented as *most popular tags by user* and *most popular tags by resource* in [14]. Although the efficiency of the combination of these approaches – known as *most popular ρ-mix* – is comparable to our approach, the accuracy results, however, are worse than those of FolkRank. Note that in our approach, the popularity information is used as a factor in the rank computation of each tag in $T_u \cup T_r$. The (non-local) information how often other users have tagged other resources with these tags, however, is not exploited in LocalRank.

Rank computation and weight propagation in LocalRank is done similar to FolkRank but without iterations. The arrows in Figure 1 indicate the direction of the propagation of user and resource weights (see below) towards the tags.

In the FolkRank implementation the weight of a node v depends on the total number of nodes $|V|$ in the folksonomy and is set to $w = 1/|V|$. The frequency of the node's occurrence in Y is denoted as $|Y_v|$ and is defined as the number of (u, t, r)-assignments in Y, in which v appears. Overall, in FolkRank, the amount of weight spread by a node v to all its adjacent nodes is $w/|Y_v|$.

LocalRank, in contrast, approximates the weights for a given u and r with $w = 1/N$, where N is the total number of their neighbors in $\mathbb{G}_\mathbb{F}$. The amount of weight that is spread by the user and resource is calculated as $w/|Y_u|$ and $w/|Y_r|$ respectively.

In $\mathbb{G}_\mathbb{F}$, both algorithms calculate the weight gained by a node x by multiplying the spread weight $w/|Y_v|$ with the weight of the edge (v, x) which is equal to $|Y_{v,x}|$. While FolkRank repeatedly computes the weight gained by x for each (v, x) pair of nodes, LocalRank computes it once for each tag t in $T_u \cup T_r$.

The rank of each $t \in T_u$ is calculated as follows:

$$rank(t) = |Y_{u,t}| \times \frac{1/N}{|Y_u|} \tag{1}$$

The rank of tags in T_r is calculated similarly:

$$rank(t) = |Y_{r,t}| \times \frac{1/N}{|Y_r|} \tag{2}$$

Intuitively, we finally assume that tags that appear in both sets ($t \in T_u \cap T_r$) are on principle more important than the others and should receive a higher weight. Therefore we sum up the individual rank weights obtained from the two calculations.

LocalRank propagates the weight of the given user and resource nodes to all their adjacent tags. Therefore, it computes rankings for user and resource relevant tags and returns a list of tags and their ranks. The recommendation of tags can then be done by picking the top n elements with the highest rank values.

Note that in our evaluation we also experimented with a variation of the calculation scheme in which we introduced a weight factor to balance the importance of the different tag sets. The intuition behind this idea was that tags in T_r are generally more important than those in T_u because they already describe the resource. Elements of T_u capture the popularity of a tag with the particular

user and should have less importance as they are not necessarily meaningful to the resource. A similar approach to balancing the influence of user and resource related tags was presented in [14]. The experiments however showed that the introduction of such a weight factor did not help to further improve the results.

3 Evaluation

3.1 Data sets

In order to evaluate our approach both with respect to accuracy and run-time behavior, we ran tests on different versions of the Delicious data set, which is also used by many other researchers in the area of data mining and tag recommendation.

Delicious is a "social bookmarking tool", where users can manage collections of their personal web bookmarks, describe them using keywords (tags) and share them with other users. For our experiments, we used a data set of users, bookmarks and tags provided on courtesy of the DAI-Labor[6], which in its raw version contains more then 400 million tags applied to over 130 million bookmarks by nearly 1 million users.

In order to compare our work with previous work, we first extracted a smaller subset of manageable size from the large data set which included only the tag assignments posted between July 27 and July 30, 2005. By recursively adding tag assignments posted prior to July 27 for all users and resources present in the subset, a "core folksonomy" was constructed (as was also done in [15]). After this initial extraction step, we also applied p-core preprocessing to the data set. This preprocessing step guarantees that each user, resource, and tag occurs in at least k posts. That way, infrequent elements are removed from the folksonomy, thus reducing potential sources of noise in the data. At the same time, the *density* of the data is increased. Varying the p-core level therefore helps us to analyze the predictive accuracy of our methods at different density levels. In summary, experiments have been run on the three p-core levels 1, 5, and 10. As suggested in literature we removed for the p-core 5 and p-core 10 data sets all posts that had more than 30 tags, as they usually are spam.

	p-core 1	p-core 5	p-core 10
Users	71,756	48,471	36,486
Tags	454,587	47,984	21,930
Resources	3,322,519	169,960	70,412
Y-assign.	17,802,069	8,963,895	7,157654

Table 1. Data sets used in experiments.

[6] http://www.dai-labor.de/en/irml/datasets/delicious

3.2 Evaluation procedure

We use the *LeavePostOut* evaluation procedure described in [15], a variant of leave-one-out hold-out estimation.

For all preprocessed folksonomies, we first created a subset $\tilde{U}$ consisting of 10% randomly chosen users from U (the test set). For each user in $\tilde{U}$, we pick one of the user's posts randomly. A post p is a tuple $(u, r, tags(u, r))$, where $tags(u, r) := \{t \in T \mid (u, t, r) \in Y\}$ is the set of tags associated with the post. The task of the tag recommender consists of predicting a set of tags $\tilde{T}(u, r)$ for p based on the folksonomy $\mathbb{F} \setminus \{p\}$.

The predictive accuracy is determined using the usual information retrieval metrics *precision* and *recall*:

$$precision(\tilde{T}(u,r)) = \frac{|tags(u,r) \cap \tilde{T}(u,r)|}{|\tilde{T}(u,r)|} \tag{3}$$

$$recall(\tilde{T}(u,r)) = \frac{|tags(u,r) \cap \tilde{T}(u,r)|}{|tags(u,r)|} \tag{4}$$

The F1 metric, finally, is computed as the harmonic mean of precision and recall. The size of $\tilde{T}(u, r)$, that is, the length of the recommendation list, influences precision and recall. Longer recommendation lists naturally lead to higher recall values and lower precision. In the experiments, we therefore varied the length of the recommendation lists n from 1 to 20[7].

We used the following other parameters in our experiments. For FolkRank, we used the parameters suggested in [15] and set the weight parameter d to 0.7. The parameter ϵ is used in FolkRank as an indicator of reaching convergence. This means that no further iterations were made and the results were returned when the sum of all weight changes was less than 10^{-6}. As suggested in [15] we set the maximum number of iterations to 10 as an alternative stop condition.

3.3 Accuracy results

Figures 2 to 4 show the accuracy results for the different p-core levels. On the left hand side of the figures, we plot precision and recall values for the different recommendation list lengths. At the right hand side, the values of the F1 measure are shown for recommendation lists of varying length.

Regarding the F1 measure, no strong differences between FolkRank and our LocalRank metric can be observed for all data sets. On the p-core 1 data set, LocalRank is slightly better on the overall F1 measure. A closer look reveals that LocalRank achieves higher precision and recall values for list lengths of $n > 11$. LocalRank also leads to slightly better values than FolkRank with respect to both measures for the p-core 5 data set (Figure 3) and for list lengths

[7] Note that for the p-core level 1 folksonomy and also for the p-core level 5 folksonomy, the average number of tags per resource is below 20 (3 for p-core 1, 17 for p-core 5), which means that a precision of 100% cannot be achieved.

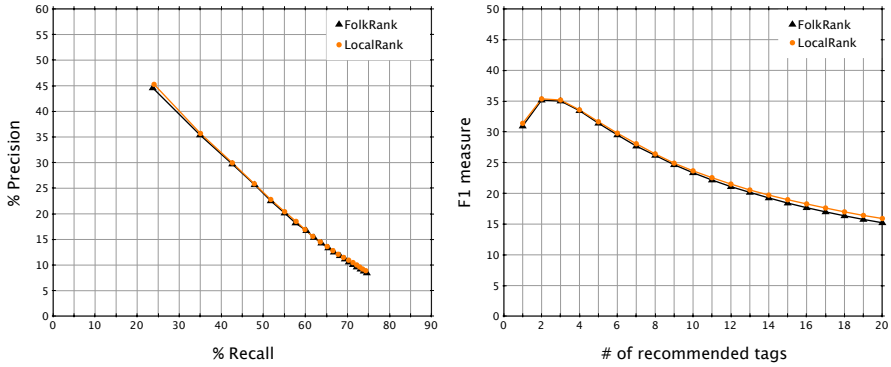

Fig. 2. Results for the p-core level 1 data set.

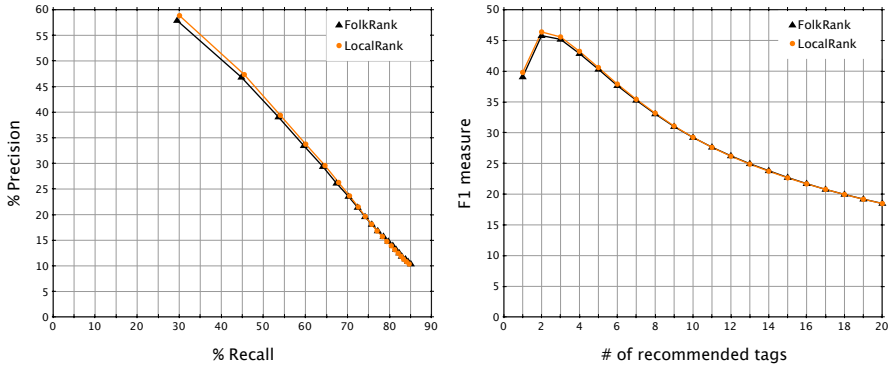

Fig. 3. Results for the p-core level 5 data set.

$n < 8$. The results for the p-core 10 data set are nearly identical for all evaluated recommendation list lengths, see Figure 4.

We conducted a sign test to analyze whether the observed differences are statistically significant [16]. For the p-core 5 and p-core 10 data sets, no significant differences regarding the obtained F1 measure for the two algorithms could be observed for all list lengths. For the largest and most realistic p-core 1 data set, however, LocalRank's F1 values are significantly higher ($p < .05$) for list lengths greater than 11. Overall, we therefore conclude that LocalRank is mostly on a par with FolkRank with respect to predictive accuracy on the Delicious data set at the examined p-core levels and even outperforms FolkRank in certain situations on low-density data sets.

We are aware that in very recent works new algorithms have been proposed which outperform FolkRank's predictive accuracy on certain data sets, collected for example from BibSonomy[8]. Gemmel et al. in [11] for example evaluate their hybrid approach on a p-core 20 data set collected from Delicious and observed

[8] http://www.bibsonomy.org

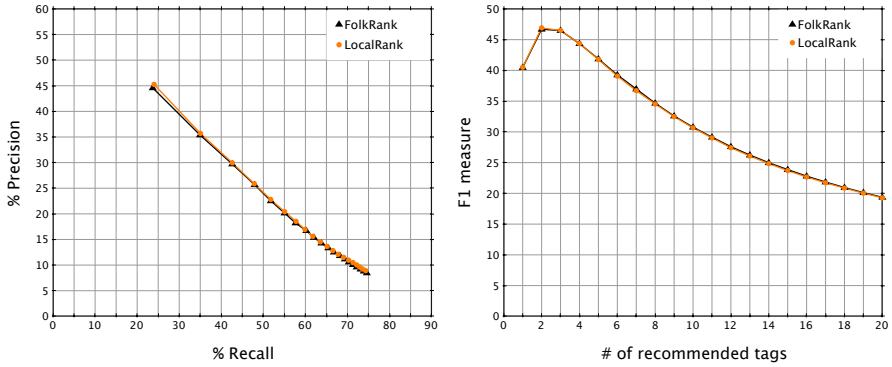

Fig. 4. Results for the p-core level 10 data set.

an improvement over FolkRank. This more recent and very dense data set (p-core 20), which also involved manual selection of users and tags was however not available to us, so that a direct comparison was not possible. Rendle et al. in [17] compare their tensor factorization approach with FolkRank on a very small BibSonomy data set and could show that for longer top-n recommendation lists, their approach is slightly better on the F1 measure. Overall, we view FolkRank therefore still as one of the state-of-the-art techniques for tag recommendation and use it as a baseline for comparison because most current literature refers to it as a baseline. The availability of the source code is also a reason to chose FolkRank in order to ensure a fair comparison between algorithms.

3.4 Run-time efficiency results

As mentioned above, because of FolkRank's approach to propagate weights over the full folksonomy for each query, the algorithm suffers from scalability problems which are mentioned also in [6] and [11].

Time measurements. Table 2 shows the average time needed for generating one recommendation list for the different p-core levels of the Delicious data sets. Note that with the Java-based version of the original FolkRank implementation from [4], more than 20 seconds are required for generating one single recommendation list using the above-described hardware configuration. As described in Section 2.1, FolkRank computes the rank vector $\overrightarrow{w}$ using the Folksonomy-Adapted PageRank (FA-PR) two times: with and without the preference vector. The first two columns of Table 2 show the computation time needed for these two phases. When we assume that the folksonomy does not change, the non-biased preference weights can be computed in advance and do not have to be re-computed for each recommendation. When relying on this re-use the computation time for FolkRank can be cut by about 50%.

Implementation and memory requirements. Similar to the implementation of FolkRank, our implementation of LocalRank is memory-based, that is, all the

	FA-PR w. preferences	FA-PR w/o preferences	FolkRank total	LocalRank
p-core 1	18,774	20,336	39,110	< 1
p-core 5	15,320	16,959	32,279	< 1
p-core 10	9,390	10,466	19,856	< 1

Table 2. Running times for recommendations in milliseconds.

required data is kept in memory. Actually, the time needed for the calculation of a recommendation list is on average constantly below one millisecond and does not increase when the size of the folksonomy increases. Beside the lower computational complexity of the neighborhood-based LocalRank algorithm itself, the more or less constant access time is made possible through a compact in-memory representation of the data and a pre-processing step at startup. In the pre-processing step, simple statistics such as $|Y_u|$, $|Y_r|$ and the number of neighbors for each user and resource are pre-computed. In addition, two adjacency lists are constructed that represent the graph structure and are required for the weight propagation step: one stores the information which user posted which tags, the other one contains information about the tags attached to each resource. Once the pre-processing step is performed, the generation of recommendation lists at run-time is based on simple arithmetical operations based on the data which are organized in lookup tables. Note that when new data comes in, the lookup tables can be very quickly updated because only local changes in the "neighborhood" of the newly added elements have to be made.

The required overhead in terms of additionally required memory is limited. For the simple counting statistics (e.g., number of assignments per tag) 4 integer arrays with a total size of $2*|U|+2*|R|$ are required. Two further hash maps are used to store the weights $|Y_{u,t}|$ and $|Y_{r,t}|$ of existing user/tag and resource/tag combinations in $|Y|$. Finally, the two adjacency lists are of length $|U|$ and $|R|$, where each list entry points to its assigned tags, the total number of which is $|T|$. Overall this means that $|Y|$ pointers to elements of T are required.

Comparison with other approaches. Based on our compact in-memory representation, even the p-core 1 data set can be kept in memory. Note that for example in the work by Gemmel et al. [11] "due to memory and time constraints" only a 10% fraction of a given Delicious data set was used. This data set was by the way the largest one in their evaluation and with 700,000 tag assignments, which is more than twenty times smaller than the p-core 1 data set used in our experiments. Note that for even larger data sets, one additional implementation option for LocalRank would be to store the most memory-intensive adjacency lists on disk in a (NoSQL) database. Typical database lookups with the given hardware configuration and data volumes usually take a few milliseconds per query. A prototypical implementation of a disk-based recommender for very large folksonomies is part of our current work.

Another work which reports prediction run times is [6]. Here, Rendle et al. compare the run times of their tensor factorization approach with FolkRank. After a linear time learning phase, their algorithm makes predictions only based on the learned model. The needed prediction time depends only on the relatively

small number of factorization dimensions for users, resources, and tags as well as the number of tags $|T|$. A characteristic of their method is that it achieves better accuracy results when the model contains more dimensions (64 and 128) but is not accurate as FolkRank when the number of dimensions is lower (e.g., 8 or 16). In their paper, a graphical illustration with no exact number of running times is given. Running times range from nearly zero for the low-dimensional case up to about 10 or 15 milliseconds for the 64-factor model. Unfortunately, no numbers are given for the most accurate 128-dimensional model. While their implementation based on Object-Pascal very clearly outperforms their C++ implementation of FolkRank, the data sets taken from BibSonomy and last.fm[9] used in their evaluation are comparably small (2,500 and 75,000 assignments). The number of assignments in $|Y|$ used in their experiments is less than a 1% of our data sets. Unfortunately, also no information about the time needed to train the model (in particular for the higher-dimensional case) is given. Overall, while some accuracy improvements over our LocalRank method can be achieved using the approach described in [6] when a high-dimensional model is learned, it remains partially unclear how their approach scales to larger problem sizes both with respect to training time and prediction time.

In [13], a clustering-based, probabilistic approach for "real-time tag recommendation" is proposed and evaluated on data sets derived from Delicious and CiteULike[10]. The approach is based on a two-stage framework consisting of a learning phase and an online tag recommendation phase. The authors report running times of about a bit more than 1 second that are required to determine suitable tags for a given document on a server machine with 3GHz. Compared to our evaluation, their data set obtained from Delicious is very small (218,088 tags) when compared to the 17 million tags used in our p-core 1 data set. Unfortunately, the authors of [13] do not compare the accuracy of their approach with the one of FolkRank but with a relatively simple method based on vector similarity.

4 Summary and outlook

In this paper we proposed LocalRank, a runtime-efficient tag recommendation algorithm, which despite its simplicity is capable of generating highly-accurate tag recommendations in real-time and even slightly outperforms FolkRank on the Delicious p-core level 1 data set. Compared to other approaches, LocalRank is not only quicker but also allows us to process larger data sets. Finally, from a practical perspective, our algorithm is also very easy to implement.

Our future work includes the analysis of the algorithm on further data sets in order to determine whether it is sufficient also for other Social Tagging platforms to consider only the neighborhood of a given user-resource recommendation query. From an algorithmic perspective, we are currently working on an

[9] http://www.last.fm
[10] http://www.citeulike.org

algorithm variant in which the "depth" of the weight-spreading process can be increased, for example to the second or third level, without increasing the prediction times too much.

Beyond that, we plan to develop a disk-based implementation of the algorithm, e.g., based on a database system, in order to analyze how massive tagging data can be processed in an efficient and scalable manner.

References

1. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender Systems - An Introduction. Cambridge University Press (2010)
2. Sen, S., Harper, F.M., LaPitz, A., Riedl, J.: The quest for quality tags. In: Proc. ACM GROUP'07, Sanibel Island, Florida, USA (2007) 361–370
3. Begelman, G., Keller, P., Smadja, F.: Automated tag clustering: Improving search and exploration in the tag space. In: Proc. Collaborative Web Tagging Workshop at WWW'2006, Edinburgh, Scotland (2006)
4. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Proc. ESWC'2006, Budva, ME (2006) 411–426
5. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks **30**(1-7) (1998) 107–117
6. Rendle, S., Balby Marinho, L., Nanopoulos, A., Lars, S.T.: Learning optimal ranking with tensor factorization for tag recommendation. In: Proc. ACM SIGKDD'09, Paris, France (2009) 727–736
7. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: A unified framework for providing recommendations in social tagging systems based on ternary semantic analysis. IEEE Trans. Knowl. Data. En. **22** (2010) 179–192
8. Krestel, R., Fankhauser, P., Nejdl, W.: Latent dirichlet allocation for tag recommendation. In: Proc. ACM RecSys'09, New York, USA (2009) 61–68
9. Hu, M., Lim, E.P., Jiang, J.: A probabilistic approach to personalized tag recommendation. In: Proc. IEEE SocialCom'10, Minneapolis, MN, USA (2010) 33–40
10. Bundschus, M., Yu, S., Tresp, V., Rettinger, A., Dejori, M., Kriegel, H.P.: Hierarchical bayesian models for collaborative tagging systems. In: Proc. IEEE ICDM'09, Miami, Florida, USA (2009) 728–733
11. Gemmell, J., Schimoler, T., Mobasher, B., Burke, R.: Hybrid tag recommendation for social annotation systems. In: Proc. ACM CIKM'10, Toronto (2010) 829–838
12. Chirita, P.A., Costache, S., Nejdl, W., Handschuh, S.: P-tag: Large scale automatic generation of personalized annotation tags for the web. In: Proc. WWW'07, Banff, Alberta, Canada (2007) 845–854
13. Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W.C., Giles, C.L.: Real-time automatic tag recommendation. In: Proc. SIGIR'08, Singapore (2008) 515–522
14. Jäschke, R., Marinho, L., Hotho, A., Lars, S.T., Gerd, S.: Tag recommendations in social bookmarking systems. AI Commun. **21** (December 2008) 231–247
15. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Proc. PKDD'07, Warsaw (2007) 506–514
16. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7** (2006) 1–30
17. Rendle, S., Lars, S.T.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proc. ACM WSDM'10, New York, USA (2010) 81–90