Article:
Doerfel, S., Jäschke, R. and Stumme, G. (2016) The Role of Cores in Recommender Benchmarking for Social Bookmarking Systems. ACM Transactions on Intelligent Systems and Technology, 7 (3). 40. 40:1-40:33. ISSN: 2157-6904

https://doi.org/10.1145/2700485

The Role of Cores in Recommender Benchmarking for Social Bookmarking Systems

STEPHAN DOERFEL, KDE Group, ITeG, University of Kassel, Germany
ROBERT JÄSCHKE, L3S Research Center, Hannover, Germany
GERD STUMME, KDE Group, ITeG, University of Kassel, Germany and L3S

Social bookmarking systems have established themselves as an important part in today's web. In such systems, *tag recommender systems* support users during the posting of a resource by suggesting suitable tags. Tag recommender algorithms have often been evaluated in offline benchmarking experiments. Yet, the particular setup of such experiments has rarely been analyzed. In particular, since the recommendation quality usually suffers from difficulties like the sparsity of the data or the cold start problem for new resources or users, datasets have often been pruned to so-called *cores* (specific subsets of the original datasets) – however without much consideration of the implications on the benchmarking results.

In this paper, we generalize the notion of a core by introducing the new notion of a *set-core* – which is independent of any graph structure – to overcome a structural drawback in the previous constructions of cores on tagging data. We show that problems caused by some types of cores can be eliminated using set-cores. Further, we present a thorough analysis of tag recommender benchmarking setups using cores. To that end, we conduct a large-scale experiment on four real-world datasets in which we analyze the influence of different cores on the evaluation of recommendation algorithms. We can show that the results of the comparison of different recommendation approaches depends on the selection of core type and level. For the benchmarking of tag recommender algorithms, our results suggest that the evaluation must be set up more carefully and should not be based on one arbitrarily chosen core type and level.

CCS Concepts: •**General and reference** → **Evaluation;** *Experimentation; Measurement;* •**Information systems** → **Recommender systems;** •**Human-centered computing** → *Social recommendation; Social tagging; Social tagging systems;*

Additional Key Words and Phrases: Recommender; Core; Benchmarking; Graph; Evaluation; Preprocessing

ACM Reference Format:
Stephan Doerfel, Robert Jäschke and Gerd Stumme, 2013. The Role of Cores in Recommender Benchmarking for Social Bookmarking Systems. *ACM Trans. Intell. Syst. Technol.* V, N, Article XXXX (July 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2700485

1. INTRODUCTION

Recommender systems have become a vital part of the social web, where they assist users in their content selection by pointing to personalized sets of resources. Such systems often have to deal with sparse data since only little or nothing is known about many users or items. Alongside work that specifically tackles this task, in the eval-

Part of this work was funded by the DFG in the project "Info 2.0 – Informationelle Selbstbestimmung im Web 2.0".
Author's addresses: S. Doerfel and G. Stumme, Knowledge & Data Engineering Group (KDE) at the Interdisciplinary Research Center for Information System Design (ITeG) at the University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany; R. Jäschke, L3S Research Center, Appelstraße 4, 30167 Hannover, Germany;
 2157-6904/2015/07-ARTXXXX $15.00
DOI: http://dx.doi.org/10.1145/2700485

uation of recommender algorithms it is common to focus on a denser subset of the data [Sarwar et al. 2001] that provides enough information to produce helpful recommendations. For data that can be modeled as a graph, a commonly used technique are *generalized cores* [Batagelj and M. Zaveršnik 2002] which comprise a dense subgraph in which every vertex fulfills a specific constraint, e.g., the degree of each node is above a certain threshold, the so-called *core-level*. However, the influence of these cores on the evaluation of recommendation algorithms has not been analyzed so far.

In this paper, we investigate cores that have been used in the evaluation of *tag recommender systems*. Tag recommenders are useful in *social bookmarking systems* where users post resources (like bookmarks for websites, scientific publications, videos, or photos) and assign arbitrary keywords (*tags*) to them. During the posting process a recommender system typically suggests appropriate tags for annotation. The tag recommendation task is thus to find suitable tags for a given user and resource.

Previously, tag recommender algorithms have often been evaluated on a special generalized core of the raw dataset. Although the use of cores is quite common in tag recommender benchmarking (cf. Section 3), it has rarely been analyzed how the choice of

— *core type* (i.e., the method to construct the core as a subset of the original dataset; we will recall and introduce various core types in Section 2),
— *core level* (i.e., the threshold that is imposed on some property of each data point to construct the subset; see Section 2, Definitions 2.1 and 2.2),

or simply the process of constructing cores influences the results of such experiments. In fact, the choice of these setup parameters has been rather diverse in previous tag recommender benchmarking experiments (see the related work in Section 3 for details and for examples from the literature). Especially, the core level is often set ad-hoc – without a motivation for the particular choice – to such values as 2, 5, or even 100, or to dataset-dependent thresholds. In our experiments in Section 4 we show on real world datasets, that the choice of core type and core level indeed has an impact on a benchmarking's ranking of recommender algorithms. In fact, different experiments on different setups can lead to contradictory results. Thus, much like the choice of the evaluation metric or the sampling of training and test data, the core type and level are important aspects of an experimental setup. During the evaluation of different recommendation algorithms or during parameter optimization of such algorithms, it is therefore worthwhile to experiment with several cores and also to use the raw datasets (the unrestricted datasets). Moreover, the choice of particular core-levels should be motivated by the use-case and comparisons of results from different experiments must consider the different core setups in each experiment.

While the previously used cores indeed yield denser graphs, they also come with the unpleasant property of *diminishing posts*, i.e., a post of the raw dataset – consisting of a user, a resource, and several tags – might still occur in the core but with fewer tags. Thus, the core construction not only reduces the number of posts in the dataset, but modifies the posts themselves. For such "diminished" posts the recommendation problem becomes more difficult as recommended tags will not be considered good recommendations even when they actually belong to the original post but have been removed from that post by the core construction.

We show that cores of real-world datasets indeed contain many such diminished posts and that different recommender algorithms often yield lower quality scores on such posts than on those that still remain intact with all their tags in the core. To overcome this structural problem we first generalize the notion of *generalized cores* [Batagelj and M. Zaveršnik 2002] even further to yield *set-cores* (Section 2). In contrast to generalized cores, these do not require a graph structure and can be applied to any kind of dataset in which the entities have some measurable property. We show

that set-cores have similar properties as generalized cores, describe a construction algorithm, and prove its correctness. We then construct a new kind of core – a set-core – for social bookmarking systems which guarantees to leave all remaining posts intact (undiminished).

In Section 3 we discuss how the experimental setup and evaluation using cores has been handled in previous work on tag recommendations. We then choose a common setup and describe in Section 4 several experiments on four publicly available real-world datasets to investigate the influence of cores on the results of recommender systems benchmarking. We discuss the results of these experiments in Section 5 where we show how different cores can lead to contradictory results in the comparison of algorithms. We also point to a peculiarity that arises from using any type of core in that setup. To summarize, the contributions of this paper are fourfold:

— We generalize the notion of generalized cores to *set-cores* and introduce new cores for tagging data of social bookmarking systems to eliminate the particular anomaly of diminished posts in previously used cores.
— We present a thorough investigation of the influence of cores on the results of tag recommender benchmarking experiments and confirm that different choices of core type and level can indeed yield different results.
— We discuss potential pitfalls of the use of cores in recommender evaluation.
— We provide recommendations for the use of cores in future tag recommender benchmarking experiments.

2. CORES OF GRAPHS AND SETS

Before we discuss the influence of cores within the benchmarking framework for tag recommendations, we deal with the notion of a core itself. In social bookmarking systems (often also called *tagging systems*, see Section 2.3 for a formal definition of their data structure), the cores that have been used so far have the unpleasant property of diminishing posts by removing tags. In this section we present a solution to that problem by introducing post-set-cores. To accomplish that we first recall the notion of generalized cores of a graph and then extend it to arbitrary sets by introducing set-cores (Section 2.1). We present examples in Section 2.2 that illustrate different set cores and that demonstrate some advantages of set cores. We then discuss cores for tagging data in Section 2.3 where we recall the definitions of cores previously used for the evaluation of tag recommenders and we introduce a new core construction using set-cores to overcome the issue of diminished posts.

Notation. In the remainder of the paper we make use of the following usual notation:

— A graph $G = (V, E)$ is given as a set V of *vertices* together with a set E of (undirected) *edges*. Hereby each edge connects two – or more (in the case of hypergraphs) – vertices.
— $E|_C$ denotes the restriction of the set E to a subset $C \subseteq V$, i.e., to the edges from E between vertices from C.
— With $\mathfrak{P}(S)$ we denote the *power set* of a given set S.

Batagelj and Zaveršnik presented the notion of p-cores in [Batagelj and M. Zaveršnik 2002], which by itself is a generalization of the original cores introduced in [Seidman 1983]. In the sequel, we refer to their construction as *graph-p-cores* to better distinguish them from the new notion of *set-P-cores* which we introduce later in this section. The idea of graph-p-cores is to restrict a given graph by removing all nodes for which a particular quantity p (e.g., the vertex degree) does not exceed a given threshold l called the *core level*. The graph-p-core is then the largest possible subgraph such that all its vertices have the property p (measured in that subgraph) above the threshold:

Definition 2.1 (*Graph-p-Core [Batagelj and M. Zaveršnik 2002]*). Let $G = (V, E)$ be a graph, $l \in \mathbb{R}$, and p a vertex property function on G, i.e., $p\colon V \times \mathfrak{P}(V) \to \mathbb{R}\colon (v, W) \mapsto p(v, W)$. A subgraph $H = (C, E|_C)$ induced by the subset of vertices $C \subseteq V$ is called a *graph-p-core at level l*, iff $l \leq p(v, C)$, for all $v \in C$ and H is a maximum subgraph with this property. A core of G with a maximum level l such that it is not empty is called the *main core* of G.

An example for a property function p is the vertex degree in each subgraph – in fact, the original core definition of Seidman [Seidman 1983] uses just that function instead of an arbitrary function p.

The function p is called *monotone* if and only if it fulfills

$$W_1 \subseteq W_2 \subseteq V \implies \forall v \in W_1 \colon p(v, W_1) \leq p(v, W_2)\,.$$

[Batagelj and M. Zaveršnik 2002] prove that for every monotone vertex property function p a graph-p-core is uniquely determined at each level l and that it can be computed by iteratively removing vertices v from the vertex set W (starting with $W := V$) that do not fulfill $l \leq p(v, W)$. The monotonicity assumption is a mild requirement, as typical vertex property functions (like the degree) naturally fulfill it.

Note, that in Definition 2.1, the value $p(v, W)$ of the function p is dependent both on the vertex v and on the vertices W of the subgraph that it is evaluated on. For example, the degree of a vertex in a subgraph of G can be smaller than its degree in G. Alternatively, the function p in Definition 2.1 can be replaced by a set of functions

$$\{p_W \colon W \to \mathbb{R} \mid (W, E|_W) \text{ is a subgraph of } G\}\,.$$

A drawback of Definition 2.1 is that it is not possible to model simultaneous restrictions of several properties (e.g., that a vertex has at least in-degree l and at least out-degree m). This can be desirable, for example, when vertices are entities in a tagging system and we want to require that each user has at least l posts, each resource occurs in at least m posts, and each tag has been used at least n times. For the case of two thresholds on a bipartite graph a solution was given in [Ahmed et al. 2007] by the introduction of *graph-(p, q)-cores*. The *set-P-core*, which we introduce next, allows us to enforce different thresholds in a more general way by using an arbitrary partially ordered set[1] as a range instead of only the real numbers.

Another drawback of Definition 2.1 is the dependence on a graph structure. While this is quite universal already – as almost any kind of data can be modeled as a graph – it is not always particularly intuitive to construct a graph such that a graph-p-core can be constructed. In contrast, set-P-cores can be constructed on arbitrary sets.

2.1. Generalization

In the following, we present the definition of a *set-P-core*, prove its uniqueness and describe a construction. A set-P-core can be constructed on some arbitrary set S where for each element of S some property can be measured. Again, a threshold l is imposed to restrict the set to such elements where that property is above the threshold. In contrast to graph-p-cores, the level l must not necessarily be a real number but must simply belong to some partially ordered set L (like, for example, the space $\mathbb{R}^n$). Thus the properties are also no longer required to yield a single number, allowing to enforce multiple property restrictions simultaneously. Given a set S, the set-P-core is the largest subset, such that for each element of S the chosen property functions P yield a value that is larger than a fix level l. This is stated more formally in the next definition:

[1] A set L together with a binary relation $R \subseteq L \times L$ is a *partially ordered set*, iff R is reflexive, transitive, and anti-symmetric.

Definition 2.2 (*Set-P-Core*). Let S be a set, L a partially ordered set with the order relation $\leq$, $l \in L$, and P a set of property functions $p_{\tilde{S}} \colon \tilde{S} \to L$ with $s \mapsto p_{\tilde{S}}(s)$ for each $\tilde{S} \subseteq S$. A subset C of S is said to *have the l-property w.r.t. P*, if it satisfies the condition $l \leq p_C(c)$ for all $c \in C$. The subset C is called *set-P-core at level l of S*, iff it is a maximum subset of S with the l-property.

We simply say that a subset of S has the l-property, if the choice of P is clear from the context. Note, that in contrast to the generalized graph-p-cores in [Batagelj and M. Zaveršnik 2002], Definition 2.2 does neither require any kind of graph structure, nor a linearly ordered set (like the real numbers for graph-p-cores). For two elements a and b of the partially ordered set L, we denote by $a \not\leq b$ that a is not smaller than or equal to b, i.e., that either a is larger than b or that a and b are incomparable.

It is easy to see that graph-p-cores are special set-P-cores: In the notions of Definitions 2.1 and 2.2, we set $S := V$ (the vertex set of G), $L := \mathbb{R}$ and use the set of p-functions as P such that $p_{\tilde{S}}(s) := p(s, E|_{\tilde{S}})$. A trivial observation is that the empty set $\emptyset \subseteq S$ has the l-property w.r.t. P for any P and $l \in L$ and thus any set S has at least one subset with the l-property.

Similar to graph-p-cores, the unique existence of the set-P-core is guaranteed as long as the property functions in P satisfy a mild monotonicity requirement: in each subset of the original set, for each element, the property measured by the according map in P is lower than or equal to the according value measured in the original set. Furthermore, set-P-cores are nested in the sense that increasing the level l yields a smaller core. These properties are formalized and proven in the following theorem:

THEOREM 2.3. *Given S, L and P as in Definition 2.2. If the functions in P are monotone in the sense that*

$$\tilde{S}_1 \subseteq \tilde{S}_2 \subseteq S \Longrightarrow \forall s \in \tilde{S}_1 \colon p_{\tilde{S}_1}(s) \leq p_{\tilde{S}_2}(s)$$

holds, then for $l, l_1, l_2 \in L$ hold:

(*1*) *The union of subsets of S with the l-property has the l-property.*
(*2*) *There exists exactly one set-P-core at l.*
(*3*) *The set-P-cores are nested, i.e., if $l_1 \leq l_2$, then the set-P-core at l_2 is contained in the set-P-core at l_1.*

PROOF. We start with the first property: Let I be an index set and $\tilde{S}_i$ $(i \in I)$ be subsets of S with the l-property and U their union. For $s \in U$, there is some $i \in I$ such that $s \in \tilde{S}_i$. By monotonicity of P we have $l \leq p_{\tilde{S}_i}(s) \leq p_U(s)$ and thus U has the l-property. The second property follows directly from the first, with the set-P-core being the union of all subsets of S having the l-property (and thus obviously being maximal). For the third property, let C_1 and C_2 be the respective set-P-cores at l_1 and l_2. Then $l_1 \leq l_2 \leq p_{C_2}(s) \leq p_{(C_1 \cup C_2)}(s)$. Thus $(C_1 \cup C_2)$ has the l_1 property, and by the maximality of C_1 as core at level l_1 follows $C_1 = (C_1 \cup C_2)$ and thus $C_2 \subseteq C_1$. □

We have now established a generalized notion of cores and can reuse the simple construction algorithm from [Batagelj and Zaveršnik 2011] for such a set-P-core, given a finite set S (see Algorithm 1). The set-P-core can always be constructed simply by removing one element violating the l-property after another until the remaining set of elements satisfies the l-property. Note however, that it does not suffice to test each element only once, as the value of the property function depends both on the element and the (remaining) subset. Thus, through the removal of other elements, the value of the property function might have decreased (in comparison to the same value before that

removal) and thus be no longer larger than the threshold l. We prove the applicability of Algorithm 1 in the following theorem:

ALGORITHM 1: Naive set-P-core construction

Input: Dataset S, level l, monotone set of functions P.
Output: The set-P-core C of S at level l.
$C := S$;
while $\exists s \in C$ *such that* $l \not\leq p_C(s)$ **do**
  $C := C \setminus \{s\}$;
end

THEOREM 2.4. *Given S, L, $l \in L$, and P as in Definition 2.2 with P being a set of monotone functions. If S is finite, then Algorithm 1 returns the set-P-core at l of S.*

PROOF. Let D be the algorithm's result and let C be the set-P-core of S at l. The unique existence of C is already guaranteed by Theorem 2.3. From the algorithm it is clear that D has the l-property and is therefore a subset of C. Let further $s_1, s_2, \ldots, s_n$ be the elements of $S \setminus D$ in the order of their deletion by the algorithm. Assume $D \subset C$. Then we can select an index i with $1 \leq i \leq n$ such that for all j with $1 \leq j < i$, s_j is in $S \setminus C$ but s_i is in C. We set $\tilde{S}_i := S \setminus \{s_1, s_2, \ldots, s_{i-1}\}$ and yield $l \not\leq p_{\tilde{S}_i}(s_i)$, since s_i was removed in the ith step of the algorithm. From the selection of i follows $C \subseteq \tilde{S}_i$ and thus by monotonicity of P we have $p_C(s_i) \leq p_{\tilde{S}_i}(s_i)$ and therefore $l \not\leq p_C(s_i)$. This is a contradiction to the l-property of C. We have thus established $D = C$ and conclude that the algorithm's result is the set-P-core at l of S. □

2.2. Examples

Our generalization allows us to transfer the concept of a core to arbitrary algebraic structures without constructing graphs. Although it is almost always possible to model a given dataset as a graph, it is not always convenient to impose a graph model. It is especially unpleasant when data is already modeled as a graph (like in the case of social bookmarking systems in the next section) but the graph does not allow the construction of a core in the desired way and thus a new graph would have to be introduced to support it. With set-cores, this is no longer an issue.

Before we leverage set-cores to construct cores for tagging data in the next section, we discuss a very simple example where data does not have to be modeled as a graph: a core that could be used in the evaluation of item recommendation algorithms. Let U be a set of users and I a set of items. Further, let $S \subseteq U \times I$ be the (user, item) co-occurrences (i.e., the relation denoting which items a user likes). Such a setting is demonstrated with a toy example in the first column of Table I, where six users co-occur with (e.g., have expressed interest or have bought) six items in 18 (user, item) co-occurrences.

Now, let P be a set of maps $p_{\tilde{S}}$ (for every $\tilde{S} \subseteq S$) with

$$p_{\tilde{S}} : \tilde{S} \to \mathbb{N} : (u,i) \mapsto \max\left(\left|\{j \in I \mid (u,j) \in \tilde{S}\}\right|, \left|\{v \in U \mid (v,i) \in \tilde{S}\}\right|\right) . \quad (1)$$

For a given level $l \in \mathbb{N}$, the set-P-core at l then contains all (user, item) co-occurrences from S such that each user occurs with at least l items or each item occurs with at least l users. Thus, at least one entity of each user-item-pair is frequent in the dataset. In the toy example in Table I, for each (user, item) co-occurrence the maximum of its user and item frequency is larger than or equal to two. Therefore, the set-P-core

Table I. A toy example for different cores in a user-item co-occurrence setting. The set U contains six users $u_1, u_2, \ldots, u_6$ and the set I contains six items $1, 2, \ldots, 6$. The first column shows the full dataset of users together with their co-occurring items. Each further column $A_1, A_2, B, \ldots, F$ shows a different restriction of that dataset. The functions to create these subsets are described in Section 2.2.

dataset	A_1	A_2	B	C	D	E	F
u_1: 1 2 3 4	u_1: 1 2 3 4	u_1: 1 2 3 4	u_1: 1 2 4	u_1: 1 2 3 4	u_1: 1 2 3 4	u_1: 1 2 3 4	u_1: 1 2 3 4
u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4	u_2: 1 2 4
u_3: 1 3 4	u_3: 1 3 4	u_3: 1 4	u_3:	u_3: 1 3 4	u_3: 1 3 4	u_3: 1 3 4	u_3: 1 3 4
u_4: 3 5 6	u_4: 3 5 6	u_4:	u_4:	u_4:	u_4: 3 5 6	u_4: 3 5	u_4: 3 5
u_5: 2 5	u_5: 2	u_5: 2	u_5:	u_5:	u_5:	u_5: 2 5	u_5:
u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4	u_6: 1 2 4

at level $l = 2$ is the full dataset. For $l = 3$ we obtain the dataset denoted by A_1 (second column) in which only the (user, item) co-occurrence $(u_5, 5)$ has been removed, as both user u_5 and item 5 occur in only two user-item-pairs. In this first example we can observe that the resulting core actually has a lower density (computed as the number of (user, item) co-occurrences divided by the number of possible user-item pairs) than the original dataset, since through the removal of the pair $(u_5, 5)$ neither a user nor an item have been removed completely from the dataset. This might reduce the computational complexity in an item recommender scenario (for algorithms that depend on the number of pairs) but usually, artificially introducing sparseness is not desirable. The next examples will show cores where the density rises. Furthermore, we will see in Section 4.1 that our core constructions yield an increase in density on all four real world datasets.

Increasing the level to $l = 4$ yields the core denoted by A_2 in Table I. Here all pairs are removed where user and item both occur in less than four pairs. Hereby, all pairs containing user u_4 and all pairs containing items 5 and 6 are eliminated. Thus, these three entities can be removed from the dataset completely. In comparison to the core for $l = 3$ we now indeed yield a dataset with higher density than the original dataset. A_2 is the main set-P-core, as for $l = 5$ the core vanishes since no user or item occurs in more than four pairs.

Using the minimum instead of the maximum in the definition of $p_{\tilde{S}}$ in Equation 1, results in a core containing (user, item) co-occurrences where both user *and* item are frequent, as here the smaller of the two frequencies – and thus both frequencies – must exceed the threshold l. In the toy example in Table I, the set-P-core for $l = 3$ is denoted by B. It can be constructed using Algorithm 1 by first removing the co-occurrences $(u_4, 5)$, $(u_5, 5)$, and $(u_4, 6)$, since items 5 and 6 both are not frequent. The pair $(u_5, 2)$ is removed since user u_5 is not frequent. Then the remaining co-occurrence of user u_4 – $(u_4, 3)$ – is removed, since after the elimination of $(u_4, 5)$ and $(u_4, 6)$ u_4 has become infrequent. Then all co-occurrences with item 3 and finally those of user u_3 must be removed. In the example, the set B is the main set-P-core since for level $l = 4$ all user-item-pairs would be removed from the dataset.

An example for a core, where different thresholds can be imposed on users and items, results from the maps $p_{\tilde{S}}$:

$$p_{\tilde{S}} \colon \tilde{S} \to \mathbb{N}^2 \colon (u, i) \mapsto \left(\left| \{ j \in I \mid (u, j) \in \tilde{S} \} \right|, \left| \{ v \in U \mid (v, i) \in \tilde{S} \} \right| \right) \tag{2}$$

together with a level $l := (l_u, l_i) \in \mathbb{N}^2$. This setting yields a core where each user occurs with at least l_u items and each item with at least l_i users. Thus, we have made use of two thresholds at the same time, which could not have been modeled with graph-cores. In the toy example in Table I, dataset C shows the $(3, 2)$-set-core. In contrast to the previous result B – where user and item both had to occur three times in the dataset – item 3 still has co-occurrences in the dataset. Note, that setting two thresholds l_u and l_i at the same time is not the same as first setting one threshold l_u on the users, then

Fig. 1. An example post from BibSonomy for the user ID 1015 and resource http://www.google.com/trends.

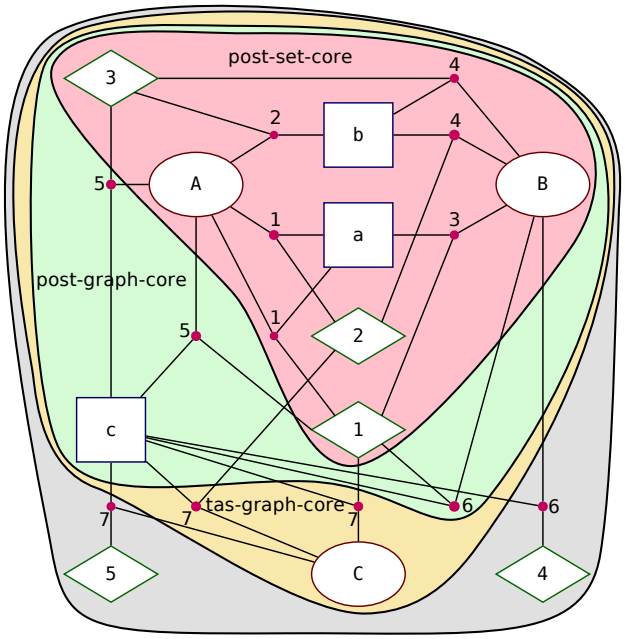
setting one threshold l_i on the items and taking the intersection of the resulting sets. The latter procedure would not necessarily yield a set where each user occurs at least l_u and each item at least l_i times. This is demonstrated in the toy example with the datasets D, E, and F: they contain the restricted datasets with $l_u = 3$ (D) and with $l_i = 2$ (E) as well as their intersection (F). We can observe that dataset F is different from C and that user u_4 violates the constraint on the user frequency by having only two co-occurrences instead of the required three and item 5 does not satisfy the lower bound on the item frequency as it is part of only one co-occurrence.

In these examples we have demonstrated the ability of set-cores to restrict datasets according to individual thresholds on different sets of entities, like in the latter example with one threshold for users and one for items. We have also seen the application of combined thresholds like the first two examples, using the maximum or minimum of user and item co-occurrences. Both aspects allow a great flexibility for the practitioner: Thresholds can be chosen individually for different entities and at the same time, combined thresholds can be imposed. For the latter, $\min$ and $\max$ are only simple examples: we could just as easily use sums, products, or other functions, as long as they comply with the monotonicity requirement in Theorem 2.3. Using the sum instead of $\max$ or $\min$ in the above example would impose a threshold for the combined popularity of user and item in a user-item pair. Finally, it is also possible to combine the maps of the different examples in Equations (1) and (2) (and thus yield maps $p_{\tilde{S}} \colon \tilde{S} \to \mathbb{N}^3$) to have individual requirements on users and items as well as a combined requirement for each user-item pair.

2.3. Cores of Folksonomies

We employ the use case of social bookmarking systems to demonstrate different types of cores. These cores are also the subject of the experimental investigation of tag recommender evaluation frameworks in the following sections. A *folksonomy* $\mathbb{F}$ is the data structure underlying social bookmarking systems and can be represented as a tuple $\mathbb{F} = (U, T, R, Y)$ where U is the set of users, T the set of tags, R the set of resources, and $Y \subseteq U \times T \times R$ a ternary relation between the three sets [Hotho et al. 2006]. An element (u, t, r) of Y – called *tag assignment* (or *tas*) – expresses the fact that user u tagged resource r with tag t. This structure $\mathbb{F}$ can be interpreted as a ternary hypergraph or as a three-dimensional Boolean tensor. Several tag assignments of one user u on one resource r are typically subsumed in a *post* p which is a tuple $p = (u, T_{ur}, r)$ with $T_{ur} = \{t \in T \mid (u, t, r) \in Y\}$ (with the requirement that $T_{ur} \neq \emptyset$). The perspective of a folksonomy as a *collection of posts* typically corresponds to the view the users have on the system (cf. Figure 1). Since every post contains at least one tag assignment, the number of tag assignments an entity (i.e., a user, a tag, or a resource) of a folksonomy belongs to is always at least as large as the number of posts the entity is part of.

Running Example. Figure 2 shows an exemplary folksonomy hypergraph with users A, B, C (drawn as ◯), resources a, b, c (□), and tags $1, 2, 3, 4, 5$ (◇) which are connected by thirteen tag assignments in seven posts (numbered 1 through 7). The hyperedges that represent the tag assignments are visualized by small circles (·) which are connected to the three vertices of each hyperedge. The number next to each circle de-



post	tag assignments (u, t, r)
1	$(A, 1, a), (A, 2, a)$
2	$(A, 2, b)$
3	$(B, 1, a)$
4	$(B, 2, b), (B, 3, b)$
5	$(A, 1, c), (A, 3, c)$
6	$(B, 1, c), (B, 4, c)$
7	$(C, 1, c), (C, 2, c), (C, 5, c)$

Fig. 2. A folksonomy toy example with a tas-graph-core, post-graph-core, and post-set-core. The table on the right lists the tag assignments that belong to each post.

picts the number of the post the tag assignment belongs to. The differently colored areas that enclose parts of the graph depict the various types of cores that are explained in the sequel.

2.3.1. The Tas-Graph-Core of a Folksonomy. We can regard the folksonomy $\mathbb{F} = (U, T, R, Y)$ as a hypergraph $G = (V, E) := (U \uplus T \uplus R, Y)$. Together with the level $l \in \mathbb{N}$ and the vertex property function

$$p\colon V \times \mathfrak{P}(V) \to \mathbb{N}\colon (v, W) \mapsto \begin{cases} |(\{v\} \times T \times R) \cap E|_W| & \text{if} \quad v \in U \\ |(U \times \{v\} \times R) \cap E|_W| & \text{if} \quad v \in T \\ |(U \times T \times \{v\}) \cap E|_W| & \text{if} \quad v \in R \end{cases} \tag{3}$$

that assigns to every $W \subseteq V$ and every $v \in W$ the number of tag assignments that v is part of in W, we get the *tas-graph-core at level l of the folksonomy* $\mathbb{F}$. It has the property that every user, tag, and resource is part of at least l tag assignments. Note, that for a tag to be part of a tas-graph-core at level l, it must have been used in at least l posts, while for a user (resource) it is sufficient to annotate (be part of) only a single post with at least l tags (cf. [Jäschke et al. 2008]).

Running Example. Figure 2 shows the *tas-graph-core* at level 2, in which every entity belongs to at least two tag assignments. The tag assignments $(B, 4, c)$ from post 6 and $(C, 5, c)$ from post 7 are lost because the tags 4 and 5 belong each only to the one corresponding tag assignment. Note, that the tas-graph-core does *not* have level 3, since the tag assignment $(C, 5, c)$ does not belong to the tas-graph-core and thus user C occurs in only two tag assignments.

2.3.2. The Post-Graph-Core of a Folksonomy. To circumvent the aforementioned problem, in [Jäschke et al. 2007] (and more formally in [Jäschke et al. 2008]) we defined another core which we call *post-graph-core* here (to distinguish it from the other types of cores).

Therefore, we define for $v \in W \subseteq V$ (using the same notation[2] as in Equation 3) a new vertex property function that counts for every entity of the folksonomy the number of *posts* (instead of tag assignments, as before) it is part of:

$$P\colon V \times \mathfrak{P}(V) \to \mathbb{N}\colon (v, W) \mapsto \begin{cases} |\{(v, T_{vr}|_W, r) \mid r \in R|_W, T_{vr}|_W \neq \emptyset\}| & \text{if} \quad v \in U \\ |\{(u, v, r) \in E|_W\}| & \text{if} \quad v \in T \\ |\{(u, T_{uv}|_W, v) \mid u \in U|_W, T_{uv}|_W \neq \emptyset\}| & \text{if} \quad v \in R \end{cases}$$

This definition intuitively violates the symmetry of the ternary structure of a folksonomy. This can be seen from the property function P, where – in contrast to the previous core – the value for tags ($v \in T$) is no longer defined analogously to the values for users and resources. This is because one post always contains exactly one user and exactly one resource but can have more than one tag. However, the post-graph-core more closely resembles the view of a folksonomy as 'a collection of posts' that collaborative tagging systems typically provide. The post-graph-core at level l has the property that each user, tag, and resource occurs in at least l posts. We illustrate in Section 3 that post-graph-cores have been frequently used in the evaluation of recommender systems for folksonomies.

Running Example. In the example in Figure 2 we can see that in the *post-graph-core* at level 2 every entity belongs to at least 2 posts. Due to the removal of the user C (which only belongs to post 7), all tag assignments from post 7 (and thus the post itself) are removed. Similarly, tag 4 is removed as it belongs only to post 6.

Diminished Posts in Tas-Graph-Cores and Post-Graph-Cores. In the previous two constructions, the core is computed by removing single tag assignments. Thus, from one post, several tag assignments can be removed, while others (of the same post) remain in the core. This rather unfortunate behavior is illustrated in Table II using a post from the BibSonomy *book* dataset which we use and describe in Section 4.1. The post (that is also shown in Figure 1) consists of five tag assignments in the original dataset. By restricting the data to a tas-graph-core or a post-graph-core, some of these tag assignments are omitted and the post is diminished. In the tas-graph-core at level 2, first the two rare tags "requetes" and "webmetrics" vanish from the post. At level 3 and also in the post-graph-cores, also the tag assignment with the tag "comparateur" is removed. The tags "statistics" and "trends" are well connected with other folksonomy entities through tag assignments in other posts. Thus they remain in the cores for several levels until the complete post vanishes from the dataset at levels higher than 12 for tas-graph-cores and levels higher than 5 for post-graph-cores.

Running Example. In our running example we can also observe a diminished post. In the constructions of both the tas-graph-core and the post-graph-core post 6 is diminished: tag assignment $(B, 1, c)$ still belongs to the core, while tag assignment $(B, 4, c)$ is removed. Thus post 6 has now only one tag, instead of the original two.

The use of set-cores now allows us to overcome the phenomenon of diminished posts by regarding posts as atomic entities. We call this new construction the *post-set-core* of a folksonomy:

2.3.3. The Post-Set-Core of a Folksonomy. Let S be the set of all posts in $\mathbb{F}$ and for some subset $\tilde{S} \subseteq S$ of posts let $\tilde{S}_u, \tilde{S}_t, \tilde{S}_r$ be the sets of posts in $\tilde{S}$, that a user u, a tag t, or a resource r occurs in, respectively. Note, that these can be empty sets, if the according

[2] $U|_W$, $R|_W$, $T|_W$, $T_{vr}|_W$, and $T_{uv}|_W$ denote the restrictions of U, R, T, T_{vr}, and T_{uv}, respectively, to the set $W \subseteq V$ – e.g., $U|_W := U \cap W$.

Table II. An example post from the *book* dataset (cf. Section 4.1) that is diminished by the construction of cores. In the post – stored by a user with ID 1015 – the resource is a bookmark to the URL http://www.google.com/trends and the post was annotated with five tags (see also Fig. 1). Through the core constructions, some tags are removed from the data, while others remain (column "tags") leaving the post diminished in the respective core. Finally for tas-graph-cores at levels $l \geq 13$ and post-graph-cores at levels $l \geq 6$ the post vanishes completely from the core.

core	tags
full dataset	statistics, trends, comparateur, requetes, webmetrics
tas-graph-core at level $l = 2$	statistics, trends, comparateur
tas-graph-core at level $3 \leq l \leq 12$	statistics, trends
post-graph-core at level $2 \leq l \leq 5$	statistics, trends

entity of $\mathbb{F}$ does not occur in any post contained in $\tilde{S}$. Now, the functions

$$p_{\tilde{S}} \colon \tilde{S} \to \mathbb{N}^3 \colon (u, T_{ur}, r) \mapsto \left(|\tilde{S}_u|, \min_{t \in T_{ur}} |\tilde{S}_t|, |\tilde{S}_r| \right)$$

are monotone in the way required by Theorem 2.3 (where $\mathbb{N}^3$ is partially ordered as usual by $(a_1, b_1, c_1) \leq (a_2, b_2, c_2) \iff a_1 \leq a_2, b_1 \leq b_2, c_1 \leq c_2$). The monotonicity simply follows from the observation that by shrinking $\tilde{S}$ the sets $\tilde{S}_u, \tilde{S}_t$ and $\tilde{S}_r$ can loose but never gain cardinality.

The functions assign to each post a triple: the number of posts each of the post's entities is part of within the subset $\tilde{S}$. For any vector $l \in \mathbb{N}^3$ we can now construct a *post-set-core* as a set-P-core at l. In particular, this notion of core allows us to select three different thresholds $(l_u, l_t, l_r) \in \mathbb{N}^3$ for the number of occurrences of users, tags, and resources, respectively. The following examples illustrate use cases for choosing different thresholds:

— When one goal of a tag recommender is to consolidate the tag vocabulary of the system, a large threshold l_t ensures that only frequently used tags remain in the dataset for evaluation. The thresholds l_u and l_r can remain low.
— If the cold-start problem for users and resources shall be neglected in the evaluation, high values for l_u and/or l_r can be selected while l_t can be low.

For the sake of simplicity, we say that a post-set-core is of level l when all three thresholds are equal to l.

Running Example. The *post-set-core* at level $(2, 2, 2)$ is shown in Figure 2, where every user, tag, and resource of the four posts $1, 2, 3$, and 4 belongs to at least two of these four posts. The example also illustrates an important property of the post-set-core construction: None of the remaining posts is diminished, all remaining posts are complete in the sense that they still contain all the tags they have in the original dataset, as each post as a whole is treated as an atomic part of the dataset. This is neither the case for the tas-graph-core (e.g., post 7 loses tag 5) nor in the post-graph-core (post 6 loses tag 4), since here the posts are modeled as collections of tag assignments and tag assignments are removed individually during the core construction.

The example in Figure 2 illustrates the property that a tas-graph-core always contains the post-graph-core at the same level, and the latter contains all posts of the post-set-core at that level. This property follows directly from the core construction and is formalized in the following lemma.

LEMMA 2.5. *Given a folksonomy $\mathbb{F}$ and a level $l \in \mathbb{N}$.*

(*1*) *Each user u, tag t, and resource r, as well as each tag assignment (u, t, r) of the post-graph-core at level l is contained in the tas-graph-core at level l.*

(2) *For each post* (u, T_{ur}, r) *in the post-set-core at level* l*, the entities* u*,* r*, and* $t \in T_{ur}$*, as well as all tag assignments* (u, t, r) *(for* $t \in T_{ur}$*) are contained in the post-graph-core at level* l*.*

Finally, a trivial example for each core type is the 1-core, which is the full folksonomy itself, excluding isolated nodes (e.g., users that registered with the system but never used it and thus do not occur in a post). Tag recommender evaluation usually ignores isolated nodes and therefore the cores at level 1 are just the original evaluation datasets (in the following also referred to as the *raw data*).

Similar Constructions on Other Data. The core constructions described for folksonomies can easily be generalized to other similar data structures where entities have some countable properties. For example a tweet in the micro blogging system Twitter[3] consists of a user, URLs, hashtags, and several words. Much like in the case of folksonomies we can derive countable properties for each tweet t, e.g., the minimum number of tweets the URLs of t occur in, the minimum number of tweets that each hashtag of t occurs in, or the minimum number of tweets each word of t occurs in, etc. Using a set-core like in 2.3.3, we can then simply impose individual thresholds on the URL, hashtag or word frequencies. Depending on the particular use case, one might, for example, set high thresholds on the hashtag and URL frequencies to select only trending topics and resources, while setting a low threshold for words. Moreover, it would be possible to combine two aspects, say URLs and hashtags, by using maps that count the number of tweets that share either the URL or a hashtag with a tweet t.

In contrast to the use of set-cores, graph cores would require to impose a graph structure first, e.g. by connecting all entities of a tweet by 4-dimensional hyperedges where each edge connects the user to one of the hashtags, to one of the words, and to one of the URLs of a tweet. Other than with set-cores however, such graph-cores would yield "diminished tweets" (e.g., missing some infrequent words or hashtags). Furthermore, since including URLs or hashtags in a tweet is optional, the graph model would have to be able to deal with tweets that do not contain all these components, e.g., by using edges of different dimensionality.

3. RELATED WORK

In this section, we review and discuss several examples from the literature that deal with the topics of this work. We start with the previous use of cores in various areas of research before we turn our attention to the evaluation of recommender systems. We discuss the well-known problem of *sparse data*, which can be tackled by focusing on the dense part of the data, e.g., by using *graph cores*. Since the latter is often the case in the benchmarking of tag recommender algorithms we review the state of the art in that area next, covering different approaches as well as variations in the experimental setups. Finally, we compare several previous tag recommender benchmarking studies regarding their use of cores.

Graph Cores. One widely applicable methodology to create dense subsets of graphs are the so-called *graph-cores* which were introduced by Seidman [Seidman 1983]. Batagelj and Zaveršnik [Batagelj and M. Zaveršnik 2002; Batagelj and Zaveršnik 2011] extended this work by introducing *generalized cores* – see Section 2 for details. Cores have previously been used to create generative models of graphs [Baur et al. 2007] or to improve the visualization of large networks [Ahmed et al. 2007]. Angelova et al. [Angelova et al. 2008] analyzed cores of various derived graphs (friendship, common entities, and similarity graphs) of a social bookmarking dataset. The number of

[3]http://www.twitter.com/

connected components quickly drops to one, already for small core levels. In general, an increasing core level results in a decreasing average node distance and a more complex behavior of the average clustering coefficient. In [Wang and Chiu 2008], cores of a transaction network of an online auction system were used to identify densely connected subgraphs of malicious traders in order to recommend trustworthy auction sellers. By now, cores are an established methodology to analyze the structure and dynamics of graphs with applications as diverse as, e.g., community detection [Giatsidis et al. 2011], temporal analysis of the internet topology [Alvarez-Hamelin et al. 2005], or the study of large-scale software systems [Zhang et al. 2010]. Our generalization to arbitrary sets in Section 2 therefore opens up new possibilities for core-based analyses on data other than graphs.

Evaluation of Recommender Systems. Research on recommender systems evaluation typically focuses on the selection of proper metrics and performance criteria (like user preference, coverage, trust, novelty, etc.) as well as on data processing and selection methods. A good overview is presented in [Shani and Gunawardana 2011]. A fixed selection of metrics and criteria constitutes an *evaluation framework* like the one presented in [Pu et al. 2011] with a focus on criteria to measure the recommendation performance from the users' point of view. More often, however, such a framework is chosen ad-hoc and the implications of the selection have rarely been investigated. Often several choices are valid and plausible yet can lead to contradictory results. The consequence of different choices of an evaluation metric was, for example, demonstrated in [Schein et al. 2002], where two metrics were compared. One metric focuses on a broad coverage of users (good recommendations for each user) while the other rewards as many good recommendations as possible independent from the distribution over the users. In practice, the choice of the 'best fitting' metric is up to the operator of the actual recommender system. Many studies comparing recommender algorithms find that (part of) their results are dataset dependent. For example [Karypis 2001] showed that in item-based collaborative filtering algorithms, varying a particular parameter (controlling the influence of popular items) yields different behavior on different datasets.

[Cremonesi et al. 2010] used a movie recommendation scenario to demonstrate that different recommender algorithms respond differently to a subsampling of the test set. Their approach was to remove items from the test set that belong to the most popular (most frequent) items in the datasets. Thus a strictly popularity-based algorithm exhibited a heavy performance decrease (compared to its score on the full test data). In contrast, other algorithms had less strongly decreased scores, such that the resulting ranking of the top performing algorithms was different to that on the full test set. The motivation for the exclusion of the most popular items in [Cremonesi et al. 2010] was to demote algorithms that tend to favor the most popular items as such items are often already known to the user and thus do not present interesting recommendations. In contrast, the core construction in the tag recommender setting is not used to filter out particularly unwanted recommendations but rather to mitigate the cold start problem. We will however demonstrate in Section 5.1 that also in this scenario, different recommenders react differently to changes of the core setup and in Section 5.3 we discuss the changes in the resulting rankings of algorithms. [Adomavicius and Zhang 2012] investigate for the use case of the classical recommender scenario – where users assign ratings to items – how different dataset characteristics influence the resulting scores of recommendation algorithms. Several recommender algorithms are evaluated on various subsamples of five datasets using RMSE. The results show that properties like the size of the rating space and the data density have a positive impact on the recommendation quality, while others have a negative impact (e.g., the standard deviation of the ratings in a dataset). Furthermore, there are properties of the rating frequency

distribution that have a positive influence for some of the tested algorithms, but a negative influence on the performance of others. In our experiments in Section 5.1 we can similarly observe that the properties core type and core level have an influence on the benchmarking of tag recommender algorithms and that different tag recommender algorithms react differently to different cores.

Sparse Data. In many recommendation scenarios, the sparsity of the data is a classical problem since it can lead to overfitting and impede the performance of recommenders. In particular, sparse user rating data limits the identification of similar users and items in collaborative filtering [Sarwar et al. 2000]. The sparsity problem has been tackled either by dealing with the sparsity in particular or by focusing on the dense part of the data, e.g., [Sarwar et al. 2001]. A typical approach to reduce sparsity is dimensionality reduction, e.g., via matrix factorization. For instance, [Ma et al. 2008] proposed a matrix factorization approach that combines traditional rating data with social network data to reduce the sparsity of the ratings matrix. [Sarwar et al. 2000] used singular value decomposition to compute user neighborhoods on dense, low-dimensional product matrices. Content-based approaches have also be used to increase the density of the rating matrix for collaborative filtering, e.g., [Melville et al. 2002], or have been combined with collaborative methods, e.g., [Popescul et al. 2001] using a unified probabilistic framework. Most of the approaches that focus on the dense part of the data are rather ad-hoc, e.g., by defining some threshold for the minimal number of ratings an item or user should have. There are few theoretical considerations or experiments that investigate the implications of such thresholds on the performance of different recommender algorithms or the validity of the experiments. For example, [Herlocker et al. 2004] addressed the density of datasets as one of the properties that influence recommender systems evaluation. While they empirically compared different (classes of) evaluation metrics, they do not further investigate density as a factor of the evaluation. In this work, we show how using cores (to increase the density of the data) can influence the results of a tag recommender benchmarking.

Tag Recommender Systems and Their Evaluation. Since the emergence of social bookmarking, the topic of tag recommendations has raised considerable interest of researchers such that a vast amount of related work exists. Recommending tags can serve various purposes, such as: increasing the chance of getting a resource annotated, reminding a user what a resource is about, and consolidating the vocabulary across the users. Furthermore, as [Sood et al. 2007] pointed out, tag recommenders lower the effort of annotation by changing the process from a *generation* to a *recognition* task, i.e., rather than "inventing" tags the user only needs to find and select a recommended tag. Here, we introduce a selection of important results and observe different experimental setups for the comparison of different algorithms.

Early approaches include [Mishne 2006] and [Byde et al. 2007] who devised methods using content-based filtering techniques. [Mishne 2006] combined a manual evaluation of the recommended tags for 30 randomly selected blog posts (using precision@10) with an automatic comparison against 6000 randomly selected already tagged posts (using precision@10 and recall@10). Only posts that have three or more tags were considered for the automatic evaluation and for comparing tags string distance was used instead of exact matching, though no details about the maximal allowed string distance were given. The setup in [Byde et al. 2007] is only briefly explained, it appears as if the coverage is computed for 6180 posts from 36 users, though no details about the hold-out set are given. [Xu et al. 2006] identified properties of good tag recommendations like high coverage of multiple facets, high popularity, or least-effort and introduced an approach based on the HITS algorithm [Kleinberg 1999]. A goodness measure for tags, derived from collective user authorities, is iteratively adjusted by a

reward-penalty algorithm. The approach was evaluated qualitatively on 18 resources (URLs). Other early works include [Basile et al. 2007], who suggested an architecture of an intelligent tag recommender system based on words extracted from the resources and a Naive Bayes classifier (no evaluation is performed), and [Vojnovic et al. 2007], who tried to imitate the learning of the true popularity ranking of tags for a given resource during the assignment of tags by users. The method in [Vojnovic et al. 2007] was evaluated with precision over 1 200 resources, though no details about the hold-out set are given. The mentioned approaches address important aspects of the tag recommendation problem, but they diverge on the notion of tag relevance and evaluation protocol used. In [Xu et al. 2006; Basile et al. 2007], e.g., no quantitative evaluation is presented, while in [Mishne 2006], the notion of tag relevance is not entirely defined by the users but partially by experts. An extensive evaluation of collaborative filtering, the graph-based FolkRank algorithm [Hotho et al. 2006], and simpler methods based on the usage frequency of tags was performed in [Jäschke et al. 2008] on three datasets from CiteULike, Delicious, and BibSonomy. FolkRank outperformed the other methods and a hybrid that combined frequently used tags of the user with tags that were frequently used to annotate the resource was second. The evaluation was performed using post-graph-cores in a setup called *LeavePostOut* – a variant of the leave-one-out hold-out estimation [Herlocker et al. 2004] – where for each user a post is randomly selected and removed from the dataset while all remaining data is used for training.

The ECML PKDD Discovery Challenges 2008 and 2009 [Hotho et al. 2008; Eisterlehner et al. 2009] addressed the tag recommendation task and resulted in many new approaches. In particular, the 2009 Discovery Challenge established a common evaluation protocol against which more than 20 approaches were evaluated: on datasets from BibSonomy, posts from the most recent six months were used as test data and the approaches were evaluated with the F1 measure over the top five recommended tags. One task focused on graph-based recommendations and ensured that every tag, user, and resource from the test dataset were already contained in training data by using a post-graph-core at level 2. The content-based task was evaluated on the complete six months of the test data. The 2nd of 2008 and winner of the 2009 content-based task [Lipczak et al. 2009] was a hybrid recommender that combined tag suggestions from six sources (e.g., words from the title expanded by a folksonomy driven lexicon, tags from the user's profile) and re-scored them, e.g., by the frequency of usage of the posting user. The winners of the graph-based task [Rendle and Schmidt-Thieme 2009] produced recommendations with a statistical method based on factor models. They factorized the folksonomy structure to find latent interactions between users, resources and tags. Using a variant of the stochastic gradient descent algorithm, the authors optimized an adaptation of the Bayesian Personal Ranking criterion [Rendle et al. 2009]. The learned factor models are ensembled using the rank estimates to remove variance from the ranking estimates. Finally, the authors estimated how many tags to recommend to further improve precision using a linear combination of three estimates. A novelty of the challenge was the evaluation of some algorithms in an online setup where the click-rate of BibSonomy users could be measured. There, again the recommender presented in [Lipczak et al. 2009] clearly outperformed other approaches. For further results of the challenges we refer to the proceedings [Hotho et al. 2008; Eisterlehner et al. 2009] and the summary in [Jäschke et al. 2012].

[Krestel et al. 2009] presented a tag recommendation algorithm based on Latent Dirichlet Allocation. The evaluation was performed per resource, i.e., almost all posts (except up to five) for a resource were removed and the recommender then tried to predict their tags. The test data consists of 10% of the posts and the whole experiment was repeated five times. [Ramezani 2011] introduced a new weighted and directed folksonomy graph model on which she applied the PageRank algorithm. More recently,

[Seitlinger et al. 2013] proposed an approach that simulates human category learning in a three-layer connectionist network. Similar to [Krestel et al. 2009], in the input layer Latent Dirichlet Allocation is used to characterize the resource (and user). [Ma et al. 2013] proposed 'TagRank', a variant of topic-sensitive PageRank upon a tag-tag correlation graph which they integrate into a hybrid with collaborative filtering and popularity-based algorithms. The selection of the algorithms for the hybrid is guided by a greedy algorithm. While the approaches in [Ramezani 2011] and [Seitlinger et al. 2013] were evaluated with the same setup as in [Jäschke et al. 2008] (using LeavePostOut on cores and selecting one post per user at random), [Ma et al. 2013] performed five-fold cross validation and the results were "averaged over each user, then over the final five folds" though no details on how the data was split (e.g., per user) were given. Overall, algorithms are typically evaluated on offline datasets, the posts for the test sets are selected at random, and measures like precision, recall, and F1 are used for evaluation. There is a tendency to use the LeavePostOut methodology, though other cross-validation procedures (LeavePostOut is $|U|$-fold cross validation where $|U|$ is the number of users in the dataset) and other types of splits are also used.

Cores and Recommender Systems. As part of the evaluation of recommender systems, cores have first been used in [Jäschke et al. 2007] to focus on the dense part of folksonomies – the tripartite hypergraphs of collaborative tagging systems. Experiments with different tag recommenders were conducted on subsets of folksonomies, constructed as generalized cores – so-called *post-cores* like explained in the previous section. Cores were then commonly used in the evaluation of (tag) recommendation algorithms for collaborative tagging systems, e.g., in [Ramezani 2011] to compare different PageRank variants on cores from Delicious, BibSonomy, and CiteULike at levels 20, 5, and 5, respectively, in [Krestel et al. 2009] to evaluate a tag recommendation algorithm based on Latent Dirichlet Allocation on a core at level 100 of a dataset from Delicious, in [Seitlinger et al. 2013] to evaluate a category-based tag recommender on a Delicious core at level 14, in [Ma et al. 2013] to evaluate 'TagRank', a variant of topic-sensitive PageRank upon a tag-tag correlation graph on a Delicious core at level 9 and cores at level 5 from Last.fm and Movielens, and in [Nanopoulos et al. 2013] to evaluate a matrix factorization-based song recommender with the core level set such that it is equivalent to 0.001% of the total playcounts. As mentioned earlier, a post-graph-core at level 2 of a BibSonomy dataset was also used in the ECML PKDD Discovery Challenge 2009 for the graph-based task.

As this overview shows, the choice of the particular core level is very diverse and typically neither justified nor evaluated. The arguments for using cores are similar throughout these approaches and are summarized in [Ma et al. 2013]: "the size of each dataset is dramatically reduced allowing the application of recommendation techniques that would otherwise be computationally impractical, and by removing rarely occurring users, resources and tags, noise in the data can be considerably reduced." Except for [Jäschke et al. 2008], all these works did neither question or challenge the use of cores nor did they compare their findings on several cores or to results on the raw data. In [Jäschke et al. 2008] the results on a Delicious core at level 10 were compared to results on a dataset where only users and resources with less than two posts were removed. Recall and precision of all algorithms except the adapted PageRank were found to be similar. Furthermore, besides the typical lack of evaluation on the raw data, all aforementioned evaluation setups suffer from the problem of diminished posts which we described in Section 2.3.2 together with a solution by introducing post-set-cores.

Summary. As we pointed out in the previous paragraphs, one commonly used framework for collaborative tagging systems comprises graph cores in an offline setting

where coverage and accuracy (which correspond to recall and precision, resp.) are measured. In this paper we do not aim at the evaluation of different properties of recommender systems nor at the presentation of a new evaluation framework. Instead, we investigate the robustness of that common evaluation framework itself and thereby challenge commonly used methodologies. Therefore (and to be comparable with previous works), we investigate the influence of different core types and levels within the fixed framework for *offline* evaluation of *tag recommender systems* in *folksonomies* using *graph cores* in combination with the *LeavePostOut* method and the standard measures *precision, recall, and MAP*.

A first rigorous evaluation of the influence of different types and levels of cores was performed in [Doerfel and Jäschke 2013]. In this paper we extend this work by performing a thorough assessment of the evaluation framework based on the use of cores, which has been employed in the aforementioned works. In particular, we extend [Doerfel and Jäschke 2013] by 1) introducing our generalization of cores to solve the problem of diminished posts in Section 2, 2) extending the experiments to the CiteULike dataset, to new core levels, and to two new benchmarking setups (comparing recommenders on post-set-cores and on diminished posts), and 3) investigating further properties of the datasets, algorithms, and cores (Sections 4.1 (properties), 5.2, 5.4, and 5.5).

4. EXPERIMENTAL EVALUATION

The main goal of the experiments in this work is to demonstrate how benchmarking results act over different core types and levels. In the experiments we show that the quality of recommendations depends on (mostly increases with) the core level, that diminished posts indeed occur frequently in tas-graph-cores and post-graph-cores and these posts influence the overall results, and that different core setups (different core types or levels) can lead to conflicting results in a benchmarking's ranking of algorithms. Furthermore, we point to a peculiarity of using cores that arises from their use in the LeavePostOut evaluation scenario. To that end, we choose a fix evaluation setup for tag recommender algorithms – like it has been used frequently in previous studies – and apply it to four real world datasets. In that setup we then vary the cores and discuss the differences in the results using different metrics.

In this section we describe the setup of our experiments to test different evaluation procedures with different cores, levels, and metrics for tag recommender algorithms. More specifically, we

— describe four datasets from three collaborative tagging systems, namely BibSonomy, CiteULike, and Delicious,
— explain the cleansing procedure that includes, amongst others, the removal of imported posts,
— show some basic properties of the datasets like size and density for different core types and levels, and
— detail which cores, evaluation protocol (*LeavePostOut*), metrics, and recommender algorithms we use.

4.1. Datasets

We use four datasets from three tagging systems (for an overview, cf. Table III): The *BibSonomy*[4] dataset from 2012-07-01 is a regular dump of the system's publicly available data.[5] The generation of the dataset is described in [Jäschke et al. 2012]. BibSonomy supports bookmarking of both bookmarks and publication metadata, hence we

[4]http://www.bibsonomy.org/
[5]http://www.kde.cs.uni-kassel.de/bibsonomy/dumps/

Table III. The sizes of the four datasets, the levels l_m of their main cores for the three different core types: tas-graph-core ($\mathcal{T}$), post-graph-core ($\mathcal{P}$), and post-set-core ($\mathcal{S}$), and the levels chosen for the experiments. (* Some levels of the post-set-core were ignored in the evaluation, cf. Section 4.2.)

dataset	$\|U\|$	$\|T\|$	$\|R\|$	$\|Y\|$	$\|\text{posts}\|$	$l_m(\mathcal{T})$	$l_m(\mathcal{P})$	$l_m(\mathcal{S})$	chosen l
publ	4 777	57 639	94 427	397 081	109 984	23	12	4	2–6
book	4 959	80 603	231 907	1 032 037	268 589	60	9	6	2–6*
deli	75 071	397 028	2 999 487	17 280 065	7 268 305	200	75	58	2–10, 20
cite	75 657	421 874	1 604 856	7 712 798	2 400 489	153	19	19	2–10, 15*

split the data into two parts: *book* and *publ*. From *CiteULike*[6] we use the official snapshot (*cite*) from 2012-05-14.[7] From *Delicious*[8] we use a dataset (*deli*) that was obtained from July 27 to 30, 2005 [Hotho et al. 2006] which is a subset of the Tagora dataset.[9]

Cleansing. As [Lipczak et al. 2009] pointed out, tags from automatically imported posts are problematic for training and evaluating tag recommenders, since their provenance is unknown. They might have been automatically extracted from the title of a resource or resemble the folder structure of a browser's bookmark directory and thus do not necessarily reflect the user's view on the resource. The (in)ability of a recommender to predict such tags does not allow us to draw any conclusion about its performance on predicting user-generated tags. Moreover, in most systems, recommendations are usually not provided during import. Hence, we applied a similar cleansing strategy as described in [Lipczak et al. 2009]: we removed sets of posts that were posted at exactly the same time by the same user. Furthermore for the *cite* dataset, additional cleansing was necessary. A thorough inspection of the data had revealed that the tags *no-tag*, *todo_mendeley* and (many different) tags like *bibtex-import*, **file-import-10-07-11*, or *imported-jabref-library* were frequently (and exclusively) used to indicate imported posts. However, the posts of such an import had not identical but slightly different (consecutive) timestamps and were thus not removed by the previously applied strategy. Therefore, we additionally removed all posts from *cite* that were exclusively tagged with the tags *no-tag* or *todo_mendeley*, or a tag matching the regular expression $\backslash bimport\backslash b$ or $\backslash bimported\backslash b$ (where $\backslash b$ indicates a word boundary). In addition, we cleaned all tags as described in [Jäschke et al. 2012], i.e., we ignored tag assignments with the tags *imported*, *public*, *system:imported*, *nn*, *system:unfiled*, converted all tags to lower case, and removed characters which were neither numbers nor letters.

Properties. The core construction process rapidly reduces the number of tags, users, and resources, e.g., from 2 999 487 resources (397 028 tags) in the raw *deli* dataset (cf. Table III) to 588 816 resources (65 050 tags) in the tas-graph-core at level 5. The decline of the number of users for an increasing core level can exemplarily be seen in Figure 3(c). The smaller datasets *book* and *publ* quickly vanish with rising core level and although the number of users for *cite* and *deli* is very similar, the number drops much quicker in *cite* than in *deli*. Due to the decrease of the number of nodes, experiments using cores with higher levels require a much lower computational effort since the complexity of most recommender algorithms depends on the number of entities (users, tags, and resources) or tag assignments.

Since the usual argument for the use of cores is their higher density compared to raw datasets [Krestel et al. 2009; Ramezani 2011; Nanopoulos et al. 2013; Seitlinger et al. 2013], we compare this property for all three core types on the four datasets. The

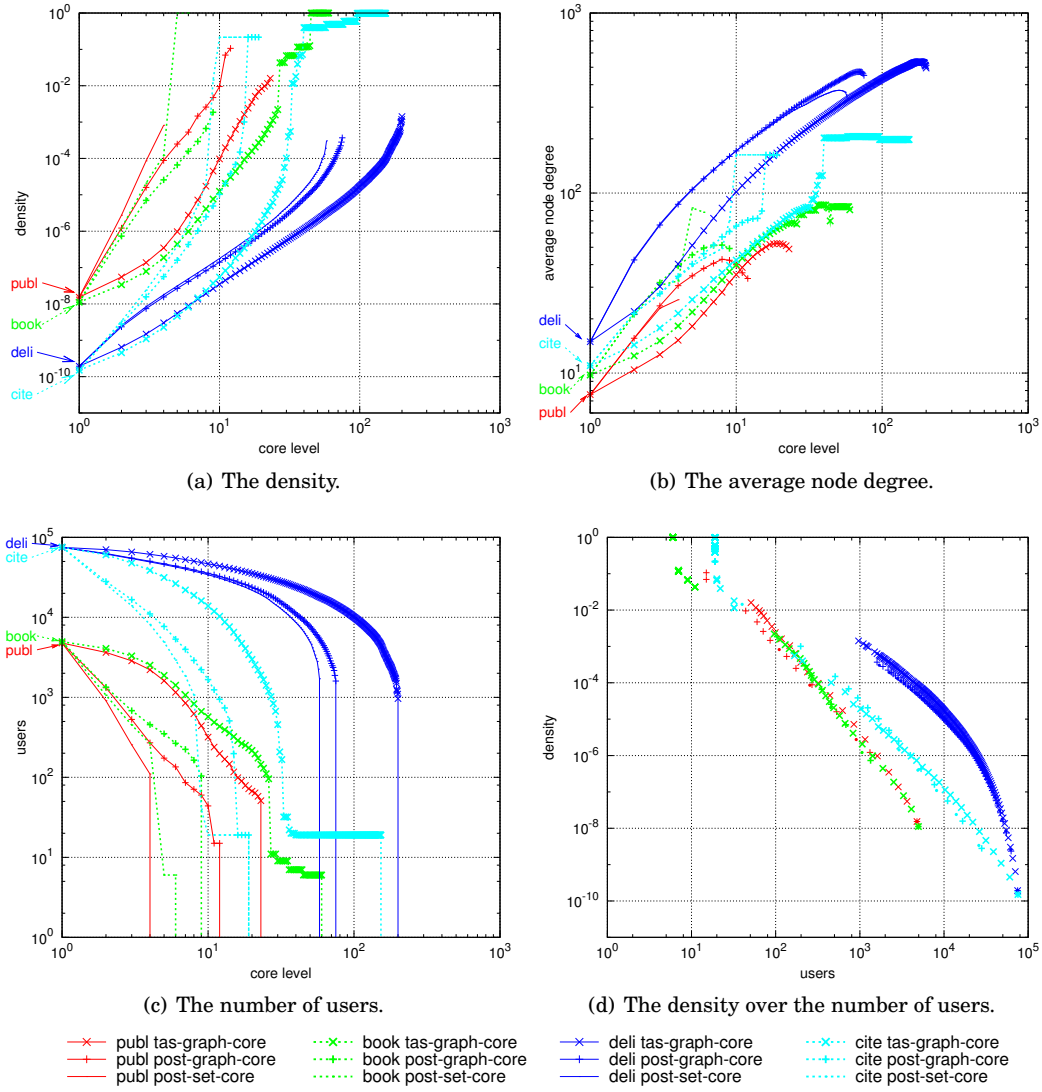
[6] http://www.citeulike.org/
[7] http://www.citeulike.org/faq/data.adp
[8] http://www.delicious.com/
[9] http://www.tagora-project.eu/data/#delicious

density of a graph is often defined as the fraction of realized edges among the theoretically possible edges, e.g., in [Diestel 2005] referred to as *edge density*. In a folksonomy, the edge density is equal to $|Y|/(|U| \cdot |T| \cdot |R|)$. Other sources (e.g., [Janson et al. 2000]) define the density rather as the ratio between edges and nodes. In that case the density is proportional to the average node degree in the graph. In a folksonomy the average node degree is three times the edge-node ratio: $3 \cdot |Y|/(|U| + |T| + |R|)$, since every (hyper-)edge in $|Y|$ connects three nodes. The edge density is also related to the node degree: it can be understood as the ratio between the actual sum of degrees (in a folksonomy that is $3 \cdot |Y|$) and the theoretically possible sum of degrees (in a folksonomy that is $3 \cdot |U| \cdot |T| \cdot |R|$).

Figures 3(a) and 3(b) show for each dataset and each core type the edge density and the average node degree, respectively, depending on the chosen core level. As expected, the edge density increases with the core level and for the same level and the same dataset the post-set-core is the densest core, followed by the post-graph-core and the tas-graph-core. The average node degree at first also rises with the core level, however, with the last levels before the core vanishes we can observe a decrease for most of the cores. In the case of the tas-graph-core on the book dataset, the average node degree drops quickly at core level 44 and then rises again at the next level. This behavior coincides with a sharp drop of the number of users in Figure 3(c) and a sharp rise in density in Figure 3(a). An inspection of the graph properties showed that from level 43 to 44 the *book* dataset lost almost half of its remaining edges (tag assignments) but only few nodes, while from 44 to 45 it lost two thirds of the edges but also about 75% of the nodes, resulting in a much smaller and denser graph.

Comparing the behavior with respect to both density and node degree between the three core types, we see that post-graph-cores and post-set-cores are more similar to each other (especially the average node degrees are close together) than to the tas-graph-cores which always have a lower density and a lower average node degree than the other two types (compared for the same dataset and level). It is also worth noting, that the smaller datasets (*publ* and *book*) have fewer users and lower average degrees than the larger datasets (*deli* and *cite*), yet higher densities. This was to be expected: it is a consequence of the number of possible edges (that enters the formula for the density) which rises super-linear with the number of nodes. Among the four datasets and three core types, we can see three cores that reach the maximal density of 1: the post-set-core at level 6 of *book* (at level 5 the density is $826/840 = 0.98\bar{3}$), the tas-graph-core at level 45 of *book*, and the tas-graph-core at level 96 of *cite*. There is no general pattern that indicates which main core has the highest density, e.g., for the *publ* dataset the densest main core is a post-graph-core but for the *deli* dataset it is a tas-graph-core.

Another observation is that, although the density of the raw *cite* dataset ($0.15 \cdot 10^{-9}$) is slightly smaller than that of *deli* ($0.19 \cdot 10^{-9}$), it is growing much quicker with the core level than on *deli* and is already higher than on *deli* at a level of 2 for both the post-set-core and the post-graph-core. Such a rapidly increasing density can be explained by a larger share of sparsely connected nodes compared to well-connected nodes: nodes that are not well-connected are removed in cores of higher levels, while well-connected nodes are more likely to remain in the (thus denser) core. In *deli*, on the contrary, we can infer that the share of well-connected nodes is higher than on *cite*, since the density increases less rapidly. Finally, Figure 3(d) confirms that the density increases with a decreasing number of users (and thus with increasing level). We can observe that per dataset the three curves (one for each core type) are almost indiscernible, which indicates, that the core type has no pronounced influence on the relation between density and the size of the dataset. Comparing the curves of different datasets, we also note that their slope is a dataset dependent property.



(a) The density.

(b) The average node degree.

(c) The number of users.

(d) The density over the number of users.

Fig. 3. The density 3(a), the average node degree 3(b), and the number of users 3(c) in the graphs of the different cores for each dataset as a function of the core level, and the density as a function of the number of users 3(d).

4.2. Evaluation Methodology

The dimensions of our experiments are the four datasets, the three different core types, the chosen levels (see Table III), the recommendation algorithms, and the evaluation metrics.

Cores and LeavePostOut. For the experiments we used – besides the raw datasets (or 'cores at level 1') – all three types of cores we described in Section 2.3. Although the post-set-core allows us to select different thresholds for users, tags, and resources, we used only one single threshold l for three reasons: 1) to be comparable to the tas-graph-cores and post-graph-cores which do not allow separate thresholds, 2) to be consistent

with most of the previous tag recommender evaluation works without particular focus on special use cases like the consolidation of the tag vocabulary (cf. Section 2.3), and, finally 3) to keep the dimensionality of the experiments manageable.

For each dataset we chose several core levels on which we conducted the experiments (see 'chosen l' in Table III). The difference in choice is due to the different characteristics of the datasets (size, level of the main core, unchanged cores over several levels, etc.). I.e., for the two smaller datasets (*book* and *publ*), we selected five levels (2–6). The two larger datasets (*deli* and *cite*) allow the selection of higher levels and thus we chose consecutive levels up to level 10 and then for each dataset one larger level (20 for *deli* and 15 for *cite*), taking into account that the cores of *cite* vanish much faster than those of *deli*. Due to the rapid rise in density with rising core level, some cores have only very few nodes (cf. Figure 3(c)). In particular, the post-set-cores at levels 9 or higher of the *cite* and at levels 5 and 6 of the *book* dataset contain less than 40 users. Such cores do not allow a representative evaluation of recommender algorithms, since it would rely on the judgment of very few users. They have therefore been excluded from the analyses. All other considered cores have more than 100 users.

To evaluate the recommenders, we used the variant of the leave-one-out hold-out estimation [Herlocker et al. 2004] called *LeavePostOut* [Jäschke et al. 2007]. It is a very common choice in tag recommender evaluation (e.g., [Ramezani 2011; Seitlinger et al. 2013; Montañés et al. 2011; Kubatz et al. 2011]). Given a dataset (or a core), one experiment consists of the following steps: For each user u: 1) One post p is selected at random. 2) The post p is eliminated from the dataset and the remaining data is used for training. 3) The task for the recommender algorithm then is to produce tag recommendations (i.e., to predict the tags of p) given both the user and the resource of p, while the tags of p serve as gold-standard. 4) A score is assigned that measures the prediction quality of the recommendation. This procedure is repeated for every user and the resulting scores are averaged. To ensure statistical validity, we repeated each experiment five times – such that every time a post is randomly drawn for each user – and report the averages of the resulting scores.

Evaluation Metrics. The evaluation metric determines the quality of a recommender by measuring how successful an algorithm can predict the tags of the left-out post. We use the two common metrics *recall* and *precision* at a given cut-off level k ($\mathrm{rec@}k$ and $\mathrm{pre@}k$). For a left-out post p, $\mathrm{rec@}k$ is the share of p's tags among the top k positions of a recommender's ranking and $\mathrm{pre@}k$ the share of the top k tags in the ranking that belong to p. In the experiments we let k run from 1 through 10. The *mean average precision* (MAP) computes the arithmetic mean of the precision taken at each position of a ranking where the recall changes. [Manning et al. 2008].

Recommender Algorithms. Since the goal of our experiments is not to find the best algorithm, but rather to analyze the experimental setup itself, we select a set of well-studied tag recommendation algorithms, namely *most popular tags*, *most popular tags by resource*, *most popular tags by user*, *adapted PageRank*, and *FolkRank*. Adapted PageRank and FolkRank were first presented in [Hotho et al. 2006]. Both are adaptations of the original PageRank algorithm [Brin and Page 1998] to the ternary hypergraph of folksonomies. The FolkRank algorithm computes the difference between one run of the adapted PageRank without preference (i.e., the global ranking) and one run with preference for the user/resource pair for which tags should be recommended. The latter run also constitutes the result for the adapted PageRank. We do not explore particular properties of the algorithms and therefore omit further descriptions. They can be found in [Jäschke et al. 2008]. In particular, we use the same parameter setting of $d = 0.7$ for both the adapted PageRank and FolkRank as in [Jäschke et al. 2008]. Note that all 'most popular' recommenders count the number of *posts* that contain the

tag – in contrast to counting the number of *users* that have used the tag. The latter does not make sense for *most popular tags by user*, of course, and it does not make a difference for *most popular tags by resource*, since a user is typically allowed to bookmark a resource only once. We discuss the impact of the two counting strategies on the *most popular tags* recommender in Section 5.5. All chosen algorithms are graph-based (in contrast to content-based methods) and thus their performance may depend on the way the folksonomy graph is restricted through the core construction. Furthermore, we employ the *least popular tags* recommender to demonstrate an anomaly that affects the LeavePostOut methodology on cores (cf. Section 5.4). The algorithm is deliberately designed to produce bogus recommendations by always recommending those tags that occur the least often in the training dataset.

5. RESULTS

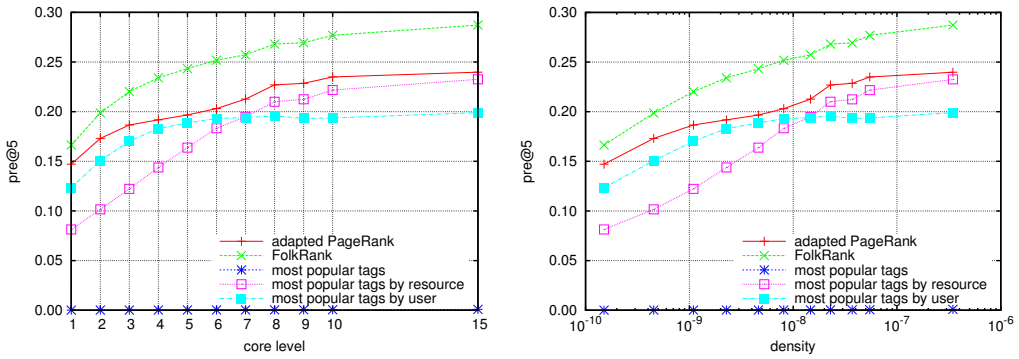
In total we conducted 937 experiments using different recommendation algorithms in different setups – each time conducting *LeavePostOut* once for each user. Each single experiment was repeated 5 times and evaluated using 21 different metrics. In this section we present and discuss our findings:

— We start by summarizing some general results on the performance of recommenders on different cores.
— In Section 5.1, we find that the performance of a recommender varies not only over different datasets, core types, and core levels but also changes, when using the same training data and choosing only the test posts from within denser cores.
— Section 5.2 addresses the problem of diminished posts, showing that such posts occur frequently in cores and influence the overall performance of recommenders.
— Section 5.3 is dedicated to the correlation between rankings of algorithms on different setups. We find that despite high consistency among those rankings, different setups may well lead to different conclusions about the performance of algorithms.
— We point to a statistical flaw of the use of cores within a *LeavePostOut* setup in Section 5.4.
— Finally in Section 5.5, we demonstrate how the *most popular tags* baseline can be affected by irregular tag distributions.

When we compare recommenders' scores on different cores and levels with different metrics we first observe that they tend to yield better scores on the post-set-cores than on the post-graph-cores of the same level (in 97.1% of the experiments) and better scores on the post-graph-cores than on the tas-graph-cores (88.0%). The performance of the algorithms on the tas-graph-cores, post-graph-cores, and post-set-cores is better than that on the raw datasets in 94.2%, 99.6%, and 99.7% of the cases, respectively. This increase of the scores raises the question whether the choice of the core has an influence on the comparison of different algorithms against each other.

As an example for how the ranking of algorithms can change with different core levels, Figure 4 shows a comparison of the $\mathrm{pre@5}$ of the five algorithms on the *cite* tas-graph-core. The observation that *FolkRank* shows the best performance is in line with prior results [Jäschke et al. 2008]. Although the ranking of the algorithms' performance is quite stable over the levels, the results of *most popular tags by user* are better than those of *most popular tags by resource* for tas-graph-cores for core levels 1 through 6 and smaller for higher levels. Thus, a single experiment using the raw data would have yielded another conclusion on the performance of these two recommenders than an experiment using only cores, e.g., at level 10. We further investigate correlations between such algorithm rankings on different cores in Section 5.3.

The plot also shows the unexpectedly bad performance of the *most popular tags* recommender on *cite* – a phenomenon we investigate in Section 5.5.



(a) Dependency on the core level. (b) Dependency on the density.

Fig. 4. The pre@5 scores of all algorithms on the *cite* tas-graph-core over different core levels and densities.

5.1. Recommendation Performance Depends on Core Type and Level

In our experiments, the most prominent observation is that the performance at different core levels depends both on the dataset and on the algorithm – as expected from previous work in recommender systems literature: e.g., [Cremonesi et al. 2010] found that different algorithms for item (movie) recommendation react differently to manipulation of the test set while [Jäschke et al. 2008] showed for different tag recommenders that their scores vary over datasets of different tagging systems. In Figure 5 (the lines labeled (a) tas-graph-core), we see exemplarily the pre@5 scores[10] for the five algorithms over different levels of the tas-graph-core for all four datasets. A strong visible tendency is that scores rise with an increasing level – exceptions are a few levels on *cite* for *most popular tags by user* or *deli* for *most popular tags*.

Further, we leverage the property that the tas-graph-core always contains the post-graph-core, which in turn contains the post-set-core at the same level using Lemma 2.5: Next to the scores on the tas-graph-cores (a) we plotted the scores of the same experiments with only a slight modification of LeavePostOut's post selection process. Where we usually choose one post per user at random, we now choose one post per user randomly such that it is also contained in the post-graph-core (b) or also contained in the post-set-core (c) – scores on diminished posts (d) will be relevant in the next section. Note, that only the selection of the left-out posts is different to (a), as all four variations use the same core (the tas-graph-core) for training. Comparing the scores on arbitrarily chosen posts to those particularly chosen from one of the smaller cores, we see that for most of the algorithms it is easier to predict tags for posts from the post-graph-core than for arbitrarily chosen posts. We yield even better results for posts also contained in the post-set-core. The exceptions to that tendency are the same we have observed before. We can conclude that focusing on posts from the dense part of the data often overestimates the performance of recommendation algorithms.

5.2. Diminished Posts

As already mentioned in Section 2.3, diminished posts, i.e., posts having fewer tags in cores than in the raw dataset, are a result of the design of the tas-graph-core and the post-graph-core. In contrast, post-set-cores do not suffer from this issue. To illustrate the influence of such diminished posts, we once more modified LeavePostOut's post

[10]To suggest five tags is a typical choice in tagging systems. The resulting diagrams for rec@5 are similar to those for pre@5.

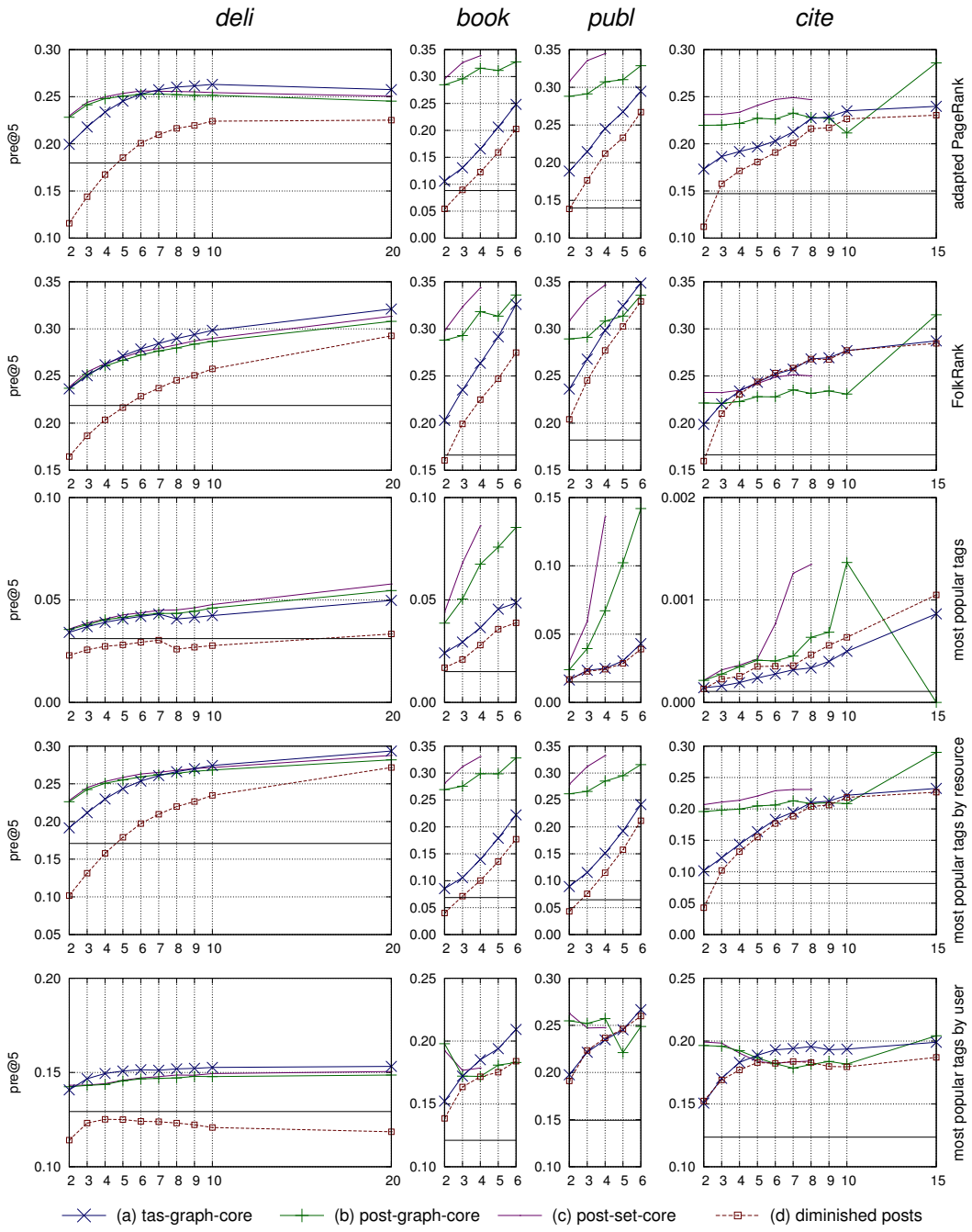


Fig. 5. The $\mathrm{pre@5}$ (on the y-axis) scores over the core level l (on the x-axis) for *deli*, *book*, *publ*, and *cite* for the five recommenders using modifications of LeavePostOut. Each column of plots represents the dataset specified at the top, each row contains results for the algorithm specified at the right, respectively. The horizontal lines depict the $\mathrm{pre@5}$ value for the respective raw dataset.

Table IV. Statistics on diminished post in each dataset for several core levels of tas-graph-cores and post-graph-cores. Listed are a core's share of posts that have lost tags (compared with the original dataset) as well as the average number of tags, that such posts loose. The table shows these statistics for the levels l=2, 3 and $l^{\max}$ where the latter denotes the highest level that was considered in our experiments i.e., $l^{\max} = 6$ for *publ* and *book* and $l^{\max} = 20$ and 15 for *deli* and *cite* respectively (cf. Table III).

dataset	share of diminished posts in %						average number of lost tags					
	tas-graph-core			post-graph-core			tas-graph-core			post-graph-core		
	2	3	$l^{\max}$	2	3	$l^{\max}$	2	3	$l^{\max}$	2	3	$l^{\max}$
publ	18	26	42	17	27	49	1.66	1.82	2.26	1.51	1.60	1.87
book	11	17	31	12	22	40	1.38	1.52	1.85	1.35	1.53	1.77
deli	2	3	7	2	2	7	1.20	1.23	1.29	1.15	1.16	1.22
cite	5	9	30	4	9	41	1.45	1.55	2.30	1.29	1.39	1.75

selection process (like in the previous section) to randomly choose only such posts (line (d) in Figure 5). We can observe that in most cases (with the exception of *most popular tags* on *cite*), the recommenders perform comparably well or worse on posts that have lost tags than on arbitrary posts. Regarding the exception, we have to consider that in general the scores of *most popular tags* are extremely low and thus only very few correctly predicted tags more or less can yield relatively large changes in the scores. The largest difference between the $\mathrm{pre@5}$ scores on arbitrary posts and on diminished posts can be observed on *deli*, e.g., 0.192 vs. only 0.102 for the diminished posts. In general, the amounts by which the scores differ are diverse without a clear tendency.

Table IV shows that diminished posts are not only a theoretical problem, but do indeed occur frequently in cores. We can see that on the two smaller datasets (*publ* and *book*) even for level 2 more than 10% of the posts in the core have lost tags, while there are fewer such posts in the larger datasets. Raising the core level, however, raises the share of diminished posts – most dramatically in the *cite* dataset, that has 5% diminished posts in the tas-graph-core (4% in the post-graph-core) at level 2 but a share of 30% (41%) at level 15 (the highest level used in our experiments). The *deli* dataset has significantly lower shares of diminished posts, yet also shows the tendency of a rising share for a rising core level. The second half of Table IV shows the average number of tags that diminished posts have lost. Again, we can observe that the numbers rise with a rising level. Each such post loses one or two (and even more in the higher levels of the tas-graph-core) tags on average. Lost tags pose an artificially introduced difficulty to the evaluation of tag recommendation algorithms, as there are less correct tags that could be predicted. Especially with a metric like $\mathrm{pre@5}$, one or two more tags to predict can make an enormous difference.

These observations support the assumption that diminishing posts has indeed an influence on the evaluation and is thus undesirable, as it is not clear how different algorithms react to such artificially modified posts. A reason for the weaker performance might be that it is easier to yield a higher precision when there are more tags to predict and thus it is more likely that one of these tags is recommended. However, we could observe the same behavior for the $\mathrm{rec@5}$ scores (without exceptions).

5.3. Recommender Ranking Correlation

Evaluating recommender systems usually has the goal to determine one algorithm that performs best on one or several datasets and therefore several algorithms are ranked according to their performance. Since several setups for experiments are possible – several core types, levels and metrics – the question arises, whether the ranking of recommenders varies depending on the chosen setup. To investigate this question we determine the algorithm rankings, where the algorithms are ranked according to their recommendation quality. A ranking can be computed on the raw datasets, on all three

Table V. The mean pairwise Pearson's r and the number of discordant pairs d in the recommender algorithm rankings on different cores together with the standard deviation σ.

dataset	metric	avg. r	σ	avg. d	σ
publ	MAP	0.912	0.074	1.473	1.148
publ	pre@5	0.909	0.079	1.593	0.977
publ	rec@5	0.920	0.076	1.516	1.026
book	MAP	0.908	0.092	1.429	1.087
book	pre@5	0.878	0.117	1.462	1.148
book	rec@5	0.912	0.090	1.330	1.076
deli	MAP	0.994	0.008	0.512	0.500
deli	pre@5	0.992	0.010	0.361	0.481
deli	rec@5	0.993	0.009	0.503	0.501
cite	MAP	0.981	0.026	0.651	0.735
cite	pre@5	0.972	0.043	0.492	0.676
cite	rec@5	0.976	0.043	0.595	0.766

core types and at all chosen levels.[11] Between two rankings (on two different setups) we can determine Pearson's correlation coefficient r, as a measure of how likely the score rankings of the recommenders are (linearly) correlated. The coefficient ranges from -1 (anti-correlation) through 0 (no correlation) to 1 (perfect correlation). As Pearson's r takes the particular score values (the value describing one recommender's performance on one setup) of the algorithms into account, we additionally use another metric that only considers the order of the algorithms in a ranking: the *number of discordant pairs* d.[12] Given two rankings, the algorithms A and B are discordant, when in one ranking A performs better than B while in the other ranking B is better than A. Thus in our case of five algorithms, d is between 0 (the rankings agree completely) and 10 (one ranking is the reverse of the other).

Table V shows the mean pairwise (averaged over any pair of two different setups) values of r and d together with the standard deviations exemplarily for the metrics $\mathrm{pre@5}$, $\mathrm{rec@5}$, and MAP. We can observe that on no dataset we get perfect correlations. Generally, the correlations are rather high, but we clearly see that the rankings are inconsistent. The most stable are the rankings on *deli*. Here, only in every second pair of setups two recommenders change their order. The correlations on *cite* are only a little lower than on *deli* and on the two BibSonomy datasets the values are similar and again lower than those on *cite*: on average, in two rankings one or two pairs of recommenders have a different order.

Further, we computed which of the cores yield the ranking that is most consistent with the raw data. For all datasets these are the tas-graph-cores at levels 2 and 3, i.e., the two largest cores. More generally we could observe that higher levels (and thus higher densities) tend to yield results less consistent with the raw data. We conclude that in experiments cores with lower levels are preferable to others, since they resemble the original dataset more closely.

The consistency of the rankings also depends on the particular metric that is employed. In Figure 6 we see the mean pairwise values of r and d for $\mathrm{rec@}k$ and $\mathrm{pre@}k$ with k running from 1 through 10. We see that the behavior of the consistency measures over the levels is dataset-specific. For *deli* the consistency is quite stable for both metrics precision and recall. However, for the two BibSonomy datasets the values vary

[11] That is, 14 different setups for *book* and *publ* each and 28 and 31 setups for *cite* and *deli* respectively. These numbers are determined by the choice of levels (see Table III) and the exclusion of cores with only few users (see Section 4.1).

[12] The number of discordant pairs is closely related to the ranking correlation measure *Kendall's* τ. In fact, since all rankings have the same length of five (algorithms), and no two algorithms have equal scores in one ranking, we have $\tau = 1 - 0.2d$.

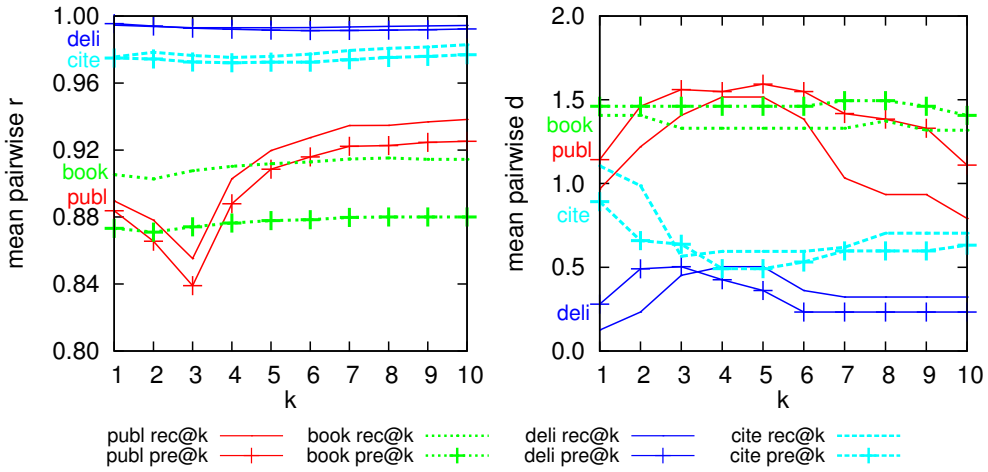


Fig. 6. The mean pairwise Pearson's correlation r and number of discordant pairs d over the cut-level k for the metrics $\mathrm{rec@}k$ and $\mathrm{pre@}k$.

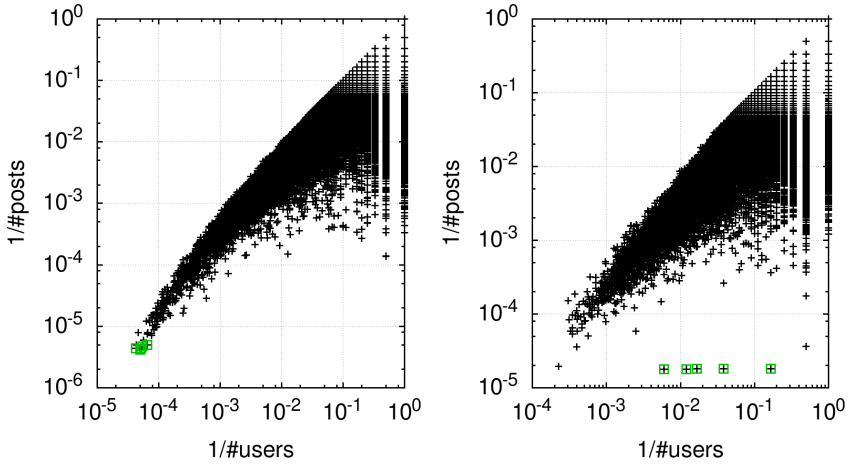
Table VI. For each dataset and each core type, the core levels l (among those considered in the previous experiments, cf. Table III) where the algorithm *most popular tags* outperforms *least popular tags* (column "mpt") or otherwise (column "lpt") according to precision at five ($\mathrm{pre@5}$) and recall at five ($\mathrm{rec@5}$). For both metrics the comparisons are identical except for the tas-graph-core of *publ* at level 5. The difference is indicated using the metric as superscript.

dataset	tas-graph-core		post-graph-core		post-set-core	
	mpt	lpt	mpt	lpt	mpt	lpt
publ	1, $5^{\mathrm{rec@5}}$, 6	2 – 4, $5^{\mathrm{pre@5}}$	1, 3 – 6	2	1, 3, 4	2
book	1, 4 – 6	2, 3	1, 3 – 6	2	1 – 4	–
deli	1 – 10, 20	–	1 –10, 20	–	1 – 10, 20	–
cite	1	2 – 10, 15	1, 4 – 10, 15	2, 3	1, 4 – 8	2, 3

and the highest consistency is achieved for $k = 10$, indicating that especially among top recommendations the recommenders' success changes with the setups. Finally, on *cite*, the difference in consistency is more dramatic when measured by the number of discordant pairs than with r. This means that recommenders switch places in the performance rankings although their scores develop similarly with changing levels or core types.

5.4. Exploiting Cores using LeavePostOut

To demonstrate a critical statistical flaw of the use of any core in connection with the LeavePostOut method, we employ the bogus *least popular tags* recommender that always suggests the rarest tags. It is expected that this method's scores should be inferior to those of the other algorithms. They are indeed, when the raw datasets are used – recall and precision always yield 0 and the MAP-score is below 10^{-4}. However, on the two BibSonomy datasets and on *cite* this changes once we use cores at a level $l > 1$: On many of the investigated cores, *least popular tags* actually outperforms *most popular tags* (and occasionally even *most popular tags by resource*). Table VI shows for the three core types those levels on which *least popular tags* yields better scores – measured in terms of precision and recall – than *most popular tags* or the other way around.



(a) *deli* (b) *cite*

Fig. 7. The distribution of the number of users per tag versus the number of posts per tag. Shown are the reciprocal values in a logarithmic plot to focus on the high-frequency tags. The top five tags of the *most popular tags* recommender are highlighted.

The algorithm *least popular tags* can profit in cases where the left-out post contains many rare tags: through LeavePostOut, these tags become even more rare and, in particular when they occurred exactly l times in a core at level l before LeavePostOut, they occur $l - 1$ times afterwards. Instead of being removed from the dataset like in the case of $l = 1$ (the raw data), for higher levels l they become the rarest tags in the core. We can observe that this effect is mitigated with an increasing core level and the scores of *least popular tags* tend to fall below those of *most popular tags*. Only on *deli* *least popular tags* is always worse than *most popular tags* (although its scores are still significantly higher than zero). This can be explained by a much higher average number of tag assignments per tag: 43.5 on *deli* compared to only 6.9, 12.8, and 18.3 on *publ*, *book*, and *cite*, respectively. The higher the number, the less likely it is to select posts with tags that occur exactly l times during LeavePostOut. The same argument explains why *least popular tags* falls behind *most popular tags* as the level increases: together with l also the average number of tag assignments per tag rises.

5.5. The Most Popular Baseline

In Section 5.1 we have seen that the *most popular tags* recommender performs very badly – especially on *cite*. To explain this phenomenon, recall that the most popular tags are computed based on their *post frequency*, i.e., the number of posts that contain a tag. Thus, if there are tags that are used extremely often by only a few users, they will be among the most popular tags and thus be recommended to many users. In Figure 7, for *deli* and *cite*, for each tag its post frequency is plotted against the number of users that have tagged at least one post with it. To put emphasis on those tags that occur most often, we have plotted the reciprocal values – and thus small values correspond to high frequencies – on a log-log scale. Relevant for the *most popular tags* recommender are the tags with the highest post frequency – these are the ones closest to the x-axis. We can see that the top five tags for *deli* are also close to the y-axis, which means they have both a high post frequency and a high user frequency. In contrast, the tags with the highest post frequency in *cite* have a rather low user frequency, therefore they are a bad recommendation for most of the users. A closer look at these top five tags (namely

"celegans", "elegans", "nematode", "caenorhabditiselegans", and "wormbase") reveals that they are all related to *Caenorhabditis elegans*,[13] a worm which is frequently used as a model organism in biology. These five tags were very frequently used (27 735 times) in the same posts by two users (with IDs 33569 and 28123) and less frequently by 165, 81, 58, 24, and 4 other users, respectively (cf. also the related tags for "celegans" on CiteULike[14]). Thus, the posts from the two users were likely created automatically but were not detected by the approaches described in Section 4.1, since they describe the content of posts and not their creation process (as do tags like "imported"). Also, if we recall the evaluation procedure LeavePostOut, which randomly picks *one post for every user*, it becomes clear that these most popular tags are only a weak baseline. A better choice would be to measure the popularity of tags based on user frequency.

6. CONCLUSION

We have analyzed the use of cores for the evaluation of tag recommendations. The main contribution of the paper is the extension of the framework of core constructions through the introduction of set-cores as generalization of graph-cores. This new core type allows us to transfer the idea of cores to datasets without imposing a graph structure on them. In contrast to graph-cores, it allows the use of several thresholds at once and for flexible combinations of individual and combined thresholds for different entities. We have successfully used them in tag recommender benchmarking experiments to avoid the problem of diminished posts.

6.1. Lessons Learned

In the experiments, we have confirmed that benchmarking results not only depend on the dataset and preprocessing procedures, but also on the chosen cores and that using cores for offline evaluation has its pitfalls: The use of tas-graph-cores and post-graph-cores results in diminished posts (post with fewer tags than they had originally) in the dataset. With the use of post-set-cores we have presented a suitable solution for this problem. The anomaly of the successful *least popular tags* recommender directly exploits the combination of cores and LeavePostOut. For other recommenders it is unclear whether and how they can profit from the particular setup or the artificial rareness of the left-out tags. We have also confirmed that recommenders perform differently in different core setups of the same dataset. Thus, focusing on one particular core can produce non-stable results. Evaluating the performance of recommenders on another core type or at another core level might cause changes in the results. There is no guarantee that a recommender performing best in one setup is also the best in another setup (even on the same dataset). The correlations of recommender rankings over various setups were relatively high. Yet, the fact that in a comparison of different algorithms some switch ranks on different cores suggests that the choice of the core and its level is even more critical for the comparison of algorithms with similar performance and for the optimization of parameterized algorithms (where usually scores change only little through fine adjustments of parameter values).

6.2. Recommendations for Future Tag Recommender Benchmarking Experiments

Following our findings, we can draw the following conclusions for future experiments with recommender algorithms:

— In general, the comparison of tag recommender algorithms should always be performed directly on the raw data or on several core types and levels.

[13] http://en.wikipedia.org/wiki/Caenorhabditis_elegans
[14] http://www.citeulike.org/tag/celegans

— Differences in the rankings, resulting from such comparisons, indicate strengths or weaknesses of individual algorithms in the presence of datasets with different densities.
— We could observe that even cores at higher levels still yield correlated results to those of the raw data. It is therefore worth to compare recommenders on several of these smaller subsets of the raw data to get a first impression of their overall performance, before running the computationally more expensive experiments on the raw data.
— We suggest to still use small choices for the core level (thus larger cores), since they yield more consistent results with the raw dataset.
— We recommend not to run an evaluation on only one arbitrary chosen core, but to carefully select several levels that suit the investigated use case. The particular choice of the core level should be motivated by the use-case – examples are given in Section 2.3.3.
— To avoid the problem of diminished posts, post-set-cores should be used. Investigating posts with all their tags (as they are in the raw data) is closer to the real online use case. Allowing diminished posts increases the divide between offline evaluation and actual online usage further.
— To tailor the test dataset to a particular use case, post-set-cores – in contrast to tas-graph-cores and post-graph-cores – allow to impose thresholds individually for each dimension of the data.

6.3. Future Research

We have shown that the choice of core type and core level has an impact on a benchmarking experiment's result. It is well-known that there are many other parameters of the experimental setup which are influential as well. While, for instance, the choice of the evaluation metric can often be justified by the use case – e.g., by the design of the service in which the recommendations are provided – other choices are often rather arbitrary or for the sake of minimizing the computational effort. These aspects of the experimental setup include the method of splitting test and training data (e.g., LeavePostOut, different variations where more than one post is removed from the dataset, or time splits), the sampling of the training data (e.g., selecting some randomly chosen post per user, or selecting the most recent post of each user, or using some user-independent selection of posts), the preprocessing of the data, etc. Further experiments could reveal the influence of these choices on the results of tag recommender benchmarking as well as insights about how particular algorithms can profit or suffer from the chosen setup.

The fact that different core setups yield different recommender rankings is an indicator that different algorithms have strengths and weaknesses when dealing with rather sparse or with more dense data. In Section 5.1 we have shown that performance differences occur even through choosing only the test posts (the user-resource combinations to recommend tags for) from different regions of the data (i.e., from the tas-graph-core, the post-graph-core, or the post-set-core) while leaving the training dataset the same (i.e., in our experiments the tas-graph-core). This encourages approaches using different recommender algorithms in different situations: A recommender that performs well (compared to others) on sparse data can be applied for new (or sparsely connected) users and resources. An algorithm that dominates on the more dense datasets (higher core levels) can be chosen for user-resource pairs from an already dense section of the data. The dynamic selection of the appropriate recommender – depending on the user and resource at hand – can be investigated as a machine learning problem.

An open aspect regarding the offline evaluation of a recommender is the current way of distinguishing good recommendations from bad ones. In the current setup a tag is only a good suggestion if it fits the user's actual choice exactly (in our experiments ignoring upper and lower case). Thus, for example, the recommended tag "work" is considered a bad recommendation even if the user had in fact used seemingly related tag "working" where in an online setting the user might have accepted the recommended tag. Several approaches to "softer" measures are conceivable: word stemming of both recommended tags and the actual tags (actually the conversion to lower case is already a mild form of stemming), differentiating between exact and close fits of recommended tags, etc. In Section 3 we have already mentioned the approach by [Mishne 2006] using string distance to compare tags. Different evaluation scenarios could be compared in a similar setup like in this paper, variating the evaluation functions instead of core type and level. Such experiments should also be accompanied by a user-study to investigate, for instance, which forms of stemming are acceptable for many users.

Finally, since set-cores can be constructed on arbitrary sets, they can be used in the analysis of all kinds of datasets. In the related work on cores in Section 3, we have mentioned several applications of graph-cores for diverse purposes, such as community detection, data visualization, or the discovery of dynamics in datasets. It is now possible to adapt these methods using set-cores and thus to extend their analytic capabilities – through new, flexible, and multi-valued property functions – and their scope.

References

Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of Data Characteristics on Recommender Systems Performance. *ACM Trans. Manage. Inf. Syst.* 3, 1, Article 3 (April 2012), 17 pages. DOI:http://dx.doi.org/10.1145/2151163.2151166

Adel Ahmed, Vladimir Batagelj, Xiaoyan Fu, Seok-Hee Hong, Damian Merrick, and Andrej Mrvar. 2007. Visualisation and analysis of the internet movie database. In *6th Int. Asia-Pacific Symposium on Visualization*. 17–24. DOI:http://dx.doi.org/10.1109/APVIS.2007.329304

José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2005. K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. *CoRR* cs/0511007 (2005). http://arxiv.org/abs/cs/0511007

Ralitsa Angelova, Marek Lipczak, Evangelos Milios, and Paweł Prałat. 2008. Characterizing a social bookmarking and tagging network. In *Proceedings of the Mining Social Data Workshop*. ECAI, 21–25.

Pierpaolo Basile, Domenico Gendarmi, Filippo Lanubile, and Giovanni Semeraro. 2007. Recommending Smart Tags in a Social Bookmarking System. In *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*. 22–29. http://www.kde.cs.uni-kassel.de/ws/eswc2007/proc/RecommendingSmartTags.pdf

Vladimir. Batagelj and Matjaž M. Zaveršnik. 2002. Generalized cores. *CoRR* cs.DS/0202039 (2002). http://arxiv.org/abs/cs/0202039

Vladimir Batagelj and Matjaž Zaveršnik. 2011. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classification* 5, 2 (2011), 129–145. DOI:http://dx.doi.org/10.1007/s11634-010-0079-y

Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner. 2007. Generating graphs with predefined k-core structure. In *Proceedings of the European Conference of Complex Systems*.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30, 1–7 (1998), 107–117.

Andrew Byde, Hui Wan, and Steve Cayzer. 2007. Personalized Tag Recommendations via Tagging and Content-based Similarity Metrics. In *Proceedings of the International Conference on Weblogs and Social Media*.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 39–46. DOI:http://dx.doi.org/10.1145/1864708.1864721

Reinhard Diestel. 2005. *Graph Theory* (3 (electronic edition) ed.). Springer-Verlag Heidelberg, New York. I–XVI, 1–344 pages. http://www.math.ubc.ca/~solymosi/2007/443/GraphTheoryIII.pdf

Stephan Doerfel and Robert Jäschke. 2013. An analysis of tag-recommender evaluation procedures. In *Proceedings of the 7th Conference on Recommender systems (RecSys '13)*. ACM, New York, NY, USA, 343–346. DOI:http://dx.doi.org/10.1145/2507157.2507222

Folke Eisterlehner, Andreas Hotho, and Robert Jäschke (Eds.). 2009. *ECML PKDD Discovery Challenge 2009 (DC09)*. CEUR-WS.org, Vol. 497. http://ceur-ws.org/Vol-497

Christos Giatsidis, Dimitrios M. Thilikos, and Michalis Vazirgiannis. 2011. Evaluating Cooperation in Communities with the k-Core Structure. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. 87–93. DOI:http://dx.doi.org/10.1109/ASONAM.2011.65

Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (2004), 5–53. DOI:http://dx.doi.org/10.1145/963770.963772

Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. 2006. Information retrieval in folksonomies: search and ranking. In *The Semantic Web: Research and Applications (LNCS)*, Vol. 4011. Springer, Berlin/Heidelberg, 411–426.

Andreas Hotho, Beate Krause, Dominik Benz, and Robert Jäschke (Eds.). 2008. *ECML PKDD Discovery Challenge 2008 (RSDC'08)*.

Svante Janson, Tomasz Luczak, and Andrzej Rucinski. 2000. *Theory of random graphs*. John Wiley and Sons, New York; Chichester. http://www.amazon.com/Random-Graphs-Svante-Janson/dp/0471175412

Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. 2007. Tag recommendations in folksonomies. In *Proc. 11th European Conf. Principles and Practice of Knowledge Discovery in Databases (LNCS)*, Vol. 4702. Springer, Berlin/Heidelberg, 506–514.

Robert Jäschke, Andreas Hotho, Folke Mitzlaff, and Gerd Stumme. 2012. Challenges in tag recommendations for collaborative tagging systems. In *Recommender Systems for the Social Web*. Intelligent Systems Reference Library, Vol. 32. Springer, Berlin/Heidelberg, 65–87. DOI:http://dx.doi.org/10.1007/978-3-642-25694-3_3

Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. 2008. Tag recommendations in social bookmarking systems. *AI Communications* 21, 4 (2008), 231–247. DOI:http://dx.doi.org/10.3233/AIC-2008-0438

George Karypis. 2001. Evaluation of Item-Based Top-N Recommendation Algorithms. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM '01)*. ACM, New York, NY, USA, 247–254. DOI:http://dx.doi.org/10.1145/502585.502627

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM* 46, 5 (1999), 604–632. DOI:http://dx.doi.org/10.1145/324133.324140

Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. 2009. Latent dirichlet allocation for tag recommendation. In *Proc. 3rd Conf. on Recommender Systems*. ACM, New York, NY, USA, 61–68. DOI:http://dx.doi.org/10.1145/1639714.1639726

Marius Kubatz, Fatih Gedikli, and Dietmar Jannach. 2011. LocalRank - Neighborhood-Based, Fast Computation of Tag Recommendations. In *E-Commerce and Web Technologies*, Christian Huemer and Thomas Setzer (Eds.). Lecture Notes in Business Information Processing, Vol. 85. Springer Berlin Heidelberg, 258–269. DOI:http://dx.doi.org/10.1007/978-3-642-23014-1_22

Marek Lipczak, Yeming Hu, Yael Kollet, and Evangelos Milios. 2009. Tag Sources for Recommendation in Collaborative Tagging Systems, See Eisterlehner et al. [2009], 157–172. http://ceur-ws.org/Vol-497

Feichao Ma, Wenqing Wang, and Zhihong Deng. 2013. TagRank: A new tag recommendation algorithm and recommender enhancement with data fusion techniques. In *Social Media Retrieval and Mining*, Shuigeng Zhou and Zhiang Wu (Eds.). Communications in Computer and Information Science, Vol. 387. Springer, Berlin/Heidelberg, 80–91. DOI:http://dx.doi.org/10.1007/978-3-642-41629-3_7

Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. SoRec: Social Recommendation Using Probabilistic Matrix Factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08)*. ACM, New York, NY, USA, 931–940. DOI:http://dx.doi.org/10.1145/1458082.1458205

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York.

Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*. AAAI, 187–192.

Gilad Mishne. 2006. AutoTag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*. ACM Press, New York, NY, USA, 953–954. DOI:http://dx.doi.org/10.1145/1135777.1135961

Elena Montañés, José Ramón Quevedo, Irene Díaz, Raquel Cortina, Pedro Alonso, and José Ranilla. 2011. TagRanker: learning to recommend ranked tags. *Logic Journal of IGPL* 19, 2 (2011), 395–404. DOI:http://dx.doi.org/10.1093/jigpal/jzq036

Alexandros Nanopoulos, Dimitrios Rafailidis, and Ioannis Karydis. 2013. Matrix factorization with content relationships for media personalization. In *Proc. 11th Int. Conf. Wirtschaftsinformatik*. 87–101.

Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, and Steve Lawrence. 2001. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI'01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 437–444.

Pearl Pu, Li Chen, and Rong Hu. 2011. A user-centric evaluation framework for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11)*. ACM, New York, NY, USA, 157–164. DOI:http://dx.doi.org/10.1145/2043932.2043962

Maryam Ramezani. 2011. Improving graph-based approaches for personalized tag recommendation. *Journal of Emerging Technologies in Web Intelligence* 3, 2 (2011).

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Schmidt-Thieme Lars. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461. http://portal.acm.org/citation.cfm?id=1795114.1795167

Steffen Rendle and Lars Schmidt-Thieme. 2009. Factor Models for Tag Recommendation in BibSonomy, See Eisterlehner et al. [2009], 235–242. http://ceur-ws.org/Vol-497

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. 10th Int. Conf. World Wide Web*. ACM, New York, NY, USA, 285–295. DOI:http://dx.doi.org/10.1145/371920.372071

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. 2000. Application of Dimensionality Reduction in Recommender System – A Case Study. In *ACM WEBKDD Workshop*.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. 2002. Methods and Metrics for Cold-start Recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. ACM, New York, NY, USA, 253–260. DOI:http://dx.doi.org/10.1145/564376.564421

Stephen B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (1983), 269 – 287. DOI:http://dx.doi.org/10.1016/0378-8733(83)90028-X

Paul Seitlinger, Dominik Kowald, Christoph Trattner, and Tobias Ley. 2013. Recommending tags with a model of human categorization. In *Proceedings of the 22nd International Conference on Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 2381–2386. DOI:http://dx.doi.org/10.1145/2505515.2505625

Guy Shani and Asela Gunawardana. 2011. Evaluating recommendation systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, 257–297. DOI:http://dx.doi.org/10.1007/978-0-387-85820-3_8

Sanjay Sood, Sara Owsley, Kristian Hammond, and Larry Birnbaum. 2007. TagAssist: Automatic Tag Suggestion for Blog Posts. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*. Boulder, Colorado, USA.

Milan Vojnovic, James Cruise, Dinan Gunawardena, and Peter Marbach. 2007. *Ranking and Suggesting Tags in Collaborative Tagging Applications*. Technical Report MSR-TR-2007-06. Microsoft Research.

Jyun-Cheng Wang and Chui-Chen Chiu. 2008. Recommending trusted online auction sellers using social network analysis. *Expert Syst. Appl.* 34, 3 (2008), 1666–1679. DOI:http://dx.doi.org/10.1016/j.eswa.2007.01.045

Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. 2006. Towards the Semantic Web: Collaborative Tag Suggestions. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*. Edinburgh, Scotland.

Haohua Zhang, Hai Zhao, Wei Cai, Jie Liu, and Wanlei Zhou. 2010. Using the k-core decomposition to analyze the static structure of large-scale software systems. *The Journal of Supercomputing* 53, 2 (2010), 352–369. DOI:http://dx.doi.org/10.1007/s11227-009-0299-0

Received January 2014; revised August 2014; accepted November 2014