The VLDB Journal manuscript No.
(will be inserted by the editor)

The Social Bookmark and Publication Management System BibSonomy

A Platform for Evaluating and Demonstrating Web 2.0 Research

Dominik Benz[1] · **Andreas Hotho**[2] · **Robert Jäschke**[1,3] · **Beate Krause**[1,2] · **Folke Mitzlaff**[1] · **Christoph Schmitz**[1,3] · **Gerd Stumme**[1,3]

Received: date / Accepted: date

Abstract Social resource sharing systems are central elements of the Web 2.0 and use the same kind of lightweight knowledge representation, called *folksonomy*. Their large user communities and ever-growing networks of user-generated content have made them an attractive object of investigation for researchers from different disciplines like Social Network Analysis, Data Mining, Information Retrieval or Knowledge Discovery. In this paper, we summarize and extend our work on different aspects of this branch of Web 2.0 research, demonstrated and evaluated within our own social bookmark and publication sharing system BibSonomy, which is currently among the three most popular systems of its kind. We structure this presentation along the different interaction phases of a user with our system, coupling the relevant research questions of each phase with the corresponding implementation issues. This approach reveals in a systematic fashion important aspects and results of the broad bandwidth of folksonomy research like capturing of emergent semantics, spam detection, ranking algorithms, analogies to search engine log data, personalized tag recommendations and information extraction techniques. We conclude that when integrating a real-life application like BibSonomy into research, certain constraints have to be considered; but in general, the tight interplay between our scientific work and the running system has made BibSonomy a valuable platform for demonstrating and evaluating Web 2.0 research.

[1] Knowledge & Data Engineering Group
Interdisciplinary Research Center for Information Systems Design
University of Kassel
Wilhelmshöher Allee 73, 34121 Kassel, Germany
E-mail: {lastname}@cs.uni-kassel.de

[2] Data Mining and Information Retrieval Group
University of Würzburg
Am Hubland, 97074 Würzburg, Germany
E-mail: {lastname}@informatik.uni-wuerzburg.de

[3] L3S Research Center
Appelstr. 9a, 30167 Hannover, Germany
E-mail: {lastname}@l3s.de

Keywords BibSonomy · Collaborative Tagging · Web 2.0 · Folksonomy · Data Mining

1 Introduction

Complementing the Semantic Web effort, a new breed of so-called 'Web 2.0' applications has become an integral part of the Web's landscape during the last years. These include user-centric publishing and knowledge management platforms like wikis, blogs, and social resource sharing tools. The enormous popularity of these systems, including their large user communities and ever-growing networks of user-generated content have made them an attractive object of investigation for researchers from different disciplines like social network analysis, data mining, information retrieval or knowledge discovery. In order to cover an important part within the broad bandwidth of 'Web 2.0 research', we focus here on social resource sharing systems, which all use the same kind of lightweight knowledge representation, called *folksonomy* [50].

Social resource sharing systems are web-based systems that allow users to upload all kinds of resources, and to label them with arbitrary words, so-called *tags*. The systems can be distinguished according to what kind of resources are supported. Flickr,[1] for instance, allows the sharing of photos while Delicious[2] allows sharing of bookmarks, just to mention a few. One reason for their immediate success is the fact that no specific skills are needed for participating, and that these tools yield immediate benefit for the individual user (e. g., organizing ones bookmarks in a browser-independent, persistent fashion) without too much overhead.

These systems, along with their underlying network of users, tags and resources have been in the center of the research activities at our group during roughly the last four years. Crucial questions for each researcher interested in this topic are (i) how

[1] http://www.flickr.com/
[2] http://www.delicious.com/

and where to get large real-world datasets and (ii) how to evaluate research results in a realistic setting. For both reasons, we started to build our own resource sharing system, called BibSonomy, which allows sharing bookmarks and bibliographic references simultaneously. Started as a student project at our group in spring 2005, it quickly grew out of the prototype status and attracted until today roughly 5,000 users, putting it – to the best of our knowledge – among the three most popular social publication sharing systems at present.[3]

In this paper, we summarize our work on different aspects of Web 2.0 research, demonstrated and evaluated within BibSonomy. As many research questions are geared towards improving the user's experience with the system, we will structure our summary along the different interaction phases of a user with BibSonomy. Section 2 starts with giving an overview of a typical use case. After laying the groundwork with an explanation of the system's architecture in Section 3, we will detail on each interaction phase and identify the corresponding research questions:

- In order to support the user while *browsing* and *searching* in the system (Section 4), we have done an analysis of semantic similarity measures [10] among tags in order to enable a 'semantic' direction of browsing. Furthermore, we investigated automatic spam detection methods [33] to prevent our users from wading through inappropriate content. A core question when searching for specific content is how to rank the resources by relevance. To this end we have developed *FolkRank* [23], an adaptation of the well-known PageRank algorithm [6] to folksonomies.
- The process of *entering* new content (Section 5) can be supported in various ways. The description of our work on assisting the user during the annotation process by tag recommendations [29] is complemented by presenting a framework for online evaluation of different recommendation algorithms [30]. In addition, we provide a first analyis of the 'copy network' of BibSonomy, which emerges when users copy entries from other users.

Section 6 then shifts perspective to the content itself, how it is represented internally and how it can be accessed in various ways, e. g., via the web interface or a REST-based API. In order to show the tight interplay between our research and the running system, we splitted each section – when possible – by first detailing on research questions and how we tackled them, and then on implementation issues and how our results have been reflected in BibSonomy. Throughout the paper, we adhere to a formal model of a folksonomy [23] defined as follows:

A folksonomy is a tuple $\mathbb{F} := (U, T, R, Y, \prec)$ where

- U, T, and R are finite sets, whose elements are called *users*, *tags* and *resources*, resp.,

[3] Together with http://www.citeulike.org/ and http://www.connotea.org/.

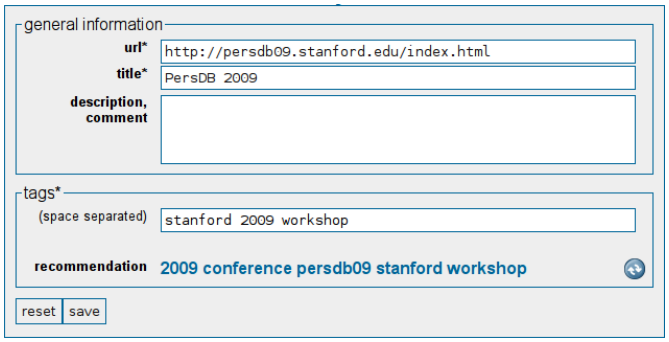


Figure 1 Categorizing a bookmark in BibSonomy by assigning the tags 'stanford', '2009' and 'workshop'.

- Y is a ternary relation between them, i. e., $Y \subseteq U \times T \times R$, whose elements are called tag assignments (*tas* for short), and
- $\prec$ is a user-specific subtag/supertag-relation, i. e., $\prec \subseteq U \times T \times T$, called *is-a relation*.

Users are typically described by their user ID, and tags may be arbitrary strings. What is considered as a resource depends on the type of system, e. g., in BibSonomy they are either URLs or publication entries.

For convenience we also define the set P of all *posts* as $P := \{(u,S,r) \mid u \in U, r \in R, S = T(u,r), S \neq \emptyset\}$ where, for all $u \in U$ and $r \in R$, $T(u,r) := \{t \in T \mid (u,t,r) \in Y\}$ denotes all tags the user u assigned to the resource r.

If we disregard the is-a relation, we can simply note a folksonomy as a quadruple $\mathbb{F} := (U,T,R,Y)$. This structure is known in Formal Concept Analysis [51] as a *triadic context* [34, 49]. An equivalent view on this structure is that of a tripartite (undirected) hypergraph $G = (V,E)$, where $V = U \dot\cup T \dot\cup R$ is the set of nodes, and $E = \{\{u,t,r\} \mid (u,t,r) \in Y\}$ is the set of hyperedges.

2 Usage of the System

A good portion of a researcher's day-to-day business consists of doing literature research. Especially when a new project is launched, a lot of literature must be categorized for further reference. Literature research nowadays typically starts by querying a major web search engine, resulting in a set of (potentially) relevant resources. BibSonomy allows the researcher to categorize and archive both bookmarks and literature references while reviewing a resource of interest in his web browser.

For storing a reference to a resource in BibSonomy, the user has to provide corresponding meta information (such as the title) as well as a non-empty set of arbitrarily chosen keywords (so-called *tags*), describing the resource, the user's opinion about the resource, the project for which the resource is relevant for and so on. Figure 1 shows BibSonomy's input form for categorizing a bookmark. Of course, when scientific articles are to be categorized, more meta information (like the author, journal, publisher, ...) must be provided to allow unambiguous references. To relieve the user from this tedious work,

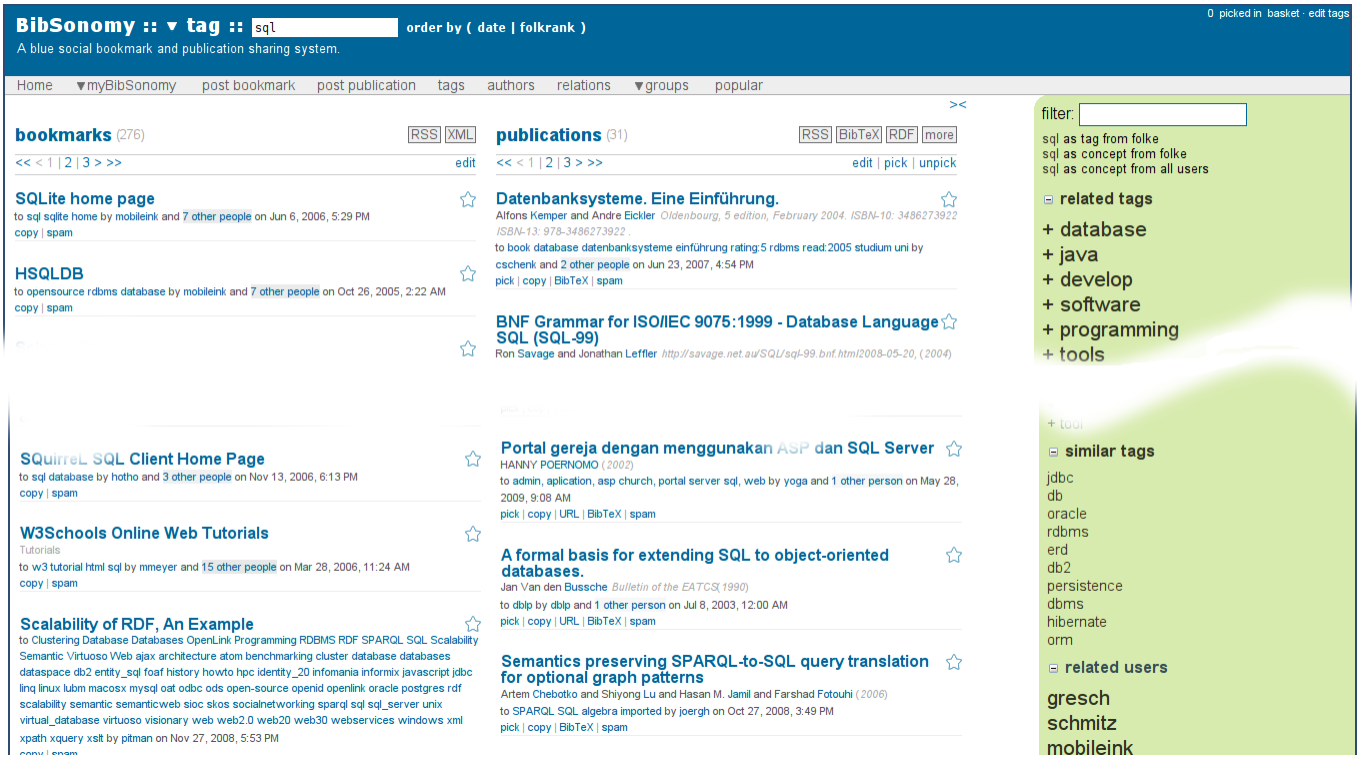


Figure 2 Searching for posts related to the tag 'sql'.

BibSonomy incorporates so-called scraping techniques for automatically extracting this information from the web page the user is visiting (as described in Section 5.2.2). To further ease the process of tag assignment, BibSonomy presents recommended tags from which the user might choose a subset, e. g., the tags '2009', 'conference', 'persdb09', 'stanford' and 'workshop' in Figure 1. Our work on evaluating methods for tag recommendations is described in Section 5.2.1.

With each annotated resource, the user contributes to BibSonomy's collaborative database, making his posts available to others. Thus, BibSonomy offers a way for searching and browsing resources, taking other users' annotations into account. Currently, more than 350,000 bookmarks and 650,000 publication references are stored in BibSonomy. Nevertheless, allowing freely typed tags leads to problems which all collaborative tagging systems have to face. Golder et al. [14] identified three major problems as polysemy, synonymy, and level variation. Polysemy refers to tags which have several meanings (e. g., turkey), synonymy to situations, where several tags share a common meaning (above all morphological variations as 'web20', 'web2.0', . . .) and level variation to situations where different people are using tags from different levels of abstraction (e. g., 'database' vs. 'mysql').

To deal with these problems, BibSonomy assists the user while searching and browsing in several ways. By applying the FolkRank algorithm to BibSonomy's underlying data structure, resources may be ranked by relevance to a given tag. Note that this ranking also considers resources whose annotations do not contain the given tag (see Section 4.2.3). Figure 2 shows BibSonomy's interface while searching for the tag 'sql', with results ordered by relevance. This is only one possible ordering of a result set – others are presented in Section 4.1.3.

Additionally, as shown in Figure 2, a list of similar and related tags is presented to the user, providing access to resources which are (by trend) annotated with synonyms or tags from different levels of abstraction, respectively (see Sections 4.1.1 and 4.2.1). The user does not see the huge part of more or less dubious posts, placed by spammers who are abusing social bookmarking services. Without any protection, reasonable content would to be lost amidst these. Therefore, users of BibSonomy are semi-automatically classified into normal and spam users, enabling separation of spam posts (see Sections 4.1.2 and 4.2.2).

Finally, when a project is finished and the corresponding report has to be written, all relevant publications have to be referenced. For that, BibSonomy offers export facilities in a variety of formats like BibTeX or RIS, suitable for integration into different text processing systems for automatic generation of bibliographies (see Section 6.2.2).

In summary, BibSonomy integrates seamlessly into the different phases of doing research work, aiming to support the researcher at each stage with suitable techniques.

3 The BibSonomy Architecture

In order to lay the groundwork for an understanding of the BibSonomy system in the following sections, we now detail the technical and conceptual aspects of its architecture which has first been described in [21, 25] and since then quite evolved. BibSonomy is a web application written in Java, which is based

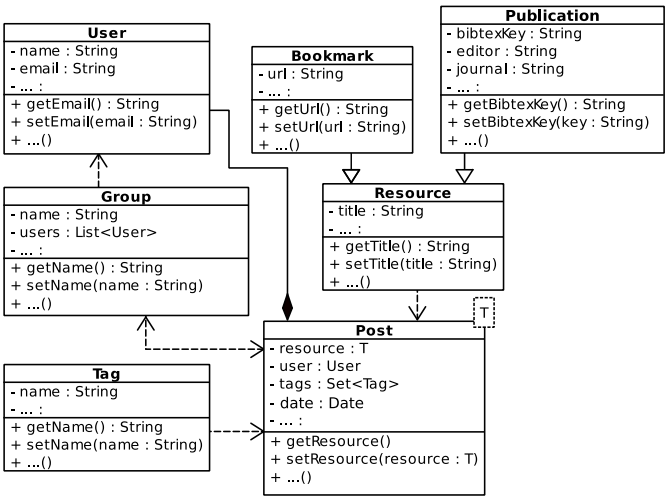


Figure 3 A simplified UML class diagram of BibSonomy's data model.

on Java Server Pages[4] and Java Servlet[5] technology. To separate the logical handling of data from its presentation, BibSonomy follows the Model View Controller (MVC) programming paradigm [42] implemented by Spring's Web MVC framework.[6] A relational database is used as back-end (see Section 6.1 for details). The system itself has a modular structure, in which several components encapsulate the different functionalities like, e. g., database access or scraping facilities. The modularisation has been performed by a reimplementation of large parts of BibSonomy, after having gained three years of experience with the first version of the system. The modularisation turned out to be necessary since maintenance and parallel code development became more and more difficult.

After giving some details on the underlying data model, the components of the reimplemented version of BibSonomy are described briefly in order to clarify the building blocks of the system. We close this section by exemplifying how the components work together in a typical control flow when answering a request.

3.1 Data Model

BibSonomy's data model (see Figure 3) is centered around the core classes *Tag*, *User* and *Resource*, implementing the formal model of a folksonomy as described in Section 1. For technical reasons, each supported resource type is modelled by a separate sub class of the *Resource* class (*Bookmark* and *Publication* for now). Tag assignments which belong together are bundled in the *Post* class. Finally, BibSonomy allows users to form groups (e. g., to jointly collect posts) which are modeled by the *Group* class.

[4] http://java.sun.com/products/jsp
[5] http://java.sun.com/products/servlets
[6] http://www.springsource.org/

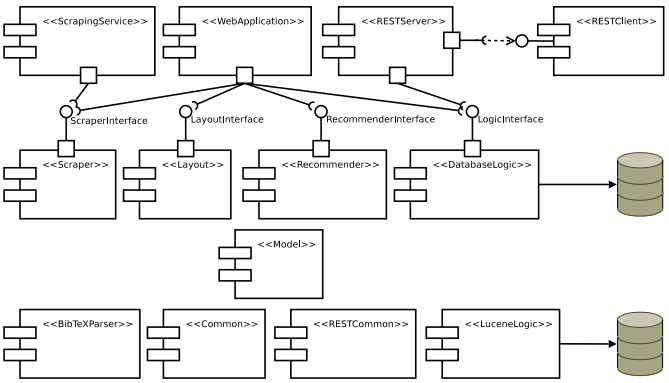


Figure 4 The UML component diagram of BibSonomy's components. All components build upon the *Model* which contains the classes that represent BibSonomy's data model. The *ScrapingService*, *WebApplication*, and *RESTServer* are the connection to the web.

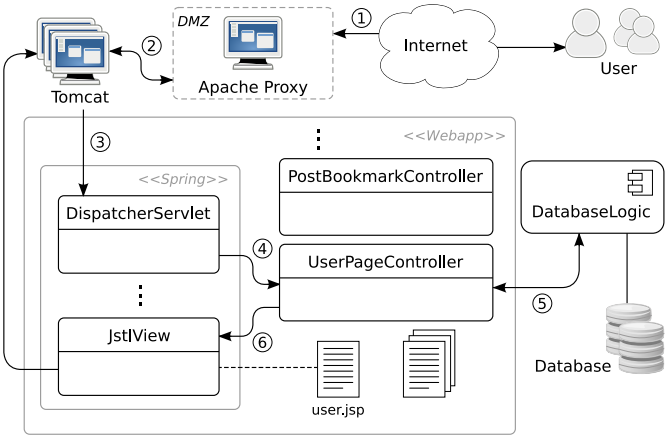


Figure 5 Diagrammatic presentation of the control flow in BibSonomy. Exemplary diagrammatic presentation of the control flow in BibSonomy.

3.2 Components

BibSonomy is built up from several *components* which encapsulate the different functionalities of the system. We here briefly describe the main components of the system which are depicted in Figure 4. The source code of components marked with an asterisk (*) in the following list is available under an (L)GPL license at http://dev.bibsonomy.org/.

BibTeXParser* Parses BibTeX strings and files into Java objects and is thus involved in every incoming publication post request.

Common* Most general-purpose classes, exceptions, and utility functions used by several other modules are contained in this component. Among others, it contains methods to handle sending of e-mails, parsing of XML, hashing, validation, and web crawling.

DatabaseLogic Handles all database access in BibSonomy by implementing the *LogicInterface* which describes the methods that operate on the objects of the model, e. g., *storePost*, *updateUser*, *getGroup*, ... It uses a relational database as back-end (cf. Section 6.1).

Layout* Implementing the *LayoutInterface*, this component provides rendering of publication posts using JabRef[7] export filters. For details see Section 6.2.2.

LuceneLogic Complementing the *DatabaseLogic*, this component provides full-text search over posts (e. g., title, tags, authors, etc.) using Apache Lucene[8].

Model* This component defines the underlying data model of BibSonomy as described in Section 3.1. In particular, all objects used for interacting with the *LogicInterface* are defined here. An important part of the model is the definition of the XML Schema of BibSonomy's REST API (see Section 6.2.3).

RESTClient* The REST client API, as counterpart of the *RESTServer*, acts as a library for Java programmers to connect their programs to the REST API of BibSonomy without caring about the underlying XML/HTTP-based interaction. Put simply, it provides remote access to the database by implementing (parts of) the *LogicInterface*.

RESTCommon* Common things needed by both the REST server and the REST client, in particular enumeration types, exceptions, and the XML renderer.

RESTServer* The REST server of BibSonomy offers REST-based access to the *LogicInterface* using XML over HTTP.

Recommender The tag recommendations shown during posting a bookmark or publication are generated by this component which implements the *RecommenderInterface*. For a detailed description of BibSonomy's tag recommendation framework see Section 5.2.1.

Scraper* More than 60 screen scrapers which extract publication metadata from various digital libraries (see Section 5.2.2). They are assembled into a chain of responsibility which also implements the *ScraperInterface*.

ScrapingService* A standalone web application representing a lightweight web service which allows to access the scraping facilities of the *Scraper* component.

WebApplication The web pages which can be accessed at http://www.bibsonomy.org/ are served by this central component. Intensively using the Spring framework, there are controllers for each of the different pages of the web application which are coupled to views by Spring. The URL scheme of the web application is further described in Section 6.2.1.

3.3 Control Flow

In order to demonstrate the interaction between some of the above-mentioned components, Figure 5 exemplarily shows how the request 'http://www.bibsonomy.org/user/nepomuk/fca' is processed and how the flow of control is managed within the system. BibSonomy itself is running in an Apache Tomcat[9] servlet container. Before getting there, the incoming request is (1) processed by an Apache web server,[10] running in a demilitarized zone (DMZ) and acting as a proxy and load balancer for several Tomcat instances (2). This setup contributes to fulfil our high availability requirements. Each instance hosts the BibSonomy web application where (3) the user request is passed to the *DispatcherServlet* of the Spring MVC framework and (4) delegated to the responsible controller. In this scenario, the *UserPageController* is invoked, which (5) retrieves all public posts of the user 'nepomuk' with the tag 'fca', sorted by date in descending order from the database. The compiled model is (6) passed to Spring's view resolver, which in this case delegates to Spring's *JstlView*, rendering an HTML page using an appropriate JSP-file. This flow of control demonstrates our modular design, which makes it possible to exchange one component (e. g., the database layer or the view component) by a new implementation.

4 Browsing and Searching the System

In the web pages of social bookmarking systems, the folksonomy structure is translated directly to hyperlinks. Each occurrence of a resource name, a tag, or a user name is a hyperlink pointing to the corresponding entity. This richly linked structure allows for serendipitous browsing. A typical retrieval session starts with typing in a tag. Depending on the output, one may either refine the query (e. g., by choosing an additional tag from the displayed list of 'related tags'), continue with the deeper analysis of a specific resource, or analyse a user who seems to have bookmarked interesting resources.

There are (at least) three observations/features that facilitate serendipitous browsing in social bookmarking systems and that are inherent to/built-in in BibSonomy: the small world property of folksonomies, measures that compute semantic closeness of tags, and the removal of spam. The latter two aspects are discussed in the next subsection, while implementation issues are described in Section 4.2.

Serendipitous browsing is facilitated by the fact that the graph of a folksonomy has the small world property. On the one hand, this means that related entities are clustered together in the folksonomy – and thus in the hyperlink graph of the bookmarking system – and on the other hand that it is possible to reach most of the other content of the system by very few clicks. For a detailed discussion we refer the reader to [9].

Many different measures for the semantic closeness of tags are used in research papers on folksonomies, but their selection is usually rather ad hoc and does not follow any scientific principle. In Section 4.1.1 we discuss the properties of different measures and derive arguments for the selection of a suitable measure. The implementation of some of these measures in BibSonomy will be discussed in Section 4.2.

A big hinderance for the usage of many internet-based services – and in particular for web 2.0 systems – is spam. BibSo-

[7] http://jabref.sourceforge.net/
[8] http://lucene.apache.org/
[9] http://tomcat.apache.org/
[10] http://httpd.apache.org/

nomy, for instance, contains more than 90 % spam. To maintain the usability, the system administrators have thus to constantly remove this spam. The implementation of BibSonomy's spam management framework is described in Section 4.2.2. It makes use of machine learning techniques which are discussed in Section 4.1.2.

Despite the fact that folksonomies allow for serendipitous discovery of interesting content, their ever-growing human-annotated collection of documents is also an attractive resource for users searching for specific contents. In contrast to a browsing user, a user searching in BibSonomy has a specific information need (e. g., finding research papers on a certain topic). Finding relevant content within a collection of documents has been investigated in depth in the field of information retrieval; however, as the underlying structure of a folksonomy differs fundamentally from the structure of repositories like the World Wide Web, existing approaches and algorithms need to be adapted.

In Section 4.1.3, we report on our work on the adaption of the well-known PageRank algorithm [6] to folksonomies and finally on an approach for topic-specific ranking in folksonomies – the *FolkRank* algorithm [23]. Its integration into BibSonomy is described in the implementation Section 4.2.3.

4.1 Research Questions

4.1.1 Similarity Measures

With the tag assignment relation Y as the central connecting structure, a folksonomy can be browsed along all contained dimensions: E.g, given a user u, one can browse through u's tags and resources, or given a tag t, one can see resources tagged with t as well as co-occuring or "related" tags. This allows for serendipitous discovery of interesting content and is seen as a major strength of folksonomies [41]. Naturally, this kind of browsing is limited to *explicit* edges in the folksonomy graph. If no direct link exists e.g. between two tags t_1 and t_2, it will be quite hard for a user interested in t_1 to discover t_2, even if both tags have a very similar *semantic* meaning.

So an interesting research question is if *implicit* links between folksonomy objects can be inferred which facilitate browsing in a 'semantic direction' instead of being restricted to explicit links. Central to this task of harvesting *emergent semantics* in folksonomies [22] are the concepts of similarity and relatedness. Budanitsky and Hirst pointed out that similarity can be considered as a special case of relatedness [7]. In recent work [10], we focussed on relatedness of tags, because they carry the semantic information within a folksonomy, and provide thus the link to ontologies. Additionally, this focus allows for an evaluation with well-established measures of similarity in existing lexical databases. We analyzed the following five measures of tag relatedness:

Co-Occurrence: Given a folksonomy (U,T,R,Y), we define the *tag-tag co-occurrence graph* as a weighted undirected graph whose set of vertices is the set T of tags. Two tags t_1 and t_2 are connected by an edge, iff there is at least one post (u,T_{ur},r) with $t_1,t_2 \in T_{ur}$. The *weight* of this edge is given by the number of posts that contain both t_1 and t_2, i. e., $w(t_1,t_2) := |\{(u,r) \in U \times R \mid t_1,t_2 \in T_{ur}\}|$.
Co-occurrence relatedness between tags is given directly by the edge weights. For a given tag $t \in T$, the tags that are most related to it are thus all the tags $t' \in T$ with $t' \neq t$ such that $w(t,t')$ is maximal.

Tag Context Similarity: The tag context similarity (TagCont) is computed in the vector space $\mathbb{R}^{|T^*|}$, whereby T^* is the set of most popular (i. e., most often used) tags in the folksonomy. We chose the top $10,000$ tags here. For a tag t, the entries of the vector $\mathbf{v}_t \in \mathbb{R}^{|T^*|}$ are defined by $v_{tt'} := w(t,t')$ for $t \neq t' \in T, t' \in T^*$, where w is the co-occurrence weight defined above, and $v_{tt} = 0$. The reason for giving weight zero between a node and itself is that we want two tags to be considered related when they occur in a similar context, and not when they occur together.

Resource Context Similarity: The resource context similarity (ResCont) is computed in the vector space $\mathbb{R}^{|R|}$. For a tag t, the vector $\mathbf{v}_t \in \mathbb{R}^{|R|}$ is constructed by counting how often a tag t is used to annotate a certain resource $r \in R$: $v_{tr} := |\{u \in U \mid (u,t,r) \in Y\}|$.

User Context Similarity: The user context similarity (UserCont) is built similarly to ResCont, by swapping the roles of the sets R and U: For a tag t, the vector $\mathbf{v}_t \in \mathbb{R}^{|U|}$ is defined as $v_{tu} := |\{r \in R \mid (u,t,r) \in Y\}|$.

FolkRank: We use the FolkRank algorithm described in Section 4.1.3 to compute a ranked list of relevant tags for a given tag by modifying the preference vector.

In all three context representations (i. e., TagCont, ResCont, UserCont), we measure vector similarity by using the cosine measure, as is customary in Information Retrieval [44]: If two tags t_1 and t_2 are represented by $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^{|X|}$, their cosine similarity is defined as: $\text{cossim}(t_1,t_2) := \cos \measuredangle(\mathbf{v}_1,\mathbf{v}_2) = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{||\mathbf{v}_1||_2 \cdot ||\mathbf{v}_2||_2}$.

Dataset and Semantic Grounding Our analysis was based on a snapshot of BibSonomy from January 2010. Initially, it contained 2.120.322 tag assignments (TAS). From that, we kept only those TAS which contained one of the 10.000 most popular tags, leading to a total of 4.990 users, 432.164 resources, 10.000 tags and 1.619.210 tag assignments. To provide a semantic grounding of our folksonomy-based measures, we mapped the tags of BibSonomy to synsets of WordNet and used the semantic relations of WordNet to infer corresponding semantic relations in the folksonomy. In WordNet, we measured the similarity by using both the taxonomic path length and a similarity measure by Jiang and Conrath [26] that has been validated through user studies and applications [7].

A first assessment of the measures of relatedness can be carried out by measuring – in WordNet – the average semantic distance between a tag and the corresponding most closely related tag according to each one of the relatedness measures we consider. Figure 6 reports the average semantic distance between the original tag and the most related one, computed in

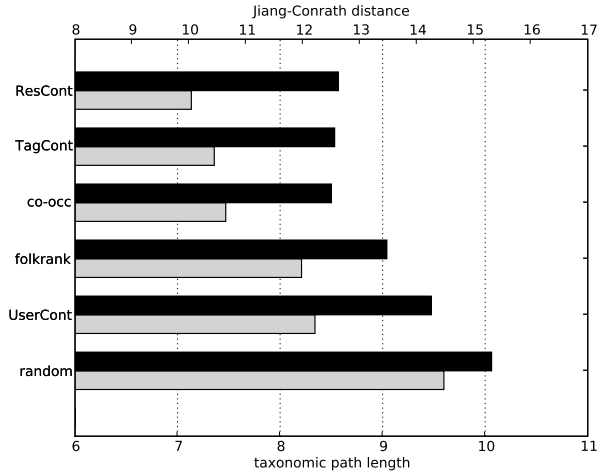


Figure 6 Average semantic distance, measured in WordNet, from the original tag to the most closely related one. The distance is reported for each of the measures of tag similarity discussed in the main text (labels on the left). Grey bars (bottom) show the taxonomic path length in WordNet. Black bars (top) show the Jiang-Conrath measure of semantic distance.

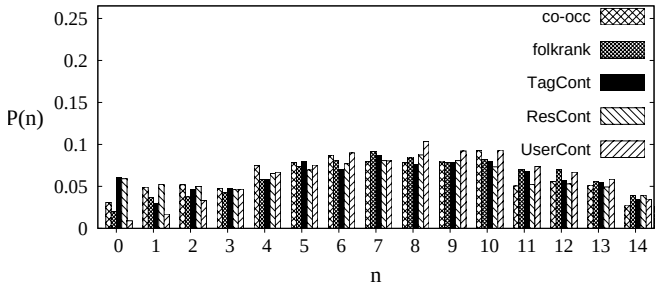


Figure 7 Probability distribution for the lengths of the shortest path leading from the original tag to the most closely related one. Path lengths are computed using the subsumption hierarchy in WordNet.

WordNet by using both the (edge) shortest-path length and the Jiang-Conrath distance. The tag and resource context relatedness (and, regarding Jiang-Conrath distance, also co-occurrence) point to tags that are semantically closer according to the semantic grounding. Remarkably, the notion of similarity by tag context (*TagCont*) has an almost optimal performance. This is interesting because it is computationally less intensive than the similarity by resource context, as it involves tag co-occurrence with a fixed number (10,000) of popular tags, only.

As a next step we focussed on the shortest paths (measured by the taxonomic pathlength) in WordNet that lead from an initial tag to its most closely related tag (according to the different measures of relatedness). Figure 7 displays the normalized distribution $P(n)$ of shortest-path lengths n (number of edges) connecting a tag to its closest related tag in WordNet. The similarities by tag context and resource context display a strong peak at $n = 0$. Tag context similarity also displays a weaker peak at $n = 2$ and a comparatively depleted number of paths with $n = 1$. For higher values of n, the histogram for resource context and tag context has the same shape as the others. While the peak at $n = 0$ is due to the detection of actual synonyms in WordNet, the higher value at $n = 2$ (paths with two edges in WordNet) for tag context may be compatible with the sibling relation. This assumption was confirmed by inspecting the path composition of these two-edge paths: In more than 80% of the cases, they consisted of a '1-up-1-down' hop for the tag context relatedness. We omit a more detailed discussion at this point for space reasons and refer the reader to [10]. In that paper, we have performed the same experiment on a much larger dataset, a snapshot of the bookmarking system Delicious, with remarkably similar findings.

Conclusions Strikingly, our analysis shows that the behavior of the most accurate measure of similarity (in terms of semantic distance of the indicated tags) can be matched by a computationally lighter measure (tag context similarity) which only uses co-occurrence with the popular tags of the folksonomy. The second contribution addresses the question of emergent semantics: Our results indicate clearly that, given an appropriate measure, globally meaningful tag relations can be harvested from an aggregated and uncontrolled folksonomy vocabulary. Specifically, we showed that the measures based on tag and resource context are capable to discover *implicit* links in the folksonomy graph, which facilitate a 'semantic' direction of browsing.

4.1.2 Spam Detection

Web spam detection is a well known challenge for search engines. Spammers add specific information to their web sites with the purpose to increase the ranking and not the quality of a page. They thereby boost the traffic to their web sites be it for commercial or political interests or to disrupt the service provided. The rise of social bookmarking systems made those systems also attractive to web spam: spammers (mis)use the popularity and high PageRank for their own purposes. Following [19], we consider spam in folksonomies as (1) content which legitimate users do not wish to share and (2) content which is tagged in a way to mislead other users (i. e., spammers add keywords that do not match the content of the bookmarks). The following summary of our findings from [33] is the basis of the spam detection framework described in Section 4.2.2.

Since complicating access to the system, e. g., by using captchas, does not repel the majority of spammers, techniques need to be developed which prevent spammers from publishing in these systems or hide their posts from serious users. Spam detection is a binary classification task. Based on different features describing known users a model is built to classify unknown users either as 'spammer' or 'non-spammer'.[11]

Dataset We generated a dataset from the BibSonomy database comprising 20,092 users, 306,993 tags, 920,176 resources and profile information of all BibSonomy users until the end of 2007. Considering only bookmarks, the system consists of 1,411

[11] Another task would be to classify *posts*.

Table 1 Baseline with all tags as features.

setting	TP	FP	FN	TN
frequency	466	0	2,324	100
TF/IDF	530	0	2,260	99

legitimate users and 18,681 users who were flagged as spammers by the system administrators. The posts of flagged users are no longer visible for other users. The evaluators did not follow official guidelines, but acted upon a common sense of what distinguishes users from spammers. Normally, characteristics of a user's profile (e. g., name, e-mail address), the composition of posts (e. g., the semantics of tags, the number of tags) and the content of the bookmarked web sites served as the basis for the administrator's decision.

Evaluation Setting As the experiments aim at reliably classifying new users, we split the instances chronologically: the training set (17,202 users, 282,473 tags, 774,678 resources, 7,904,735 TAS) comprehends all instances until 11/30/2007, the test set (2,890 users, 49,644 tags, 153,512 resources, 804,682 TAS) all instances of December 2007.

The results of the classification are presented in terms of true/false positives/negatives with TP being the number of spam instances that were correctly classified, FP the number of non-spam instances that were incorrectly classified as spam, FN the number of instances that were incorrectly classified as non-spam, and TN the number of instances that were correctly classified as non-spam. For our evaluation, we consider the F1-measure (F1M) which is the harmonic mean of precision and recall $F1M = \frac{2PR}{P+R}$ and the area under a ROC curve (AUC) which assesses the portion of the area of the unit square under the *receiver operating characteristics (ROC) curve*. These curves show the relative tradeoffs between benefits (true positives rates) and costs (false positives rates). The curves are plotted according to a pre-determined order of the test instances – for instance, the Naive Bayes classifier provides an instance probability which can be used for such a ranking.

Baseline As baseline we consider the tags used to describe a resource as features and use a classifier that has been shown to deliver good results for text classification such as Naive Bayes. Each user u can then be represented as a vector $\mathbf{u}$ where each dimension corresponds to a unique tag t. We consider two different settings: (a) the weight corresponds to the absolute frequency the tag t occurs with the user u, (b) each tag's weight is normalized using TF/IDF. Table 1 shows the TP, FP, FN, TN values for both approaches. The TF/IDF weighted baseline slightly identifies more spammers. The AUC for the frequency baseline is 0.801, the F1-measure 0.286. The AUC for the tfidf baseline is 0.794, the F1-measure 0.319.

Features Overall, we considered 25 features in four different groups (cf. Table 2). Each user is represented as a vector with an entry for each feature. All values are normalized over the set of users.

Table 2 The features of the different feature groups.

feature	description
	profile
namedigit	name contains digits
namelen	length of name
maildigit	email address contains digits
maillen	length of mail address
realnamelen	length of realname
realnamedigit	realname contains digits
realname2	two realnames
realname3	three realnames
	location based
domaincount	number of users in the same domain
tldcount	number of users in the same top level domain
spamip	number of spam user with this IP
	activity based
datediff	difference between registration and first post
tasperpost	number of tags per post
tascount	number of total tags added to all posts of this account
	semantic
co(no)spamr	user co-occurrences (related to resources) with (non) spammers
co(no)spamt	user co-occurrences (related to tags) with (non) spammers
co(no)spamtr	user co-occurrences (related to tag-resources pairs) with (non) spammers
spamratio(r/t/rt)	ratios of spam/non spam co-occurrences
grouptag	number of times the tag '$group=public' was used (entries with this tag have most likely been automatically generated by a specific software)
spamtag	ratio of spam tags to all tags of a user

The *profile features* are extracted from a user's registration data. For example, spammers often select names or e-mail addresses with many numbers. *Location based features* represent a user's location and domain. Often, the same spammer publishes the same content using several accounts with the same IP address. Thus, if one user with a specific IP (domain) is already marked as a spammer, the probability that other users with the same IP are also spammers is higher. *Activity properties* consider different kinds of user interactions with the system. For instance, while normal users tend to post their first bookmark soon after their registration, spam users often wait a certain time. *Semantic features* relate to the usage and content of the tags which serve as an annotation for a bookmark. For example, if users often select tags which are listed on a blacklist (created by the administrators), those users are likely to post spam. Further, co-occurrence features describe the usage of a similar vocabulary and resource usage.

Experiments We selected four different classification methods: Naive Bayes, C4.5 Decision Tree (using the J48 implementation), Logistic Regression, and Support Vector Machines (SVM). For the first three algorithms, we used the Weka implementation [52], for the SVM we used the LIBSVM package [11].

Table 3 shows the AUC, F1-measure, and the absolute FP and FN values for all algorithms, based on all features. Figure 8 depicts the ROC curves.[12] The best classifier with an AUC of

[12] We only included one baseline (tfidf) to reduce the number of curves.

Table 3 Evaluation values all features

Classifier	AUC	F1M (spammer)	FP	FN
Naive Bayes	0.906	0.876	14	603
SVM	0.936	0.986	53	23
Logistic Regression	0.918	0.968	30	144
J48	0.692	0.749	11	1112

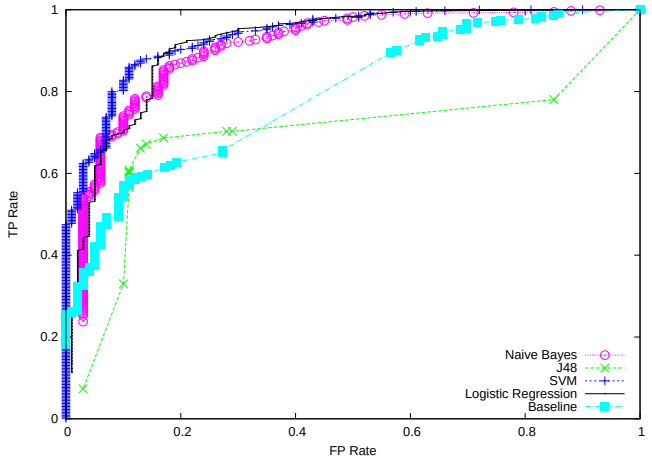


Figure 8 ROC curves of classifiers considering all features. The steepest progression shows the SVM classifier.

Table 4 Evaluation values of feature groups

features	AUC	F1M spammer	F1M nonspammer
profile	0.753	0.982	0
location	0.729	0.626	0.096
activity	0.752	0.982	0
spamtag/grouptag	0.770	0.982	0
prof+loc+act+sem	0.891	0.932	0.314
co-occurrences	0.927	0.985	0.497

Table 5 Evaluation with a cost sensitive classifier

classifier	AUC	F1M spammer	F1M nonspammer
SVM	0.936	0.924	0.299
J48	0.835	0.794	0.157
Logistic Regression	0.932	0.927	0.317
Naive Bayes	0.905	0.855	0.200

0.936 is the SVM, followed by Logistic Regression. Though the progression of the SVM's ROC shows that the false positive instances are the ones with less probability, 53 out of 100 non-spammers are ranked as spammers. The AUC's of the two baselines (0.801 and 0.794) yield lower results.

In order to find out about the contribution of the different features, we analyzed each feature group separately (cf. Table 4). We used logistic regression as this method resulted in the best AUC values for the group features. The semantic features were split in two subgroups – cooccurrence features (and the ratios) and the spamtag/grouptag.

Overall, none of the feature groups reaches the classification performance obtained when combining the features. This shows that a variation of different kinds of information is helpful. The co-occurrence features, as a subset of the semantic features, are most promising.

In order to penalize the wrong classification of non-spammers, we introduced *cost sensitive learning* [52]. We experimented with different cost options and found that a penalty of ten times higher than the neutral value delivered good results for the SVM. We also recalculated the other classifiers using cost options. Table 5 shows the changed F1M and false positive rates of classification using all features. Cost-sensitive learning on all features with logistic regression returns the best results.

4.1.3 Ranking

The web search algorithm PageRank [6] reflects the idea that a web page is important if there are many pages linking to it, and if those pages are important themselves.[13] In [23], we employed the same underlying principle for Google-like search and ranking in folksonomies. The key idea of our FolkRank algorithm is that a resource which is tagged with important tags by important users becomes important itself. The same holds, symmetrically, for tags and users. We have thus a graph of vertices which are mutually reinforcing each other by spreading their weights. In this section we briefly recall the principles of the FolkRank algorithm and explain how elements can be ranked with respect to an arbitrary subset of elements in the folksonomy.

Because of the different nature of folksonomies compared to the web graph (undirected triadic hyperedges instead of directed binary edges), PageRank cannot be applied directly on folksonomies. In order to employ a weight-spreading ranking scheme on folksonomies, we overcome this problem in two steps. First, we transform the hypergraph into an undirected graph. Then, we apply a differential ranking approach that deals with the skewed structure of the network and the undirectedness of folksonomies, and which allows for topic-specific rankings.

Folksonomy-Adapted PageRank First, we convert the folksonomy $\mathbb{F} = (U, T, R, Y)$ into an *un*directed tri-partite graph $G_{\mathbb{F}} = (V, E)$. The set V of nodes of the graph consists of the disjoint union of the sets of tags, users and resources (i. e., $V = U \dot\cup T \dot\cup R$). All co-occurrences of tags and users, users and resources, tags and resources become edges between the respective nodes, i. e., each triple (u,t,r) in Y gives rise to the three undirected edges $\{u,t\}$, $\{u,r\}$, and $\{t,r\}$ in E.

Like PageRank, we employ the random surfer model, that is based on the idea that an idealized random web surfer normally follows links (e. g., from a resource page to a tag or a user page), but from time to time jumps to a new node without following a link. The rank of the vertices of the graph is computed with the weight spreading computation

$$\mathbf{w}_{t+1} \leftarrow dA^{\mathrm{T}}\mathbf{w}_t + (1-d)\mathbf{p} \ , \qquad (1)$$

[13] This idea was extended in a similar fashion to bipartite subgraphs of the web in HITS [31] and to n-ary directed graphs in [53].

where $\mathbf{w}$ is a weight vector with one entry for each node in V, A is the row-stochastic version of the adjacency matrix[14] of the graph $G_{\mathbb{F}}$ defined above, $\mathbf{p}$ is the random surfer vector – which we use as preference vector in our setting, and $d \in [0,1]$ is determining the strength of the influence of $\mathbf{p}$. By normalization of the vector $\mathbf{p}$, we enforce the equality $||\mathbf{w}||_1 = ||\mathbf{p}||_1$. This[15] ensures that the weight in the system will remain constant. The rank of each node is its value in the limit $\mathbf{w} := \lim_{t\to\infty} \mathbf{w}_t$ of the iteration process. For a global ranking, one will choose $\mathbf{p} = \mathbf{1}$, i. e., the vector composed by 1's. For a topic-specific ranking as described in [17], however, we give higher weight to the node(s) we are interested in.

As the graph $G_{\mathbb{F}}$ is undirected, most of the weight that went through an edge at moment t will flow back at $t+1$. The results are thus rather similar (but not identical, due to the random surfer) to a ranking that is simply based on edge degrees. In [23] we observed that the topic-specific rankings are biased by the global graph structure. As a consequence, we developed in [23] the following differential approach.

FolkRank – Topic-Specific Ranking The undirectedness of the graph $G_{\mathbb{F}}$ makes it very difficult for other nodes than those with high edge degree to become highly ranked, no matter what the preference vector is. This problem is solved by the *differential* approach in FolkRank, which computes a topic-specific ranking of the elements in a folksonomy. A "topic" can be hereby basically an arbitrary subset of items in the folksonomy. As an example, when computing tag recommendations, the topic is determined by the user/resource pair (u,r) under consideration. Another possibility is to equate topics with tags; in this case, one can compute a ranking of all folksonomy items relative to a given tag. We will stick here to the recommendation scenario, as we will come back to it in Section 5.1.1.

In order to generate recommendations, however, $\mathbf{p}$ can be tuned by giving a higher weight to the user node and to the resource node for which one currently wants to generate a recommendation. The recommendation $\tilde{T}(u,r)$ is then the set of the top n nodes in the ranking, restricted to tags.

1. Let $\mathbf{w}^{(0)}$ be the fixed point from Equation (1) with $d = 1$.
2. Let $\mathbf{w}^{(1)}$ be the fixed point from Equation (1) with $\mathbf{p}[u] = 0.5$, $\mathbf{p}[r] = 0.5$ and $\mathbf{p} = \mathbf{0}$ else; and $d < 1$.
3. $\mathbf{w} := \mathbf{w}^{(1)} - \mathbf{w}^{(0)}$ is the final weight vector.

Thus, we compute the winners and losers of the mutual reinforcement of nodes when a user/resource pair is given, compared to the baseline without a preference vector. We call the resulting weight $\mathbf{w}[x]$ of an element x of the folksonomy the *FolkRank* of x.

In [23] we showed that $\mathbf{w}$ provides indeed valuable results on a large-scale real-world dataset while $\mathbf{w}^{(1)}$ provides an unstructured mix of topic-relevant elements with elements having high edge degree. In [24], we applied this approach for detecting trends over time in folksonomies. In Section 5.1.1 we present FolkRank as a method to compute tag recommendations – an example for a tag-specific FolkRank computation is described in the implementation Section 4.2.3.

[14] $a_{ij} := \frac{1}{\text{degree}(i)}$ if $\{i,j\} \in E$ and 0 else

[15] ...together with the condition that there are no rank sinks – which holds trivially in the undirected graph $G_{\mathbb{F}}$.

Figure 9 Example of a semantic browsing facility by displaying semantically 'similar' tags. The example here is the tag page for *python*; one can see that the 'similar' tags (upper arrow) are substantially different from the 'related' tags (lower arrow), yielding other programming languages like *perl* or *c++*.

4.2 Implementation Issues

When browsing a folksonomy, users can directly experience its small-world properties, as they will discover new and interesting content in a serendipitous way. From an implementation point of view, these properties are inherent to a folksonomy system and do not require further special considerations or design choices on the developers side. This is why we now focus on the implementation of the similarity measures, the spam classification framework and the FolkRank ranking, which exemplify the knowledge transfer from our research into the running system.

4.2.1 Similarity Measures

As stated in the previous research section, our primary goal when investigating similarity measures among tags was to facilitate a 'semantic' direction of browsing along *implicit* edges in the folksonomy graph. As a result of our research, it turned out that cosine similarity in the vector space of the 10,000 most popular tags yields semantically close tags at a comparatively low cost of complexity. However, due to the size of our system (and the size of folksonomy systems in general), even this computationally lightweight measure is not feasible for online computation of similar tags. In order to make available our findings to our users, we hence decided to implement a daily update mechanism which executes the following steps:

1. Extract the 10,000 most popular tags of the system.
2. Extract the candidate tag set T_{cand} by including tags used at least 10 times within the system.
3. Build a vector representation for each tag $t \in T_{cand}$ in the vector space of the top 10,000 tags.
4. Compute pairwise cosine similarity among all tags $t \in T_{cand}$.
5. Store the 10 most similar tags for each tag $t \in T_{cand}$.

As of May 2010, this procedure involves approximately 21.000 tags present in the candidate tag set. It takes roughly 90 minutes (using approximately 700 MB of RAM) on a machine equipped with two six-core 2.4Ghz AMD Opteron processors. This computation time is reasonable for now, but we are experimenting with more efficient methods for all-pairs similarity search like in [2] to be prepared for growing amounts of data.

Now, instead of computing similar tags online (and possibly delaying the delivery of the rendered HTML page) we can simply perform lookups in the database. When a user is browsing on a tag page for a given tag in our system (e. g., http://www.bibsonomy.org/tag/python), we provide an additional kind of browsing facility by displaying 'similar tags' (see Figure 9). From this example one can see that this does in fact introduce additional 'implicit' connections: While the 'related' tags (representing the explicit co-occurrence links) do mainly contain rather general or broadly related tags like *web*, *programming* or *software*, the 'similar' tags comprise almost exclusively other programming languages like *python* or *c++*. This allows for a previously hidden direction of browsing.

4.2.2 Spam Detection

With growing popularity, BibSonomy started to attract more and more spammers (i. e., users who abuse the system, e. g., for advertising purposes). While manual classification was feasible in the initial phase, the issue of a scalable automatic spam classification framework became more and more pressing. A main reason for this is that the starting page of BibSonomy displays the most recent posts of *all* BibSonomy users. When spammers post inappropriate content, this is hence directly visible when visiting our homepage. We think that this is a critical factor for the uptake of our system in possible new user communities.

We included the insights of our research in the design of this framework. In addition, the implementation was guided by the following requirements:

Quick classification: A typical usage pattern of a spammer is that he registers at BibSonomy and directly starts to post malicious content. Hence, it is important that the spammer classification takes place quickly after registration or by the latest after the first posting activity.

Data sparsity: When trying to classify a user quickly after registration, there is naturally not much data available; hence we needed robust classification algorithms able to deal with this data sparsity.

Concealment: When spam users realize that they have been classified as spammers, they will possibly perform counteractive measures or create even more accounts in order to conceal their purpose or identity. Our goal was to hide (as far as possible) the fact of being classified from the spammers; basically they can keep on using the system like normal users, but their content is only visible to themselves. As a side effect, this behaviour does not impede regular users that have been misclassified from using the system.

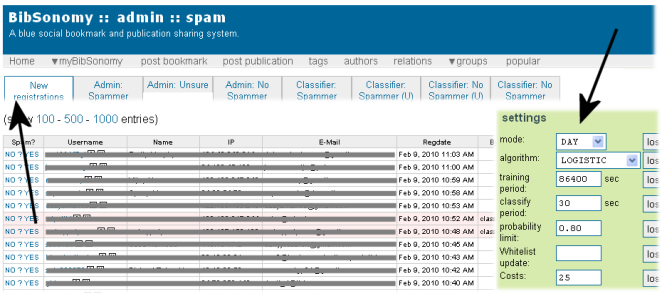


Figure 10 Administrator interface of the automatic spam classifcation framework (partial screenshot). There are several lists of new and classified users available (left arrow), and the classificator settings can be modified (right arrow).

Reversibility and manual intervention: One insight of our research was that none of the investigated algorithms had an optimal performance. Hence it is important to be able to manually intervene and reverse a classification in case of an error without any loss of data for the user.

Figure 10 shows the outcome as a screenshot of the administration interface of our spammer classification framework. The classification itself takes place on an external machine, which exchanges data with BibSonomy via our API. Regularly (currently each 30 seconds) the framework reports new users to the classification engine, which makes a graded decision if each new user is a spammer or not: The possible classes are *spammer, spammer (unsure), no spammer (unsure)* and *no spammer*. The assignment of a new user to one of these classes depends on the output of the chosen classifier and some configurable thresholds. We randomly keep on surveying manually the results in order to revert misclassifications and to fine-tune the parameters. In general, we are very satisfied with the overall performance and have the feeling that our framework leads to a remarkably better usability of BibSonomy both for our users and us as administrators.

4.2.3 Topic-specific Ranking by FolkRank

In Section 4.1.3, we introduced the FolkRank algorithm [23] for search and ranking in Folksonomies.It has been integrated into BibSonomy. The computation of FolkRank is done offline, due to performance reasons.

Usually, resources in our system are displayed ordered by date, showing up the latest posts on top. On the tag pages (reachable by the URL `/tag/`$t_1 \sqcup \ldots \sqcup t_n$), we provide a reordering of the resources based on FolkRank, relative to the chosen tag(s) $t_1, \ldots, t_n$. Since FolkRank computes a ranking for all three dimensions of a Folksonomy – users, tags, and resources – BibSonomy also shows in the sidebar the ranked tags as 'related tags' and the users as 'related users', providing another 'semantic' navigation facility, which is not restricted to explicit edges in the folksonomy graph. An example of the top five bookmarks and publication references for the tag *folksonomy* can be seen in Figure 11, the corresponding related users and tags in Figure 12.

Figure 11 A screenshot of the top five bookmark and publication posts for the tag *folksonomy* sorted by FolkRank. The screenshot shows a part of the page http://www.bibsonomy.org/tag/folksonomy?order=folkrank on June 8th, 2009.

To allow for efficient retrieval of the FolkRank ordered posts, we regularly pre-compute for each tag $t \in T$ the ranking of all resources, users, and tags and store the top 100 elements in each dimension in the database. Retrieval for queries with only one tag (i. e., `/tag/`t) then simply returns those rows for the tag t. However, due to the combinatorial explosion, the offline computation for queries with more than one tag, i. e., `/tag/`$t_1 \sqcup \ldots \sqcup t_n$ is not possible. Thus, we simply merge the rankings for the tags $t_1, \ldots, t_n$ using the function

Figure 12 A screenshot of the top twenty related users and tags for the tag *folksonomy* sorted by FolkRank. The screenshot shows the sidebar of the page http://www.bibsonomy.org/tag/folksonomy?order=folkrank on June 8th, 2009.

$\mathbf{w}[r](t_1, \ldots, t_n) := \sum_{i=1}^{n} \mathbf{w}[r](t_i)$ which sums the FolkRank weights for resource $r \in R$ – using the tags t_i as preference – as defined in Section 4.1.3.

5 Entering Content

Social resource sharing systems like BibSonomy allow users to archive, organize, search and share resources with other users, thereby enabling and exploiting collaborative interactions. Completely relying on user generated content, the process of entering data has to be kept as simple as possible. Furthermore, the user should be encouraged to annotate resources extensively. Accordingly, a crucial aspect is how to minimize the users' participation effort while maximizing the quality of the resulting metadata.

BibSonomy incorporates techniques for automatically extracting information from web pages, thus allowing the user to import resource metadata from the World Wide Web by a simple mouse click – without the need for tedious manual typing. These so-called *scraping* techniques are described in Section 5.2.2.

Beside providing a resource's meta information, the user has to annotate the resource with a set of meaningful words, so-called *tags*, describing the resource (from the user's point of view). Allowing the user to freely choose arbitrary words was a key to the success of these sharing systems, but also led to heterogeneous vocabulary among different users. For increasing tag homogeneity and simultaneously lowering the effort needed for annotating a resource in BibSonomy, a list of tag suggestions where a tag can be chosen from by a simple mouse click is presented while posting a resource. In Section 5.2.1. we evaluate different methods for *tag recommendations* in collaborative tagging systems.

Probably the easiest way for a user to generate metadata is to copy an existing post from another user in the system – this also happens extensively in BibSonomy. The resulting *copy network* exhibits an interesting structure, where possible *communities of interest* become visible. We present a first analysis of this network in Section 5.1.2.

5.1 Research Questions

5.1.1 Tag Recommendations

To support users in the tagging process and to expose different facets of a resource, most collaborative tagging systems offer some kind of tag recommendations. Recommending tags can serve various purposes, such as: increasing the chances of getting a resource annotated, reminding a user what a resource is about and consolidating the vocabulary across the users. Recommender systems can use the content of resources, the context of the user, or other sources of data as basis for their recommendation. We here focus on tag recommendation methods that require only the folksonomy data of collaborative tagging

systems. This section is based on work published in [28] and [29].

Problem Definition The task of a tag recommender system is to recommend, for a given user $u \in U$ and a given resource $r \in R$ with $T(u,r) = \emptyset$, a set $\tilde{T}(u,r) \subseteq T$ of tags.[16] In many cases, $\tilde{T}(u,r)$ is computed by first generating a ranking on the set of tags according to some quality or relevance criterion, from which then the top n elements are selected.

Notice that the notion of tag relevance in social bookmarking systems can assume different perspectives, i. e., a tag can be judged relevant to a given resource according to the society point of view, through the opinion of experts in the domain or based on the personal profile of an individual user. For all the evaluated algorithms, we focus here on measuring the individual notion of tag relevance, i. e., the degree of likeliness of a user for a certain set of tags, given a new or untagged resource.

Collaborative Filtering Due to its simplicity and promising results, Collaborative Filtering (CF) has been one of the most dominant methods used in recommender systems. The main idea of Collaborative Filtering is to predict the utility of a certain tag based on the opinion of like-minded users [45, 5, 43]. Therefore, one represents each user as a vector and computes the k-nearest neighbors of a user using a vector similarity measure (e. g., cosine measure). Then, for a given resource, the tags are ranked by decreasing frequency of occurrence in the ratings of the neighbors, weighted by the similarity of each user.

Because of the ternary relational nature of folksonomies, traditional CF cannot be applied directly, unless we reduce the ternary relation Y to a lower dimensional space [36]. To this end we consider as user-similarity matrix $\mathbf{X}$ alternatively the two 2-dimensional projections $\pi_{UR}Y \in \{0,1\}^{|U|\times|R|}$ with $(\pi_{UR}Y)_{u,r} := 1$ if there exists $t \in T$ s. t. $(u,t,r) \in Y$ and 0 else, and $\pi_{UT}Y \in \{0,1\}^{|U|\times|T|}$ with $(\pi_{UT}Y)_{u,t} := 1$ if there exists $r \in R$ s. t. $(u,t,r) \in Y$ and 0 else.

Having defined matrix $\mathbf{X}$, and having decided whether to use $\pi_{UR}Y$ (CF-UR) or $\pi_{UT}Y$ (CF-UT) for computing user neighborhoods, we have the required setup to apply Collaborative Filtering. For determining, for a given user u, a given resource r, and some $n \in \mathbb{N}$, the set $\tilde{T}(u,r)$ of n recommended tags, we compute first N_u^k as described above, followed by:

$$\tilde{T}(u,r) := \operatorname*{argmax}_{t \in T}^{n} \sum_{v \in N_u^k} \mathrm{sim}(\mathbf{x}_u, \mathbf{x}_v)\delta(v,t,r) \qquad (2)$$

where $\delta(v,t,r) := 1$ if $(v,t,r) \in Y$ and 0 else. For our setting we found the best value for k to be 20 for the UT variant and 30 for the UR variant.

FolkRank For generating a tag recommendation for a given user/resource pair (u,r), we compute the ranking as described in Section 4.1.3 and then restrict the result set $\tilde{T}(u,r)$ to the top n tag nodes. For both the *adapted PageRank* and *FolkRank* we set $d = 0.7$ and stopped computation after 10 iterations. In $\mathbf{p}$, we gave higher weights to the user u and the resource r at hand: While each user, tag and resource got a preference weight of 1, u and r got a preference weight of $1+|U|$ and $1+|R|$, resp. We then normalized $\mathbf{p}$ by division by $|T|+2|U|+2|R|$.

Table 6 Characteristics of the datasets.

dataset	$\lvert U\rvert$	$\lvert T\rvert$	$\lvert R\rvert$	$\lvert Y\rvert$	$\lvert P\rvert$
raw dump	1,037	28,648	86,563	341,183	96,972
p-core at level 5	116	412	361	10,148	2,522

Most Popular Tags Here we introduce methods based on tag counts which are particulary cheap to compute and therefore are good candidates for online computation of recommendations. (1) Recommending the *most popular tags* of the folksonomy is the most simplistic approach. It recommends, for any user and any resource the same set of tags. (2) Tags that globally are most specific to the resource will be recommended when using the *most popular tags by resource*. (3) Since users might have specific interests for which they already tagged several resources, using the *most popular tags by user* is another option. (4) We can also mix the two former recommendations which results in the *most popular tags mix 1:1*. Therefore, we add the counts for each tag and then sort the tags by their count. (5) Another mix variant is the *most popular tags ρ–mix*, where the tag counts of the two participating sets are *normalized* and *weighted* with a factor ρ before they are added. For the evaluation we set $\rho = 0.6$.

Evaluation We created a complete snapshot of all users, resources (both publication references and bookmarks) and tags publicly available at April 30, 2007, 23:59:59 CEST. From the snapshot we excluded the posts from the DBLP computer science bibliography[17] since they are automatically inserted and all owned by one user and all tagged with the same tag (*dblp*). Therefore they do not provide meaningful information for the analysis. We disregarded if the tags had lower or upper case, since this is the behaviour when querying for posts tagged with a certain tag. Many recommendation algorithms suffer from sparse data and will thus produce bad recommendations on the 'long tail' of items which were used by only few users. We follow the conventional approach and restrict the evaluation to the 'dense' part of the folksonomy. To this end, we adapt the notion of a p-core [1] to tri-partite hypergraphs. The p-core of level k is a subset of the folksonomy with the property, that *each user, tag and resource has/occurs in at least k posts*. We chose $k = 5$ for our dataset. Table 6 shows the resulting sizes of the folksonomy.

To evaluate the recommenders we used a variant of the leave-one-out hold-out estimation [18] which we call *LeavePostOut*. In all datasets, we picked randomly, for each user u, one resource r_u, which he had posted before. The task of the recommenders was then to predict the tags the user assigned to r_u, based on the folksonomy (U,T,R,Y') with $Y' :=$

[16] We are using the notation introduced in Section 1.

[17] http://www.informatik.uni-trier.de/~ley/db/

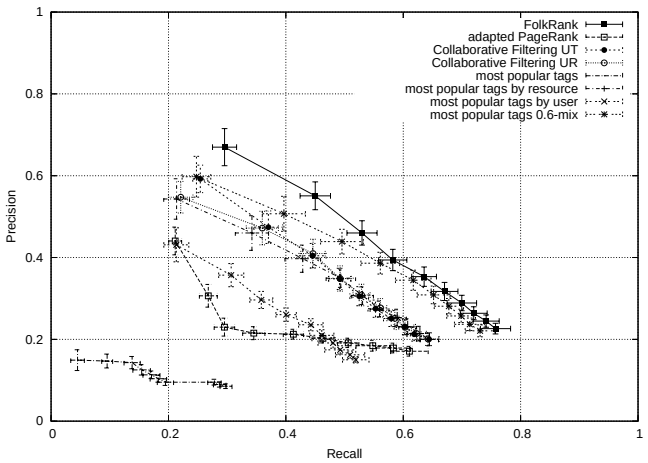


Figure 13 Recall and Precision for BibSonomy p-core at level 5.

$Y \setminus (\{u\} \times T(u, r_u) \times \{r_u\})$. As performance measures we used precision and recall which are standard in such scenarios [18]. We averaged precision and recall over all users:

$$\text{recall} = \frac{1}{|U|} \sum_{u \in U} \frac{|T(u, r_u) \cap \tilde{T}(u, r_u)|}{|T(u, r_u)|} \tag{3}$$

$$\text{precision} = \frac{1}{|U|} \sum_{u \in U} \frac{|T(u, r_u) \cap \tilde{T}(u, r_u)|}{|\tilde{T}(u, r_u)|} \tag{4}$$

This process was repeated ten times, each time with another resource per user.

Results In Figure 13 we see the results of the evaluation in a precision-recall plot. A datapoint on a curve stands for the number of tags recommended (starting with the highest ranked tag on the left of the curve and ending with ten tags on the right). The results have a rather large standard deviation, as can be seen by the error bars. This is due to the fact that every run is averaged over 116 users only and thus the performance of the ten runs differs. *FolkRank* provides on average best precision and recall, followed by the *most popular tags* 0.6*–mix* recommender. Both *Collaborative Filtering* algorithms and *most popular tags by resource* show similar results for higher numbers of tags.

Conclusion The presented results show that the graph-based approach of FolkRank is able to provide tag recommendations which are significantly better than those of approaches based on tag counts and even better than those of state-of-the-art recommender systems like Collaborative Filtering. The tradeoff is, though, that computation of FolkRank recommendations is cost-intensive so that one might prefer less expensive methods to recommend tags in a social bookmarking system. The *most popular tags* ρ*–mix* approach has proven to be considered as a solution for this problem. It provides results which can almost reach the grade of FolkRank but which are extremely cheap to generate. Especially the possibility to use index structures (which databases of social bookmarking services typically provide anyway) makes this approach a good choice for online recommendations.

Figure 14 Users can copy posts by clicking on the 'copy' link or on the star next to the title.

5.1.2 Analysis of the 'copy' Network

Perhaps the most simple way to store a post in BibSonomy is by *copying* it from another user. This is possible by just clicking on the 'copy' link or on the 'star' next to each post (see Figure 14). One then only has to add some tags to the post and then can store it.

Besides the practical act of storing a post, copying a post from another user expresses to a certain extent interest in his posts and thus in the topics the user is interested in. Hence, analysing the copying behaviour might reveal connections between users which are the result of common interests. By looking at groups of users which are connected by the posts they have copied from each other, one could thus find communities of users with common interests. To further investigate this theory, we take a look at the *copy network* inside BibSonomy. There, a user a is connected to another user b, if a copied a post from b.

We extracted from BibSonomy's access log the requests which originated from copying a publication post for the period from October 13th, 2007 to July 27th, 2009. We disregarded copy requests from posts of the user 'dblp' and from spammers. The resulting 9,255 requests form a graph with 1,553 users as nodes, connected by 3,779 (directed) edges. We weighted the edges by the number of resources the user copied from the other user. To visualize this network, we removed edges with weight less than three. The largest component of the resulting graph is shown in Figure 15. It consists of 244 users, connected by 411 edges.

Probably the most noticeable active 'consumers', i. e., users which copy other users posts, are the users 'dnoack' and 'rinnegatomai' which both have connections to more than 35 other users. While 'rinnegatomai' copied more than three posts from 45 other users which are distributed all over the graph, 'dnoack' mostly copied posts from unconnected users. Furthermore, the strongly connected users in the top right also copied posts from 'dnoack'. Thus, this user is a kind of 'hub'. There are also users whose posts have been copied by many other users. Besides the users in the strongly connected component in the top right, these are the users 'hotho', 'stumme', and 'kochm' which all have more than ten incoming connections. Thus, they are rather central and seem to posess interesting posts.

Finally, one is also able to identify various links between the users of our research group ('beate', 'dbenz', 'folke', 'hotho', 'jaeschke', 'schmitz', 'stumme'). This correctly suggests that

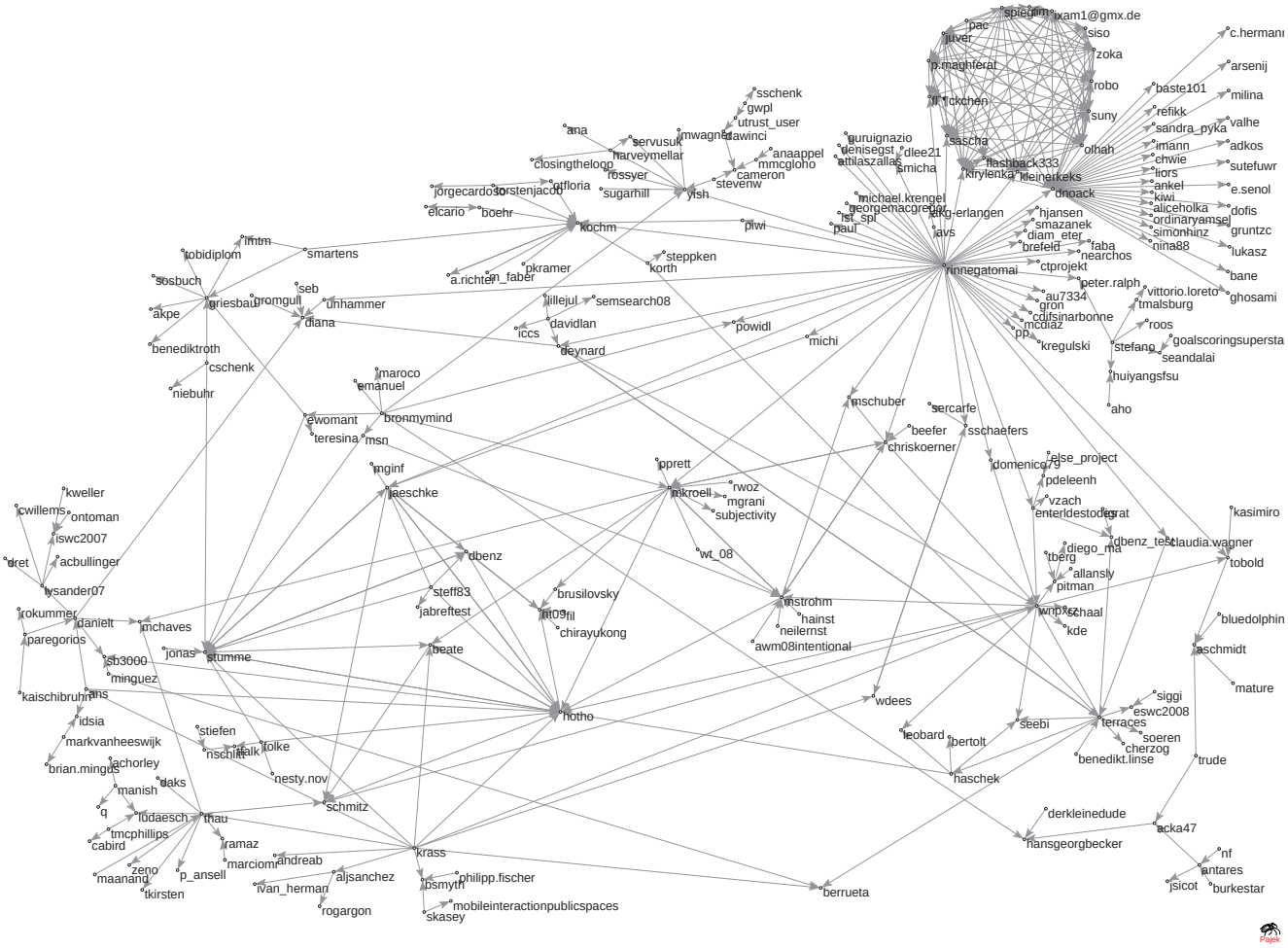


Figure 15 The copy network of BibSonomy publications. An arrow from user *a* to *b* indicates that *a* has copied at least three publications from *b*.

we are interested in similar topics and supports the idea that the copy network could be a good starting point to detect *communities of interest* in folksonomies.

5.2 Implementation Issues

5.2.1 Tag Recommendation Framework

The topic of tag recommendations in social bookmarking systems has attracted quite a lot of attention in the last years and, accordingly, different approaches for computing tag suggestions emerged. These algorithms are typically evaluated by computing some performance measure (e. g., *precision* and *recall*) in an 'off-line' setting, that is, by iterating over posts in a dataset, which was derived from a social bookmarking system, presenting only a user and a resource to the recommender system (splitting the data set for training and testing if necessary). Thus, for each post, the set of suggested tags can be compared with those the user had assigned.

These evaluation scenarios not only ignore some constraints of real live applications (e. g., response time, CPU usage and memory consumption), they also can't take into account the effect of presenting a set of recommended tags to the user. For evaluating these effects, a recommender system must be integrated into a real live application.

To facilitate such evaluations, we implemented a framework, designed for integration of arbitrary recommender systems and thus providing researchers a testbed to test and evaluate their methods in a live system. The framework was also the cornerstone of the 2009 ECML PKDD Discovery Challenge,[18] where one task required the participants to deliver live recommendations for BibSonomy. 15 recommenders, distributed over six countries (Canada, Greece, Spain, South-Korea, United Kingdom, and Germany) took part in the competition. This was a larger stress test for external recommenders and the framework itself – which it passed bravely.

Recommender Interface One central element of the framework is the recommender interface. It specifies which data is passed from a recommendation request to one of the implemented recommenders and how they shall return their result. Figure 16 shows the UML class diagram of the *TagRecommender* interface one must implement to deliver recommendations to Bib-

[18] http://www.kde.cs.uni-kassel.de/ws/dc09

Sonomy. We decided to keep the interface as simple as possible by requiring only three methods, building on BibSonomy's existing data model (see Section 3.1) and adding as few classes as possible (RecommendedTag, RecommendedTagComparator).

The *getRecommendedTags* method returns – given a post – a sorted set of tags; *addRecommendedTags* adds to a given (not necessarily empty) collection of tags further tags. Since – given a post and an empty collection – *addRecommendedTags* should return the same result as *getRecommendedTags*, the latter can be implemented by delegation to the former. Nonetheless, we decided to require both methods to cover the simple "give me some tags" case as well as more sophisticated usage scenarios (think of 'intelligent' collection implementations, or a recommender which improves given recommendations). The post given to both methods contains data like URL, title, description, date, user name, etc. that will later be stored in the database and that the recommender can use to produce good recommendations. It might also contain tags, i. e., when the user edits an existing post or when he has already entered some tags and requests new recommendations. Implementations could use those tags to suggest different tags or to improve their recommendation. With the *setFeedback* method the final post as it is stored in the database is given to the recommender such that it can measure and potentially improve its performance. Finally, the *getInfo* method allows the programmer to provide some information describing the recommender. This can be used to better identify recommenders or be shown to the user.

Two further classes augment the interface: *RecommendedTag* basically extends the data model's *Tag* class by adding floating point *score* and *confidence* attributes. A corresponding *RecommendedTagComparator* can be used to compare tags, e. g., for sorted sets. It first checks textual equality of tags (ignoring case) and then sorts them by score and confidence. Consequently, tags with equal names are regarded as equal.

Implementations are not restricted to Java: Using a remote recommender one can implement a recommender in any language that is then integrated using XML over HTTP requests.

Multiplexing Tag Recommender Our framework's technical core component is the so called *multiplexing tag recommender* Implementing BibSonomy's tag recommender interface, it provides the web application with tag recommendations, using one of the recommenders available. All recommendation requests and each recommender's corresponding result are logged in a database. For this purpose, every tag recommender is registered during startup and assigned to a unique identifier. For technical reasons, we differentiate between locally installed and remote recommenders. Whenever the *getRecommendedTags* method is invoked, the corresponding recommendation request is delegated to each recommender, spawning separate threads for each recommender. After a given timeout period (e. g., 1000 ms), one of the collected recommendations is selected. The evaluation of the algorithms submitted to the ECML PKDD Discovery Challenge 2009 showed that the timeout was indeed a hard requirement for some of the recommenders. The selection was done following a preconfigured *selection strategy*: In our current setting we implemented a "*sampling without replacement*" strategy, choosing exactly one recommender i and all of its recommended tags $T_i(u,r)$ When each recommender was chosen exactly once, this process restarts.

5.2.2 Importing Publication References from Digital Libraries

BibSonomy contains more than 60 so called "scrapers" which automatically extract publication metadata from digital libraries like SpringerLink[19] or IEEE Xplore.[20] The scrapers are an important part of BibSonomy because most publications (and their metadata) can nowadays be found online in digital libraries. When visiting such a site, users can post the publication metadata to BibSonomy with a one-button click in their web browser.[21] The scrapers extract all available information and transform it into BibTeX format (if necessary). The complete list of supported digital libraries can be found at http://www.bibsonomy.org/scraperinfo; we also provide a web service[22] which only extracts the metadata from a given URL and returns it in BibTeX or RDF format. Finally, the source code of all scrapers is available online in a Maven repository.[23]

Typically, literature references can be imported only from services known by BibSonomy (i. e., supported by a scraper) or in well-defined formats like BibTeX. This is a strong restriction, since many literature references in the web are neither available in BibTeX format nor does BibSonomy offer an appropriate scraper. Those references rather appear in the form of human readable publication lists as shown in Figure 17. Hence, our efforts to enhance import focused on techniques to allow for the import of such resources. We integrated the MALLET [38] system in BibSonomy which uses information extraction techniques [40] to fill BibTeX fields like *title*, *author*, or *year* from such references. Of course, this machine learning approach does not work perfect, but in most cases it eases the transfer of the information from such proprietary lists into the BibSonomy post form. We also ensured to save the original text besides the user corrected metadata in the database to use it for training the algorithm.

6 Accessing the Content

In the last sections, we mainly focused on how users interact with BibSonomy when browsing, searching or entering content into the system. We now shift perspective to the content itself. The first important aspect hereby is the design of the database back-end to store the large amounts of data while keeping response time to queries low. Then, we detail on the different

[19] http://www.springerlink.de/

[20] http://ieeexplore.ieee.org/

[21] The corresponding buttons are available at http://www.bibsonomy.org/buttons and can easily be added to the browser's toolbar.

[22] http://scraper.bibsonomy.org/

[23] http://dev.bibsonomy.org/

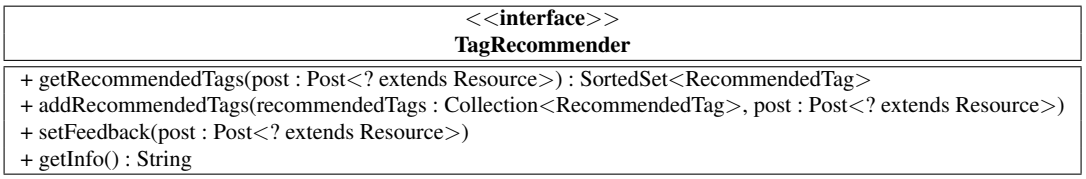
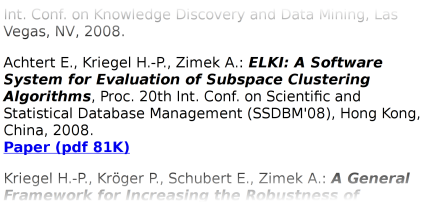


Figure 16 The UML class diagram of the tag recommender interface.


Int. Conf. on Knowledge Discovery and Data Mining, Las Vegas, NV, 2008.

- Achtert E., Kriegel H.-P., Zimek A.: ***ELKI: A Software System for Evaluation of Subspace Clustering Algorithms***, Proc. 20th Int. Conf. on Scientific and Statistical Database Management (SSDBM'08), Hong Kong, China, 2008.
  Paper (pdf 81K)
- Kriegel H.-P., Kröger P., Schubert E., Zimek A.: ***A General Framework for Increasing the Robustness of***

Figure 17 A typical literature reference as it can often be found on personal homepages of researchers. Using information extraction, the BibTeX fields *title* ("ELKI: A Software System for Evaluation of Subspace Clustering Algorithms"), *author* ("Achtert E., Kriegel H.-P., Zimek A."), *booktitle* ("Proc. 20th Int. Conf. on Scientific and Statistical Database Management (SSDBM ' 08)", and *year* ("2008") can be filled automatically.

facilities of BibSonomy to access the data, namely our web front-end with various export formats and the REST API.

6.1 Database Back-End

The core of each collaborative tagging system is a database which stores the folksonomy, answers queries to browse and search it, and allows the user to enter new or update existing data. In this section we look at the requirements on a database for this specific scenario, give an overview of our implementation, and perform an exemplary performance test of our solution.

6.1.1 Objectives and Requirements

The folksonomy use-case is a typical "few updates, many selects" setting where many users often issue select queries against the database and only few users issue insert, update, or delete queries. Thus, we want to optimize the majority case and get fast answers for typical folksonomy queries like "get all bookmarks of user 'nepomuk' with the tags 'fca' and 'folksonomy"', while sub-optimal insert statements are not an issue.

Another requirement is scalability, meaning an increase in available resources is accompanied with a comparable increase in performance. Practically, we need a system which is extensible and thereby benefiting from better or more hardware to cope with an increased number of requests in the future.

Furthermore, the database implementation should not be limited to handling of folksonomy data but rather be a standard solution with good support, extensibility and the option to exchange it by another, similar product. This also requires us to keep the number of dependencies to a specific implementation low.

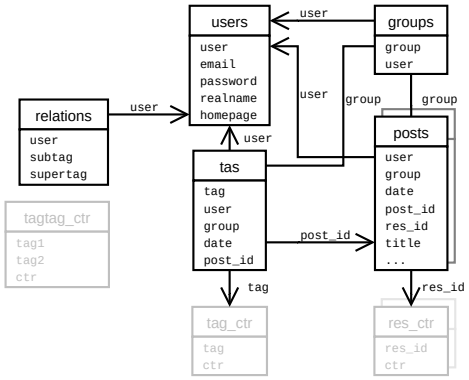


Figure 18 Relational schema of the most important tables.

6.1.2 Implementation

We decided to use a relational database management system (RDBMS) in favour of other approaches like object-relational mapping or RDF stores because of the greater adjustability. Using an RDBMS one has to a certain extent control on how the data is stored and how it is retrieved. In particular, we can optimize the schema and the queries. Furthermore, there are several standard solutions available.

Relational Schema BibSonomy's database schema is centered around four tables: one for bookmark posts, one for publication posts, one for tag assignments (*tas*) and one for *relations*. Two further tables store informations regarding *users* and *groups*. In Figure 18, the two *posts* tables (*bookmark_posts* and *publication_posts*) are shown as one, and it is only hinted that these are really two tables. The reason to show them as one table *posts* is that they're very similar – the publication post table has just some additional columns to hold all the BibTeX fields. They are separated in the database for efficiency reasons, since these extra columns just need to be stored for publications.

The posts table is connected with the tas table by the key *post_id*. The scheme is not normalized – on the contrary we have added a high amount of redundancy to speed up queries. For example, besides storing group, user name and date in the posts table, we also store this information in the tas table to minimize the rows touched when selecting rows for the various views. A comparison of this schema against a normalized version is shown in Section 6.1.3. Several other tables hold counters (i. e., how many people share one resource, how often a tag is used, ...) and a lot of indexes (12 in the tas table alone) build the basis for fast answering of queries.

Table 7 Statistics on the number of rows in the BibSonomy database on August 12th, 2009. The 'public' column shows the counts disregarding spammers and non-public posts; the publication posts table also contains 1,220,775 posts from DBLP.

rows in table	all	public
tas (= $\|Y\|$)	41,108,680	3,244,245
bookmark_posts	5,118,340	331,414
publication_posts	1,792,887	1,584,997
users (= $\|U\|$)	242,704	85,018
groups	155	155
relations	12,330	9,893

Setup As RDBMS we use MySQL[24] with the iBATIS[25] data mapper framework as an intermediate layer between our Database-Logic component and MySQL. IBatis couples SQL statements (described in XML) with the objects of the model and thereby separates the Java code from the SQL code.

Overall, we spent a lot of time investigating and optimizing SQL queries and the table schema and tested both with folksonomy data of up to 8,000,000 posts. At the moment, BibSonomy has almost reached this size (see Table 7) and still delivers reasonable response times – as we will see in the next section. The system is scalable, since distribution of queries over synchronised databases is possible with MySQL. Consequently, to reduce the load on the main system, we are already running the database in a master/slave setting. The master handles all write access (and replicates it to the slave), as well as all non-search engine read access. The slave handles a part of the search engine read access as well as all reads necessary for offline computations. The computations are necessary to create tag counts, spam predictions, FolkRank rankings, or the fulltext search indexes.

6.1.3 Performance Analysis

As described in the previous section, for performance reasons the database scheme of BibSonomy is not normalized. Here we want to confirm this decision by comparing the query time of queries using a normalized schema against queries using the redundancy we introduced.

One of the most common queries in social bookmarking systems delivers all posts of a user which have one or more given tags attached. An example of such a query is the page http://www.bibsonomy.org/user/nepomuk/fca+folksonomy which requests all public posts of the user 'nepomuk' with the tags 'fca' and 'folksonomy' sorted by date in descending order.

In a normalized scheme, the user, group, and date information is stored in the bookmark posts table only, since it is the same for each tag assignment. Thus, the corresponding query in Listing 1, which extracts the bookmarks for the exemplary request, can restrict the rows in the tas table only by the 'tag' column. The query contains a self-join on the tas table to select only those posts which have both of the tags 'fca' and 'folksonomy'.

[24] http://www.mysql.com/
[25] http://ibatis.apache.org/

Listing 1 The query using the normalized schema.

```
SELECT a.url, a.title, a.date, a.ctr, t.tag
  FROM (
    SELECT u.url, b.title, b.date, u.ctr
      FROM bookmark_posts b
        JOIN urls u USING (res_id)
        JOIN tas t1 USING (post_id)
        JOIN tas t2 USING (post_id)
      WHERE b.group = "public"
        AND b.user = "nepomuk"
        AND t1.tag = "fca"
        AND t2.tag = "folksonomy"
      ORDER BY b.date DESC LIMIT 10
    ) AS a
    LEFT OUTER JOIN tas AS t USING (post_id)
  ORDER BY a.date DESC, a.post_id DESC
```

Listing 2 The query using the redundant schema.

```
1  SELECT a.url, a.title, a.date, a.ctr, t.tag
2    FROM (
3      SELECT u.url, b.title, b.date, u.ctr
4        FROM bookmark_posts b
5          JOIN urls u USING (res_id)
6          JOIN tas t1 USING (post_id)
7          JOIN tas t2 USING (post_id)
8        WHERE t1.content_type = 1
9          AND t1.group = "public"
10         AND t1.user = "nepomuk"
11         AND t1.tag = "fca"
12         AND t2.tag = "folksonomy"
13       ORDER BY t1.date DESC LIMIT 10
14     ) AS a
15     LEFT OUTER JOIN tas AS t USING (post_id)
16   ORDER BY a.date DESC, a.post_id DESC
```

In our scheme with high redundancy, the analogous query shown in Listing 2 is able to restrict the rows in the tas table by using the redundant columns 'group' and 'user' (lines 9 and 10). The 'content_type' column allows to distinguish between joins with the bookmark_posts or the publication_posts table (line 8) and thus further narrows down the result space. Finally, the sorting by date and subsequent limiting of the result set can be accomplished by using the 'date' column of the tas table.

To assess the performance of these two exemplary queries, we have performed tests on the live system of BibSonomy using the master database. Therefore, we measured the time it took to execute a query and retrieve all of its result rows. The setup is as follows:

1. We randomly choose a number k of tags with $1 \leq k \leq 5$.
2. We randomly choose a user u who has at least one public post and at least k tags.
3. We randomly choose k tags $t_1, \ldots, t_k$ from that user.
4. We query the database for all *public bookmark* posts of user u which have all of the tags $t_1, \ldots, t_k$ attached with both query types (i. e., using the *normalized* and the *redundant* scheme). The order in which we dispose the queries is randomly chosen.

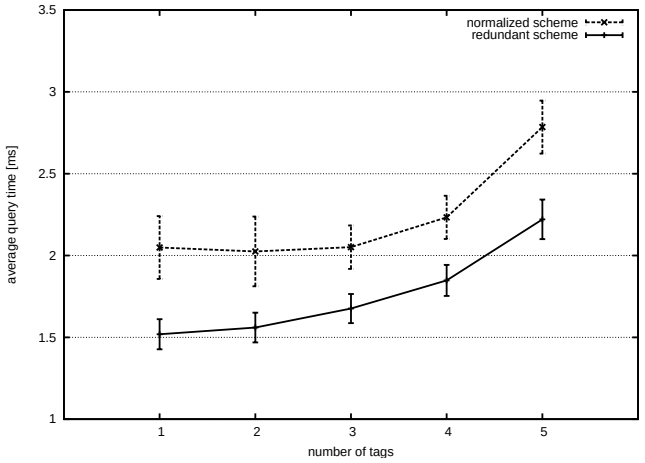


Figure 19 Comparison of the queries for bookmarks of a user – given some tags – using a normalized and a redundant scheme. The bars above and below each data point depict the standard deviation over 100 runs of step 5.

5. We measure the time it takes to execute step 4 and repeat steps 1 to 4 5,000 times. Thus, each number of tags (on average) is requested 1,000 times. Then, we compute the average query time for k tags t_k.
6. We repeat step 5 100 times and compute the average and standard deviation of t_k for each k over those 100 runs.

The machine hosting the databse has two 2GHz quad-core AMD Opteron processors. There, MySQL 5.0.67 was running with 22GB main memory and RAID-5 disks using the InnoDB storage engine.

The maximum query time was 6400 ms for the normalized and 446 ms for the redundant scheme queries. In general, the average query time is rather low with less than 3 ms for all numbers of tags (cf. Figure 19). With an increasing number of tags the growing number of self joins on the tas table causes increasing average query times. However, the average query time for one, two, and three tags for the normalized scheme does not follow this behaviour – we can't explain that, yet. As expected, the redundant scheme performs better – with both lower average query time and smaller standard deviation.

6.2 Web Front-End

The HTML-based web application is the most common way to interact with BibSonomy. It shows bookmark and publication lists together with navigational elements (see Figure 2). As described in Section 4, users can browse and search BibSonomy's underlying folksonomy by following links in the web interface that point to different pages or by entering search terms into input fields. Section 6.2.1 details on the semantics of the different pages accessible in the web application. BibSonomy also offers export facilities to a variety of formats like BIBTEX or RIS, suitable for integration into different text processing systems for automatic generation of bibliographies which are described in Section 6.2.2. Finally, the Application Programming Interface (API) which allows for easy interaction of BibSonomy with other systems is described in Section 6.2.3.

6.2.1 URL Scheme of the Web Application

Each of the pages accessible through the web application shows posts with certain properties, e. g., the `/tag/`t page shows all posts which contain the tag t. Here, we describe the *semantic* of all pages showing (bookmark and/or publication) posts but omit system pages which are necessary for the usage of BibSonomy like `/help`, `/settings` or `/postBookmark`, since their semantic is straightforward. All URLs are relative to http://www.bibsonomy.org/, i. e., only the path part is given. The following list describes the contents C of all pages which show posts in BibSonomy:

`/tag/`$t_1 \sqcup \ldots \sqcup t_n$ Every post which has all the tags $t_1, \ldots, t_n$ attached: $C_{t_1,\ldots,t_n} := \{(u,S,r) \in P \mid \{t_1,\ldots,t_n\} \subseteq S\}$

`/user/`u All posts of user u: $C_u := \{(\hat{u},S,r) \in P \mid \hat{u} = u\}$

`/user/`u`/`$t_1 \sqcup \ldots \sqcup t_n$ Every post of user u with all the tags $t_1,\ldots,t_n$: $C_{u,t_1,\ldots,t_n} := \{(\hat{u},S,r) \in P \mid \hat{u} = u, \{t_1,\ldots,t_n\} \subseteq S\}$

`/concept/`$t_1 \sqcup \ldots \sqcup t_n$ Every post $(u,S,r) \in P$ which has for every tag $t \in \{t_1,\ldots,t_n\}$ at least one of its subtags in $\prec \cap (\{u\} \times T \times T)$ or t itself attached: $C_{t_1,\ldots,t_n} := \{(u,S,r) \in P \mid \forall t_i (i = 1,\ldots,n) \exists t \in S : (u,t,t_i) \in \prec \vee t = t_i\}$

`/concept/user/`u`/`$t_1 \sqcup \ldots \sqcup t_n$ Every post of user u which has for every tag $t \in \{t_1,\ldots,t_n\}$ at least one of its subtags or t itself attached: $C_{u,t_1,\ldots,t_n} := \{(\hat{u},S,r) \in P \mid \hat{u} = u, \forall t_i (i = 1,\ldots,n) \exists t \in S : (\hat{u},t,t_i) \in \prec \vee t = t_i\}$

`/concept/group/`g`/`$t_1 \sqcup \ldots \sqcup t_n$ Every post which has for every tag $t \in \{t_1,\ldots,t_n\}$ at least one of its subtags in $\prec \cap (\{g\} \times T \times T)$ or t itself attached and where the user belongs to group g:[26] $C_{g,t_1,\ldots,t_n} := \{(u,S,r) \in P \mid u \in g, \forall t_i (i = 1,\ldots,n) \exists t \in S : (g,t,t_i) \in \prec \vee t = t_i\}$

`/url/`r If r is a bookmark: All posts of the resource r: $C_r := \{(u,S,\hat{r}) \in P \mid \hat{r} = r\}$

`/url/`r`/`u If r is a bookmark: The post of user u of the resource r: $C_{r,u} := \{(\hat{u},S,\hat{r}) \in P \mid \hat{r} = r, \hat{u} = u\}$

`/bibtex/`r If r is a literature reference: All posts of the resource r: $C_r := \{(u,S,\hat{r}) \in P \mid \hat{r} = r\}$

`/bibtex/`r`/`u If r is a literature reference: The post of user u of the resource r: $C_{r,u} := \{(\hat{u},S,\hat{r}) \in P \mid \hat{r} = r, \hat{u} = u\}$

`/group/`g All posts of all users belonging to the group g: $C_g := \{(u,S,r) \in P \mid u \in g\}$

`/group/`g`/`$t_1 \sqcup \ldots \sqcup t_n$ Every post which has all of the tags $t_1,\ldots,t_n$ attached and where the user belongs to group g: $C_{g,t_1,\ldots,t_n} := \{(u,S,r) \in P \mid u \in g, \{t_1,\ldots,t_n\} \subseteq S\}$

`/viewable/`g All posts which are set viewable for members of the group g.

`/viewable/`g`/`$t_1 \sqcup \ldots \sqcup t_n$ All posts which are set viewable for members of the group g and which have all of the tags $t_1,\ldots,t_n$ attached.

[26] Each group in BibSonomy is represented as a user and thus can participate in the $\prec$ relation.

`/search/s` All resources, whose full text matches the search expression *s*.

`/basket` All publication posts which the user has picked in his basket.

`/popular` The three most often posted resources of the last 7, 30, and 120 days.

`/` This is the home page of BibSonomy, it shows the entries posted most recently.

`/author/`a_1␣...␣a_n All publication posts which contain all of the names $a_1, \ldots, a_n$ in the author or editor field.

`/bibtexkey/`k Publication posts with the BibTeX key k.

An interesting feature, described in the following section, is the option to prepend all paths of URLs described above, by a string which changes the output format. In general, posts are shown as HTML lists surrounded by navigation elements and a tag cloud (as seen in Figure 2), but this feature allows the user to get her output in formats like BibTeX or as an RSS feed.

6.2.2 Exporting Bookmarks and Publication References

Exporting publication references in BibTeX format is accomplished by preceding the path of a URL showing publication posts with the string `/bib` – this returns all publications shown on the respective page in BibTeX format. For example, the page http://www.bibsonomy.org/bib/search/text+clustering returns a BibTeX file containing all literature references which contain the words 'text' and 'clustering' in their fulltext.

More general, every page which shows posts can be represented in several different ways by preceding the path part of the URL with one of the strings described here:

`/` the typical HTML view with navigation elements

`/csv` bookmarks and publications in CSV[27] format

`/xml` bookmarks in XML format

`/rss` bookmarks as RSS feed

`/bib` publications in BibTeX format

`/endnote` publications in EndNote format

`/publ` publications in a simple HTML format suited for integration into a web page (for an integration example see http://www.kde.cs.uni-kassel.de/pub)

`/publrss` publications as RSS feed

`/swrc` publications in RDF format according to the SWRC ontology[28]

`/burst` publications as RSS feed containing the complete metadata according to the SWRC ontology, embedded according to the BuRST[29]

`/json` bookmarks and publications in JSON[30] format (for an integration example using Simile Exhibit[31] see http://www.kde.cs.uni-kassel.de/hotho/publication_json.html)

`/layout/`l publications in one of the available JabRef[32] layouts. Valid values for l include `harvard` (RTF (Rich Text Format); can be edited by Microsoft Word and OpenOffice), `html`, `simplehtml`, `tablerefs`, `tablerefsabsbib`, `tablerefsabsbibsort` (different HTML layouts), or `docbook` (DocBook[33] XML).

`/layout/custom` publications in the custom layout of the user. Users can upload custom JabRef layouts[34] using the `/settings` page.

For an overview of the available export formats for publications, one can use the `/export` path extension which is also linked on all web pages showing publication posts. As an example, the URL http://www.bibsonomy.org/publrss/tag/fca represents an RSS feed showing the most recent publications tagged with the tag *fca*.

These export options simplify the interaction of BibSonomy with other systems. RSS feeds allow easy integration of resource lists into web sites or RSS aggregators, BibTeX output can be used to automatically generate bibliographies for scientific publications (as done with this work), JSON eases the handling of posts with JavaScript. In addition, further formats are implemented easily by extending the URL scheme and adding an appropriate JSP view which generates the output, or by writing a new JabRef layout filter. Rendering of the JabRef-based layouts is accomplished by the *Layout* component.

6.2.3 The REST-based API

Right from the start of BibSonomy, users demanded to open BibSonomy for programmatic access by providing an Application Programming Interface (API) which allows for easy interaction of BibSonomy with other systems. Furthermore, experience has shown that an API is crucial for a system to gain success. Consequently, we integrated a lightweight API [4] based on the idea of REST (Representational State Transfer) [13] which can be used and accessed also by not so experienced programmers. Every registered user can access the API at the URL http://www.bibsonomy.org/api/.

Adhering to the REST principle, the HTTP methods GET, PUT, POST, and DELETE are used to perform corresponding actions on BibSonomy's data by applying them to certain URLs. For example, a GET request to `/api/tags` returns the list of all tags of the system; a POST request to `/api/users/`u`/posts` creates a new post for user u. The API's input and output is XML data based on BibSonomy's data model which is serialized using the XML schema defined by the Model component. For a detailed description of the available methods we refer to the online documentation available at http://www.bibsonomy.org/help/doc/api.html.

[27] Comma Separated Values

[28] Semantic Web for Research Communities, see http://ontoware.org/projects/swrc/

[29] Bibliography Management using RSS Technology, see http://www.cs.vu.nl/~pmika/research/burst/BuRST.html

[30] JavaScript Object Notation, see http://www.json.org/

[31] http://code.google.com/p/simile-widgets/

[32] http://jabref.sourceforge.net/

[33] http://www.docbook.org/

[34] http://jabref.sourceforge.net/help/CustomExports.php

There is already an abundant amount of applications using the API, most of them are probably private developments of individuals and institutions using BibSonomy and thus unknown to the public. Prominent exceptions are the spam framework of BibSonomy (cf. Section 4.2.2), a plugin[35] for the Java based bibliographic reference manager JabRef,[36] and an add-on for the Firefox web browser.[37] Another important example is the Library of the University of Cologne, which has made various efforts for close interaction of its catalogue[38] with BibSonomy.

7 Related Work

The first social bookmarking system which gained wide popularity is *Delicious*. It has been founded in 2003 by Joshua Schachter and is operated by Yahoo! Inc. since December 2005. More than three million users[39] have contributed bookmarks to Delicious and thereby probably make it the largest human-annotated collection of web links.

CiteULike[40] is the largest collaborative tagging service for bibliographic references. In contrast to BibSonomy, only references imported from one of the supported digital libraries appear on the central web pages. On the one hand, this discourages spam posts, on the other hand, it restricts the freedom of the users to share publication references not listed on the main sites (e. g., books, technical reports, project works, etc.).

Another service which allows its users to share bibliographic references is Connotea,[41] operated by the Nature Publishing Group. In its size, it is comparable to BibSonomy. Both services started at the end of 2004.

In principle all these systems follow a similar usage pattern. The services provide a bookmarklet for the browser to make the posting of a new reference as easy as possible. As mentioned, reference information is automatically imported from a large fraction of publisher webpages and other digital libraries. Furthermore, every user has the freedom to organize his own references, but he can use the shared information of others to search and browse for new, potentially interesting publications.

The Connotea system is written in Perl, uses a MySQL database and is released as open source software. It provides an API which offers the basic function of a bookmarking system. The community has developed a remarkable list of tools.[42] As far as we know, there is no special search function. An item recommender which offers a list of similar items is integrated. Unfortunately, Connotea responds often very slowly which seems to be a problem of the underlying database. The FAQ states that it has some spam prevention, but no details about the applied methods are mentioned.

In contrast to BibSonomy and Connotea, CiteUlike is a closed source system and the information about the internal structure is rare. CiteULike uses Lucene as search backend (mentioned on the help pages) and PostgreSQL as a database in combination with several scripting languages (cf. [16]). It does not provide an API, but there is a community SVN for import filters. Again, an item recommender is included.[43] The service recommends new references to the user based on its currently stored references (cf. [3]). The system maintainers have worked a lot on the interface in the past, and response times of CiteUlike are always short. In a blog post[44] it is mentioned that CiteUlike has only a small fraction of spam posts, but the inspection of the released dataset shows a significant amount of spam.

Another service similar to BibSonomy is LibraryThing.[45] Its paradigm is more a copy of the user's bookshelf than a scientific library, and the system focuses on a more general audience rather than on scientists. Further away in the application landscape is the web browser plug-in Zotero, followed by stand-alone reference managing systems like JabRef, CiteSmart, Citavi, EndNote, or Mendeley. BibSonomy is integrated with the first three systems, while Mendeley goes the other way around, and is complementing its desktop based tool with a web sharing component.

Folksonomies and especially data mining on folksonomies are a relatively young research area. Meanwhile, work for specific areas has shown up. To discuss the related work for all methods mentioned in the previous sections is beyond the scope of this article. More detailed surveys can be found in the corresponding papers [27, 21, 25, 10, 33, 23, 32, 29, 30].

To start with folksonomies and to learn more about their strengths and weaknesses one may look into [16, 35, 37]. One of the first works defining a model of semantic-social networks for extracting lightweight ontologies from Delicious was [39]. Work on more specialized topics such as structure mining on folksonomies – e. g., to visualize trends [12], analyze patterns in users' tagging behavior [47] and the semiotic dynamics of the tagging vocabulary [8], or Halpin's analysis of the dynamics and semantics [15] – has been presented. Steels [48] considers – looking from a pychological perspective – collaborative tagging as distributed cognition. More practical questions, e. g., if [20] and how [54] social bookmarking can improve keyword based (web) search, have come up more recently.

[35] http://www.bibsonomy.org/help/doc/jabref-plugin/index.html

[36] http://jabref.sourceforge.net/

[37] missingreference

[38] "Kölner Universitäts-Gesamtkatalog", http://kug.ub.uni-koeln.de/

[39] This is a well-founded estimate based on the authors' regular crawling activity. On September 25th 2006, Joshua Schachter announced to have reached the one million mark [46].

[40] http://www.citeulike.com/

[41] http://www.connotea.org/

[42] http://www.connotea.org/wiki/ConnoteaTools

[43] http://blog.citeulike.org/?p=11

[44] http://network.nature.com/people/mfenner/blog/2009/01/30/interview-with-kevin-emamy

[45] http://www.librarything.com/

8 Conclusion

Data Mining on folksonomies is a new research area having attracted a lot of attention in the last years, as new types of data with unknown and interesting properties appear. In this paper we presented our own system, BibSonomy, for managing publications on a daily basis which is open for integration of new data mining techniques and thus serves as a research environment. We presented new methods for analyzing the properties of folksonomies, the application and adaptation of known data mining approaches, and the usage of this data to extract semantic information. We showed how these techniques were realized and used in BibSonomy and which constraints of such a real life application have to be considered thereby.

Recent research cooperations, as well as hosting of the 2009 ECML PKDD Discovery Challenge underpin the success of this work. BibSonomy is under active development and still evolving, so insights into new data mining methods and their application are to be expected.

Acknowledgements Parts of this research were funded by the European Union in the Tagora and Nepomuk projects, by the German Research Foundation (DFG) in the projects "Info 2.0 – Informationelle Selbstbestimmung im Web 2.0" and "PUMA – Akademisches Publikationsmanagement", and by Land Hessen in the VENUS project.

References

1. Batagelj V, Zaversnik M (2002) Generalized cores. CoRR cs.DS/0202039, URL http://arxiv.org/abs/cs/0202039
2. Bayardo RJ, Ma Y, Srikant R (2007) Scaling up all pairs similarity search. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, ACM, New York, NY, USA, pp 131–140
3. Bogers T (2009) Recommender systems for social bookmarking. PhD thesis, Tilburg University, Tilburg, The Netherlands, URL http://ilk.uvt.nl/~toine/phd-thesis/
4. Bork M (2006) Webservice API für Bibsonomy. Project report, URL http://www.kde.cs.uni-kassel.de/lehre/arbeiten/documents/bork2006webservice.pdf
5. Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14^{th} Conference on Uncertainty in Artificial Intelligence, pp 43–52
6. Brin S, Page L (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems 30(1-7):107–117
7. Budanitsky A, Hirst G (2006) Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics 32(1):13–47
8. Cattuto C, Loreto V, Pietronero L (2006) Collaborative tagging and semiotic dynamics. CoRR abs/cs/0605015, URL http://arxiv.org/abs/cs/0605015
9. Cattuto C, Schmitz C, Baldassarri A, Servedio VDP, Loreto V, Hotho A, Grahl M, Stumme G (2007) Network properties of folksonomies. AI Communications 20(4):245–262
10. Cattuto C, Benz D, Hotho A, Stumme G (2008) Semantic grounding of tag relatedness in social bookmarking systems. The Semantic Web - ISWC 2008 pp 615–631
11. Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
12. Dubinko M, Kumar R, Magnani J, Novak J, Raghavan P, Tomkins A (2006) Visualizing tags over time. In: Proceedings of the 15th International WWW Conference
13. Fielding RT (2000) Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine
14. Golder S, Huberman BA (2005) The structure of collaborative tagging systems. CoRR abs/cs/0508082, URL http://arxiv.org/abs/cs.DL/0508082
15. Halpin H, Robu V, Shepard H (2006) The dynamics and semantics of collaborative tagging. In: Möller K, de Waard A, Cayzer S, Koivunen MR, Sintek M, Handschuh S (eds) Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW'06), CEUR-WS.org, vol 209
16. Hammond T, Hannay T, Lund B, Scott J (2005) Social Bookmarking Tools (I): A General Review. D-Lib Magazine 11(4)
17. Haveliwala TH (2003) Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. Technical Report 2003-29, Stanford InfoLab, URL http://ilpubs.stanford.edu:8090/750/, extended version of the WWW2002 paper on Topic-Sensitive PageRank.
18. Herlocker JL, Konstan JA, Terveen LG, Riedl JT (2004) Evaluating collaborative filtering recommender systems. ACM Trans Inf Syst 22(1):5–53
19. Heymann P, Koutrika G, Garcia-Molina H (2007) Fighting spam on social web sites: A survey of approaches and future challenges. IEEE Internet Computing 11(6):36–45
20. Heymann P, Koutrika G, Garcia-Molina H (2008) Can social bookmarking improve web search? In: WSDM '08: Proc. of the Int. Conf. on Web Search and Web Data Mining, ACM, New York, NY, USA, pp 195–206
21. Hotho A, Jäschke R, Schmitz C, Stumme G (2006) BibSonomy: A social bookmark and publication sharing system. In: de Moor A, Polovina S, Delugach H (eds) Proc. of the Conceptual Structures Tool Interoperability Workshop, Aalborg University Press, Aalborg, Denmark, pp 87–102
22. Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Emergent semantics in BibSonomy. In: Hochberger C, Liskowsky R (eds) Informatik 2006 - Informatik für Menschen, Gesellschaft für Informatik, Bonn, Lecture Notes in Informatics, vol 94
23. Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Information retrieval in folksonomies: Search and ranking. In: Sure Y, Domingue J (eds) The Semantic Web: Research and Applications, Springer, Berlin/Heidelberg,

Lecture Notes in Computer Science, vol 4011, pp 411–426
24. Hotho A, Jäschke R, Schmitz C, Stumme G (2006) Trend detection in folksonomies. In: Avrithis YS, Kompatsiaris Y, Staab S, O'Connor NE (eds) Proc. First International Conference on Semantics And Digital Media Technology (SAMT), Springer, Berlin/Heidelberg, Lecture Notes in Computer Science, vol 4306, pp 56–70
25. Hotho A, Jäschke R, Benz D, Grahl M, Krause B, Schmitz C, Stumme G (2009) Social Bookmarking am Beispiel BibSonomy. In: Blumauer A, Pellegrini T (eds) Social Semantic Web, X.media.press, Springer, Berlin/Heidelberg, chap 18, pp 363–391, DOI 10.1007/978-3-540-72216-8
26. Jiang JJ, Conrath DW (1997) Semantic Similarity based on Corpus Statistics and Lexical Taxonomy. In: Proceedings of the International Conference on Research in Computational Linguistics (ROCLING), Taiwan
27. Jäschke R, Grahl M, Hotho A, Krause B, Schmitz C, Stumme G (2007) Organizing publications and bookmarks in BibSonomy. In: Alani H, Noy N, Stumme G, Mika P, Sure Y, Vrandecic D (eds) Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at WWW 2007, Banff, Canada
28. Jäschke R, Marinho LB, Hotho A, Schmidt-Thieme L, Stumme G (2007) Tag recommendations in folksonomies. In: Kok JN, Koronacki J, de Mántaras RL, Matwin S, Mladenic D, Skowron A (eds) Knowledge Discovery in Databases: PKDD 2007, Springer, Berlin/Heidelberg, Lecture Notes in Computer Science, vol 4702, pp 506–514
29. Jäschke R, Marinho L, Hotho A, Schmidt-Thieme L, Stumme G (2008) Tag recommendations in social bookmarking systems. AI Communications 21(4):231–247
30. Jäschke R, Eisterlehner F, Hotho A, Stumme G (2009) Testing and evaluating tag recommenders in a live system. In: RecSys '09: Proc. of the 2009 ACM Conf. on Recommender Systems, ACM, New York, NY, USA, (to appear)
31. Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. Journal of the ACM 46(5):604–632
32. Krause B, Jäschke R, Hotho A, Stumme G (2008) Logsonomy - social information retrieval with logdata. In: HT '08: Proc. of the 19th ACM Conference on Hypertext and Hypermedia, ACM, New York, NY, USA, pp 157–166
33. Krause B, Schmitz C, Hotho A, Stumme G (2008) The anti-social tagger – detecting spam in social bookmarking systems. In: AIRWeb '08: Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web, ACM, New York, NY, USA, pp 61–68
34. Lehmann F, Wille R (1995) A triadic approach to formal concept analysis. In: Ellis G, Levinson R, Rich W, Sowa JF (eds) Conceptual Structures: Applications, Implementation and Theory, Springer, Berlin/Heidelberg, Lecture Notes in Artificial Intelligence, vol 954, pp 32–43
35. Lund B, Hammond T, Flack M, Hannay T (2005) Social Bookmarking Tools (II): A Case Study - Connotea. D-Lib Magazine 11(4)
36. Marinho LB, Schmidt-Thieme L (2007) Collaborative tag recommendations. In: Proceedings of 31st Annual Conference of the Gesellschaft für Klassifikation (GfKl), Freiburg, Springer
37. Mathes A (2004) Folksonomies – Cooperative Classification and Communication Through Shared Metadata. URL http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html
38. McCallum AK (2002) MALLET: A Machine Learning for Language Toolkit. URL http://mallet.cs.umass.edu/
39. Mika P (2005) Ontologies are us: A unified model of social networks and semantics. In: Gil Y, Motta E, Benjamins VR, Musen MA (eds) Proc. of the 4th Int. Semantic Web Conference, Springer, Berlin/Heidelberg, Lecture Notes in Computer Science, vol 3729, pp 522–536
40. Peng F, McCallum A (2004) Accurate information extraction from research papers using conditional random fields. In: HLT-NAACL, pp 329–336
41. Quintarelli E (2005) Folksonomies: power to the people. URL http://www-dimat.unipv.it/biblio/isko/doc/folksonomies.htm
42. Reenskaug T (1979) Models-views-controllers. Tech. rep., Xerox PARC
43. Resnick P, Iacovou N, Suchak M, Bergstorm P, Riedl J (1994) GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In: Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work, ACM, Chapel Hill, North Carolina, pp 175–186
44. Salton G (1989) Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA
45. Sarwar BM, Karypis G, Konstan JA, Reidl J (2001) Item-based collaborative filtering recommendation algorithms. In: World Wide Web, pp 285–295
46. Schachter J (2006) now serving: 1,000,000. Blog post, URL http://blog.delicious.com/blog/2006/09/million.html
47. Schmitz C, Hotho A, Jäschke R, Stumme G (2006) Mining association rules in folksonomies. In: Batagelj V, Bock HH, Ferligoj A, Žiberna A (eds) Data Science and Classification: Proc. of the 10th IFCS Conf., Springer, Berlin/Heidelberg, Studies in Classification, Data Analysis, and Knowledge Organization, pp 261–270
48. Steels L (2006) Collaborative tagging as distributed cognition. Pragmatics & Cognition 14(2):287–292
49. Stumme G (2005) A finite state model for on-line analytical processing in triadic contexts. In: Ganter B, Godin R (eds) Proceedings of the 3rd International Conference on Formal Concept Analysis, Springer, Berlin/Heidelberg, Lecture Notes in Computer Science, vol 3403, pp 315–328
50. Vander Wal T (2007) Folksonomy. Blog post, URL http://vanderwal.net/folksonomy.html
51. Wille R (1982) Restructuring lattice theory: an approach based on hierarchies of concepts. In: Rival I (ed) Ordered sets, Reidel, Dordrecht–Boston, pp 445–470

52. Witten IH, Frank E (1999) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann
53. Xi W, Zhang B, Lu Y, Chen Z, Yan S, Zeng H, Ma W, Fox E (2004) Link fusion: A unified link analysis framework for multi-type interrelated data objects. In: Proc. 13th International World Wide Web Conference, New York
54. Yahia SA, Benedikt M, Lakshmanan LVS, Stoyanovich J (2008) Efficient network aware search in collaborative tagging sites. Proceedings of the VLDB Endowment 1(1):710–721, DOI 10.1145/1453856.1453934