

Formal Concept Analysis and Tag Recommendations in Collaborative Tagging Systems

Dissertation zur Erlangung des akademischen Grades eines Doktors der
Naturwissenschaften (Dr. rer. nat.)

im Fachbereich 16 Elektrotechnik/Informatik der Universität Kassel

vorgelegt von Robert Jäschke

Kassel im März 2010

Erklärung

Hiermit versichere ich, dass ich die vorliegende Dissertation selbständig und ohne unerlaubte Hilfe angefertigt und andere als die in der Dissertation angegebenen Hilfsmittel nicht benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Kein Teil dieser Arbeit ist in einem anderen Promotions- oder Habilitationsverfahren verwendet worden.

Kassel, den 4. März 2010

Robert Jäschke

Formal Concept Analysis and Tag Recommendations in Collaborative Tagging Systems

Contents

1	Introduction	1
1.1	Topics of this Thesis	1
1.2	Contributions of this Thesis	2
1.3	Structure of this Thesis	3
I	Collaborative Tagging Systems	5
2	Foundations and State of the Art	7
2.1	Collaborative Tagging Systems	8
2.2	Folksonomies	10
2.3	A Formal Model	12
2.4	Related Work	14
3	BibSonomy	15
3.1	Introduction	15
3.2	Related Work	16
3.3	User Interface	18
3.4	Architecture	20
3.4.1	Components	20
3.4.2	Database	23
3.4.3	Semantics of the URL Scheme	25
3.5	Features	28
3.5.1	Importing Resources	28
3.5.2	Exporting Bookmarks and Publication References	29
3.5.3	Supporting Research Groups	31
3.5.4	A REST-based API	32
3.5.5	Tag Relations	33
3.5.6	Bibliographic Hash Keys for Duplicate Detection	34
3.5.7	Collecting Publication Posts with the ‘Basket’	35
3.5.8	Tag Editing	35
3.5.9	FolkRank	35
3.6	Outlook	36

II	Knowledge Discovery	39
4	Formal Concept Analysis	41
4.1	Introduction	41
4.1.1	Discovering Shared Conceptualizations	41
4.1.2	The Problem of Closed Itemset Mining in Triadic Data	42
4.1.3	Contribution and Organization of this Chapter	43
4.2	Basic Notions and State of the Art	44
4.2.1	Ontology Learning	44
4.2.2	Formal Concept Analysis	45
4.2.3	Triadic Concept Analysis	46
4.2.4	Closed Itemset Mining	47
4.3	Mining Frequent Tri-Concepts of a Folksonomy	49
4.3.1	The Problem of Mining all Frequent Tri-Concepts	49
4.3.2	The TRIAS Algorithm for Mining all Frequent Tri-Concepts	50
4.3.3	Performance of the TRIAS Algorithm	53
4.4	Qualitative Evaluation	55
4.4.1	Conceptual Clustering of the Delicious Folksonomy	55
4.4.2	IT Baseline Protection Manual	59
4.4.3	Conceptual Analysis of the BibSonomy Publication Data	62
4.5	Neighborhoods of Triadic Concepts	66
4.6	Conclusion	70
5	Tag Recommendations	73
5.1	Introduction	73
5.2	Problem Definition and Related Work	74
5.2.1	Problem Definition	74
5.2.2	Related Work	74
5.3	Algorithms	76
5.3.1	Collaborative Filtering	76
5.3.2	A Graph-Based Approach	78
5.3.3	Most Popular Tags	80
5.4	Computational Costs	82
5.4.1	Collaborative Filtering	82
5.4.2	The Graph-Based Approach	83
5.4.3	Most Popular Tags	83
5.4.4	Comparison	83
5.5	Quantitative Evaluation	84
5.5.1	Datasets	84
5.5.2	Core Computation	85
5.5.3	Evaluation Measures	86

5.5.4	Settings of the Algorithms	87
5.6	Results	88
5.6.1	Delicious	88
5.6.2	BibSonomy	92
5.6.3	Last.fm	93
5.7	Conclusion	95
III Applications		97
6	A Tag Recommendation Framework for BibSonomy	99
6.1	Introduction	99
6.2	Related Work	101
6.3	The Framework	102
6.3.1	Recommender Interface	103
6.3.2	Meta Recommender	104
6.3.3	Remote Recommender	105
6.3.4	Multiplexing Tag Recommender	105
6.3.5	Example Recommender Implementations	106
6.4	Evaluation	107
6.4.1	Measures	108
6.4.2	Data Cleansing	108
6.4.3	Logging	108
6.5	Results	109
6.5.1	General	109
6.5.2	Influence of the ‘reload’ Button	110
6.5.3	Logged ‘click’ Events	111
6.5.4	Average F1-Measure per User	113
6.6	The ECML PKDD Discovery Challenge 2009	113
6.6.1	Setting	114
6.6.2	Methods	115
6.6.3	Results	116
6.7	Conclusion	118
7	Community Support for the Social Semantic Desktop	119
7.1	Introduction	119
7.2	Motivation	120
7.3	Related Work	121
7.4	Overview of the Nepomuk Architecture	121
7.4.1	Semantic Desktop Services	122
7.4.2	Service Registry	123

7.4.3	Social Services	123
7.4.4	Extension Services	123
7.4.5	Enduser Applications	124
7.5	Community Services Architecture	124
7.5.1	Component Descriptions	124
7.5.2	Gathering Data for Analysis	125
7.5.3	Relation to Other Services	126
7.6	Tag Recommendations	127
7.6.1	Realization	128
7.6.2	Interaction	129
7.7	Community Detection and Labeling	130
7.7.1	Motivation	130
7.7.2	Realization	131
7.7.3	Interaction	132
7.8	Conclusion	134
8	Logsonomies	137
8.1	Introduction	137
8.2	Related Work	138
8.3	Search Engine Query Logs as Logsonomies	140
8.4	Datasets	141
8.5	Strength in the Tag-Tag-Co-Occurrence Graph	142
8.5.1	Cumulative Strength Distribution	143
8.5.2	Average Nearest-Neighbor Strength	144
8.6	Outlook	146
9	Outlook	149
9.1	Collaborative Tagging Systems	149
9.1.1	New Paradigms and Challenges	149
9.1.2	BibSonomy	150
9.2	Tag Recommendations	151
9.2.1	Different Types of Tags	151
9.2.2	Personalization	151
9.2.3	Trust	152
9.2.4	Further Aspects	153
9.3	Formal Concept Analysis	153
9.3.1	Visualization	153
9.3.2	Large Datasets	154
9.3.3	Triadic Association Rules	154
9.4	Conclusion	155

List of Figures

2.1	Screenshot of a publication reference post in BibSonomy.	8
2.2	Screenshot of a user's tag cloud in BibSonomy.	9
2.3	Schematic view of a folksonomy.	11
2.4	Excerpt of a folksonomy hypergraph.	13
3.1	The growth of the number of users in BibSonomy.	16
3.2	A screenshot of BibSonomy.	18
3.3	Detailed screenshots of a bookmark and a publication post in BibSonomy.	19
3.4	Lines of code of the BibSonomy project.	21
3.5	UML component diagram of BibSonomy's components.	22
3.6	UML diagram of BibSonomy's data model.	24
3.7	Relational schema of the most important tables.	25
3.8	Screenshot of a literature reference.	30
3.9	The top five posts for the tag <i>folksonomy</i> sorted by FolkRank.	37
3.10	The related users and tags for the tag <i>folksonomy</i> sorted by FolkRank.	38
4.1	The history of iceberg tri-lattices.	43
4.2	Accessing the triples of Y in sorted order.	53
4.3	The number of frequent tri-sets vs. the number of frequent tri-concepts.	54
4.4	A comparison of the runtime of the triadic NEXT CLOSURE and the TRIAS algorithm.	55
4.5	Examples of frequent tri-concepts of Delicious.	57
4.6	The most relevant tags and resources related to the tags <i>css</i> , <i>web</i> , and <i>design</i>	58
4.7	All frequent tri-concepts of the IT Baseline Security Manual for $\tau_u = \tau_t = \tau_r = 3$	60
4.8	All frequent tri-concepts of the IT Baseline Security Manual for $\tau_u = 3, \tau_t = 2, \tau_r = 3$	61
4.9	All frequent tri-concepts of the BibSonomy publications for $\tau_u = 3, \tau_t = 2, \tau_r = 2$	64
4.10	The neighborhood graph of the neighborhood $[(\{hotho, schmitz, stumme\}, \{2006, bibsonomy, myown\}, \{4, 10\})]$	67
4.11	A neighborhood graph of the IT Baseline Security Manual.	68
4.12	Neighborhood graph of the IT Baseline Security Manual.	69

5.1	Projections of Y into the user's resource and user's tag spaces.	77
5.2	Recall and precision of the <i>most popular tags mix 1 : 1</i> and the <i>most popular tags ρ-mix</i> for $\rho \in \{0, 0.1, \dots, 0.9, 1\}$ on the Delicious p -core at level 10.	89
5.3	Recall for Delicious p -core at level 10.	90
5.4	Recall and precision for Delicious p -core at level 10.	91
5.5	Recall and precision for Delicious.	93
5.6	Recall and precision for BibSonomy p -core at level 5.	94
5.7	Recall and precision for Last.fm p -core at level 10.	95
6.1	BibSonomy's recommendation interface.	100
6.2	The schematic recommendation process.	102
6.3	The UML class diagram of the tag recommender interface.	103
6.4	Recall and precision of the two deployed recommenders.	110
6.5	Users sorted by their fraction of click/noclick-posts.	112
6.6	The fraction of matching tags which have been clicked.	113
6.7	Average f1-measure for each user and recommender.	114
6.8	Recall and precision.	116
6.9	Latency of the recommenders.	117
6.10	Offline performance of the recommenders.	117
7.1	Services of Nepomuk.	122
7.2	Tagging as modeled in NAO.	126
7.3	Example SPARQL query used to gather tagging data.	126
7.4	The dependency graph of the <i>TagRecommender</i> and the <i>FolkPeer</i>	128
7.5	Recommendations in the <i>PimoEditor</i> returned by the <i>TagRecommender</i>	130
7.6	The dependency graph of the <i>CommunityManager</i> and the <i>FolkPeer</i>	132
7.7	The <i>search</i> perspective of PSEW.	133
7.8	The community for the topic 'ontology'.	134
8.1	The cumulative strength distribution.	143
8.2	Average nearest-neighbor strength in Delicious.	145
8.3	Average nearest-neighbor strength in AOL.	146
8.4	Average nearest-neighbor strength in MSN.	147

List of Tables

4.1	The mapping of publication IDs to publication titles.	62
4.1	The mapping of publication IDs to publication titles.	63
5.1	Characteristics of the used datasets.	85
5.2	Characteristics of the p -cores at level k	86
5.3	Average number of tags per user and tags per resource.	94
6.1	The influence of the ‘reload’ button.	111
6.2	The number of posts regarded for evaluation.	115
8.1	Folksonomy and logsonomy datasets.	142

List of Algorithms

4.1	The TRIAS algorithm for mining all frequent tri-concepts.	51
4.2	The <i>FirstFrequentConcept</i> method of the TRIAS algorithm.	52
4.3	The <i>NextFrequentConcept</i> method of the TRIAS algorithm.	52
6.1	The Java method used to clean tags.	108

Nomenclature

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BuRST	Bibliography Management using RSS Technology
CSV	Comma Separated Values
DOM	Document Object Model
FCA	Formal Concept Analysis
FOAF	Friend Of A Friend
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
P2P	Peer-To-Peer
PIMO	Personal Information Model Ontology
PSEW	Personal Semantic Eclipse Workbench
RDF	Resource Description Framework
REST	Representational State Transfer
RFID	Radio Frequency Identification
RSS	RDF Site Summary
RTF	Rich Text Format
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SVM	Support Vector Machine
SWRC	Semantic Web for Research Communities
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

Abstract

One of the most noticeable innovation that emerged with the advent of the Web 2.0 and the focal point of this thesis are *collaborative tagging systems*. They allow users to annotate arbitrary resources with freely chosen keywords, so called *tags*. The tags are used for navigation, finding resources, and serendipitous browsing and thus provide an immediate benefit for the user. By now, several systems for tagging photos, web links, publication references, videos, etc. have attracted millions of users which in turn annotated countless resources. Tagging gained so much popularity that it spread into other applications like web browsers, software packet managers, and even file systems. Therefore, the relevance of the methods presented in this thesis goes beyond the Web 2.0.

The conceptual structure underlying collaborative tagging systems is called *folksonomy*. It can be represented as a tripartite hypergraph with user, tag, and resource nodes. Each edge of the graph expresses the fact that a user annotated a resource with a tag. This social network constitutes a lightweight conceptual structure that is not formalized, but rather implicit and thus needs to be extracted with knowledge discovery methods. In this thesis a new data mining task – the *mining of all frequent tri-concepts* – is presented, together with an efficient algorithm for discovering such implicit shared conceptualizations. Our approach extends the data mining task of discovering all closed itemsets to three-dimensional data structures to allow for mining folksonomies. Extending the theory of *triadic Formal Concept Analysis*, we provide a formal definition of the problem, and present an efficient algorithm for its solution. We show the applicability of our approach on three large real-world examples and thereby perform a conceptual clustering of two collaborative tagging systems. Finally, we introduce neighborhoods of triadic concepts as basis for a lightweight visualization of tri-lattices.

The social bookmark and publication sharing system *BibSonomy*, which is currently among the three most popular systems of its kind, has been developed by our research group. Besides being a useful tool for many scientists, it provides interested researchers a basis for the evaluation and integration of their knowledge discovery methods. This thesis introduces BibSonomy as an exemplary collaborative tagging system and gives an overview of its architecture and some of its features. Furthermore, BibSonomy is used as foundation for evaluating and integrating some of the discussed approaches.

Collaborative tagging systems usually include *tag recommendation* mechanisms easing the process of finding good tags for a resource, but also consolidating the tag vocabulary across users. In this thesis we evaluate and compare several recommendation algorithms on large-scale real-world datasets: an adaptation of user-based Collaborative Filtering, a graph-based recommender built on top of the FolkRank algorithm, and simple methods

based on counting tag co-occurrences. We show that both FolkRank and Collaborative Filtering provide better results than non-personalized baseline methods. Moreover, since methods based on counting tag co-occurrences are computationally cheap, and thus usually preferable for real time scenarios, we discuss simple approaches for improving the performance of such methods. We demonstrate how a simple recommender based on counting tags from users and resources can perform almost as good as the best recommender. Furthermore, we show how to integrate recommendation methods into a real tagging system, record and evaluate their performance by describing the tag recommendation framework we developed for BibSonomy. With the intention to develop, test, and evaluate recommendation algorithms and supporting cooperation with researchers, we designed the framework to be easily extensible, open for a variety of methods, and usable independent from BibSonomy. We also present an evaluation of the framework which demonstrates its power.

The folksonomy graph shows specific structural properties that explain its growth and the possibility of serendipitous exploration. Clicklogs of web search engines can be represented as a folksonomy in which queries are descriptions of clicked URLs. The resulting network structure, which we will term *logsonomy* is very similar to the one of folksonomies. In order to find out about its properties, we analyze the topological characteristics of the tripartite hypergraph of queries, users and bookmarks on a large folksonomy snapshot and on query logs of two large search engines. We find that all of the three datasets exhibit similar structural properties and thus conclude that the clicking behaviour of search engine users based on the displayed search results and the tagging behaviour of collaborative tagging users is driven by similar dynamics.

In this thesis we further transfer the folksonomy paradigm to the *Social Semantic Desktop* – a new model of computer desktop that uses Semantic Web technologies to better link information items. There we apply community support methods to the folksonomy found in the network of social semantic desktops. Thus, we connect knowledge discovery for folksonomies with semantic technologies.

Alltogether, the research in this thesis is centered around collaborative tagging systems and their underlying datastructure – folksonomies – and thereby paves the way for the further dissemination of this successful knowledge management paradigm.

Zusammenfassung

Eine der bemerkenswertesten Neuerungen, die mit dem Aufkommen des Web 2.0 in Erscheinung trat, und der Schwerpunkt dieser Arbeit sind *kollaborative Verschlagwortungssysteme*. Sie ermöglichen Benutzern, beliebige Ressourcen mit frei wählbaren Schlagworten – so genannten *Tags* – zu versehen. Die Tags dienen dabei der Navigation, dem Auffinden von Ressourcen und dem zufälligen Entdecken neuer Ressourcen und bringen dem Benutzer daher unmittelbare Vorteile. Mittlerweile haben mehrere Systeme zum Verschlagworten von Photos, Webseiten, Publikationsreferenzen, Videos, usw. mehrere Millionen Benutzer angezogen, die wiederum zahllose Ressourcen annotiert haben. Das Taggen ist so beliebt geworden, dass es mittlerweile in anderen Anwendungen, wie Webbrowsern, Softwarepaketmanagern und sogar Dateisystemen, Einzug gehalten hat. Die Relevanz der in dieser Arbeit gezeigten Methoden geht daher über das Web 2.0 hinaus.

Die begriffliche Struktur, die kollaborativen Verschlagwortungssystemen zu Grunde liegt, nennt man *Folksonomie*. Sie kann dargestellt werden durch einen tripartiten Hypergraphen mit Benutzern, Tags und Ressourcen als Knoten. Jede Kante des Graphen entspricht dabei der Verschlagwortung einer Ressource durch einen Benutzer. Dieses soziale Netzwerk stellt eine leichtgewichtige begriffliche Struktur dar, die nicht formalisiert sondern implizit ist und deswegen durch Methoden zur Wissensentdeckung extrahiert werden muss. In dieser Arbeit führen wir daher eine neue Aufgabe im Bereich des Data-mining ein: das Entdecken von häufigen Tri-Begriffen. Gleichzeitig präsentieren wir ein effizientes Verfahren zur Entdeckung solcher Begriffe. Unser Ansatz erweitert die Entdeckung von Closed Itemsets auf dreidimensionale Datenstrukturen, um das Data-mining auf Folksonomien zu ermöglichen. Dabei stellen wir durch die Erweiterung der Theorie der *triadischen Formalen Begriffsanalyse* eine formale Definition des Problems bereit und präsentieren ein effizientes Verfahren zu dessen Lösung. Wir zeigen die Anwendbarkeit unserer Methode an drei großen realen Beispielen und erhalten dabei eine begriffliche Clusterung zweier kollaborativer Verschlagwortungssysteme. Schließlich stellen wir Nachbarschaften triadischer Begriffe als Grundlage einer vereinfachten Visualisierung von Tri-Verbänden vor.

Das kollaborative Verschlagwortungssystem *BibSonomy* ermöglicht das Verschlagworten von Web-Lesezeichen und Publikationsreferenzen. Als eines der drei beliebtesten Systeme seiner Art wurde es von unserer Forschungsgruppe entwickelt. BibSonomy ist einerseits ein nützliches Werkzeug für Forscher zum Organisieren ihrer Publikationsreferenzen, andererseits stellt es interessierten Forschern eine Umgebung zum Evaluieren und Integrieren ihrer Wissensentdeckungsmethoden bereit. Diese Arbeit stellt BibSonomy beispielhaft als kollaboratives Verschlagwortungssystem vor und gibt einen Überblick

über seine Architektur und Fähigkeiten. Desweiteren wird BibSonomy als Grundlage zur Evaluierung und Integration einiger der vorgestellten Methoden genutzt.

Kollaborative Verschlagwortungssysteme enthalten üblicherweise Verfahren, um Benutzern Tags *vorzuschlagen*. Diese Verfahren vereinfachen einerseits das Finden von guten Tags für eine Ressource, andererseits helfen sie dabei, das Tag-Vokabular zwischen Benutzern zu vereinheitlichen. In dieser Arbeit evaluieren und vergleichen wir verschiedene Vorschlagsmethoden auf drei großen realen Datensätzen: eine Anpassung von Benutzer-basiertem Collaborative Filtering, ein Graphen-basiertes Verfahren auf Grundlage des FolkRank-Algorithmus und einfachere Methoden basierend auf Vorkommenshäufigkeiten von Tags. Wir zeigen, dass sowohl FolkRank als auch Collaborative Filtering bessere Resultate erzielen als nicht-personalisierte Verfahren. Desweiteren diskutieren wir einfache Möglichkeiten zur Verbesserung von Verfahren, die auf dem Zählen von Tag-Vorkommen basieren, denn diese sind besonders leicht zu berechnen und daher vorteilhaft für Echtzeitanwendungen. Dabei zeigen wir, dass ein einfaches Vorschlagsverfahren, basierend auf den Auftretenshäufigkeiten von Tags mit Benutzern und Ressourcen, fast so gut ist wie das beste Verfahren. Außerdem zeigen wir, wie wir solche Vorschlagsverfahren in ein kollaboratives Verschlagwortungssystem integriert haben, um ihre Performanz zu messen, indem wir das Tag-Vorschlags-Framework, welches wir für BibSonomy entwickelt haben, vorstellen. Da wir Vorschlagsverfahren entwickeln, testen, evaluieren und die Zusammenarbeit mit Forschern unterstützen wollen, haben wir das Framework so ausgelegt, dass es leicht erweiterbar, offen für eine große Auswahl von Verfahren, sowie unabhängig von BibSonomy benutzbar ist. Eine abschließende Evaluierung zeigt die Mächtigkeit des Frameworks.

Der Folksonomie-Graph hat strukturelle Eigenschaften, die sein Wachstum und die Möglichkeit der zufälligen Erkundung erklären. Logdateien von Suchmaschinen können als Folksonomie repräsentiert werden, indem Suchanfragen als Beschreibungen der besuchten Webseiten aufgefasst werden. Die sich dabei ergebende Struktur, die wir als *Logsonomie* bezeichnen, ist der von Folksonomien sehr ähnlich. Um ihre Eigenschaften zu untersuchen, analysieren wir die topologischen Besonderheiten des tripartiten Hypergraphen, der sich aus Suchanfragen, Benutzern und Webseiten ergibt, anhand der Logdateien zweier großer Suchmaschinen und vergleichen diese mit einem großen Folksonomie-Datensatz. Wir finden heraus, dass alle drei Datensätze ähnliche strukturelle Eigenschaften besitzen und schließen daher, dass das Klickverhalten von Suchmaschinenbenutzern von ähnlicher Dynamik wie das Taggingverhalten von Benutzern kollaborativer Verschlagwortungssysteme ist.

Wir übertragen in dieser Arbeit das Folksonomie-Paradigma auf den *Social Semantic Desktop* – eine neue Art von Computerdesktop, welcher Technologien des Semantic Web nutzt, um Informationen besser zu vernetzen. Dort wenden wir Methoden zur Unterstützung von Benutzergruppen auf die sich im Netzwerk verbundener Desktops enthaltene Folksonomie an. Damit verbinden wir Methoden der Wissensentdeckung für Folksonomien mit semantischen Technologien.

Zusammenfassend lässt sich sagen, dass die in dieser Arbeit durchgeführte Forschung sich mit kollaborativen Verschlagwortungssystemen und deren zugrundeliegender Datenstruktur – der Folksonomie – befasst und damit die Grundlagen für die weitere Erforschung dieses erfolgreichen Wissensmanagement-Paradigmas legt.

Overview of Author's Contribution

This thesis is the result of joint research with colleagues, in particular from the Knowledge and Data Engineering (KDE) Group at the University of Kassel. Therefore, it is inspired and influenced by various discussions with colleagues. The following list clarifies the original contribution of the author of this thesis and states which parts stem from collaborative work with others. All parts of this thesis which are not explicitly mentioned below or marked as citation on the spot are the sole work of the author.

Chapter 2: The formal model of a folksonomy in Section 2.3 is a result of discussions in the KDE working group.

Chapter 3: BibSonomy is developed by a team of the KDE group under supervision of Andreas Hotho. The design and implementation of a large fraction of the core functionality is the work of the author, in particular optimized database queries, posting processes, tag (relation) parsing, bibliographic hash keys, the Java version of FolkRank, and the first version of the web application. Furthermore, the author has influenced the design of most components and contributed around 25 % of the source code.¹

Chapter 4: The initial sketch for an algorithm for mining tri-concepts was provided by Bernhard Ganter. The author of this thesis extended it to mine *frequent* tri-concepts, performed the implementation and visualization and conducted the experiments.

Chapter 5: The tag recommendations for Collaborative Filtering were performed by Leandro Balby Marinho who also contributed the Last.fm dataset. The author of this thesis implemented the remaining algorithms, analyzed their computational complexity, and performed the evaluation – including the development of the *p*-core computation.

Chapter 6: The framework's core component, the multiplexer, as well as the remote recommender were implemented by Folke Eisterlehner who also organized the online part of the Discovery Challenge and contributed the plots to evaluate the influence of timeouts. The author designed the framework and implemented the remaining recommenders; he also developed and performed the evaluation.

¹According to StatCVS statistics from November 2nd, 2009.

Chapter 7: The Nepomuk project is a larger framework where the components *TagRecommender*, *CommunityManager*, and *FolkPeer* were contributed by the author. Other components referenced in that chapter and in particular the Nepomuk architecture are the joint work of the Nepomuk consortium.

Chapter 8: Based on the author's idea to regard search engine query logs as folksonomies, the analysis of the logsonomy graph is joint work with Beate Krause, based on work by Cattuto et al. (2007b). Chapter 8 presents the author's part of the analysis only.

Chapter 1

Introduction

Tagging – annotating resources with freely chosen keywords – recently has gained much popularity in the web and beyond. That is not surprising when one thinks about the fact that

“One of the greatest challenges facing people who use large information spaces is to remember and retrieve items that they have previously found and thought to be interesting.”
Millen et al., 2005

Tags fulfill this need in that they allow people to quickly annotate resources for later retrieval without putting too much overhead neither on the annotation nor retrieval process. Furthermore, in contrast to search engines which have revolutionized the web before, tagging does not rely on a textual representation of resources and thus is suitable for all kinds of items like videos, photos, audio streams, software, etc.

1.1 Topics of this Thesis

This thesis focuses on two areas of Knowledge Discovery in collaborative tagging systems: *Formal Concept Analysis* and *tag recommendations*. According to Fayyad et al. (1996), “Knowledge Discovery in Databases is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.” We here briefly relate the three topics – Collaborative Tagging Systems, Tag Recommendations, and Formal Concept Analysis – of this thesis’ title to the term Knowledge Discovery in Databases (KDD).

Collaborative Tagging Systems. These are typically web-based systems that allow their users to annotate resources with freely chosen keywords – so called tags. Once a user has registered and is logged in, he or she can save resources and assign tags to them. Resources can later be retrieved by searching for tags or clicking on them.

The datastructure of tags, users, and resources underlying collaborative tagging systems is called *folksonomy*. From a technical point of view, the systems can be seen as huge databases containing entries from thousands of users. To this end, the systems on the one hand produce and contain the data for the KDD process, on the other hand,

they are the subject that we intend to better understand, and finally, the results of our analysis shall flow back into the systems to improve their usability.

Tag Recommendations. When users annotate resources with tags they are typically supported by the collaborative tagging system that shows them some tags as recommendation. These recommendations 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.

The generation of tag recommendations is a typical scenario for an application of the KDD process. Based on data from various sources (e.g., the existing folksonomy, the content and metadata of resources) one tries to find tags which best describe the resource according to a certain quality criterion.

Formal Concept Analysis. Formal Concept Analysis (FCA) is a conceptual clustering technique that formalizes the concept of ‘concept’ as established in the international standard ISO 704: a concept is considered as a unit of thought constituted of two parts: its extension and its intension (Wille, 1982; Ganter and Wille, 1999).

The primary objective of FCA is to help the user to better understand the data and as such it supports one of the central goals of KDD. Furthermore, FCA provides the mathematical background for efficient data mining algorithms which can be part of the KDD process.

1.2 Contributions of this Thesis

This thesis contributes to the aforementioned research areas in the following way:

Collaborative Tagging Systems. We designed and implemented the collaborative tagging system BibSonomy as a tool for researchers to organize their bookmarks and publication references. We describe the architecture and features of the system and present first lessons we learnt from running the system over a period of four years. Furthermore, we use BibSonomy as an experimental platform to evaluate knowledge discovery methods.

Tag Recommendations. We compare different tag recommendation methods on three real world folksonomy datasets. We are able to provide for the first time quantitative evidence for the good performance of the FolkRank algorithm. We propose a new simple method for tag recommendations based on co-occurrence counts and prove its suitability – providing better recommendations than standard Collaborative Filtering methods. We show the transfer of our insights into a real application by implementing a tag

recommendation framework for the BibSonomy system and we present first evaluation results from the online system.

Formal Concept Analysis. We formalize the problem of mining all frequent tri-concepts in a triadic formal context and introduce the TRIAS algorithm for efficient computation of all frequent tri-concepts. We show the efficiency of the proposed algorithm and examine its outcome in a qualitative evaluation on three large real world datasets. Finally, we propose a new paradigm for lightweight visualization of tri-lattices.

Application of the Folksonomy Paradigm. We apply community support methods for folksonomies to the folksonomy found in the network of peers of the social semantic desktop. Thus, we connect data mining for folksonomies with semantic technologies. As a second application, we transfer the folksonomy paradigm to search engine query logs and compare the network properties of the resulting folksonomy network with that of a ‘standard’ folksonomy.

1.3 Structure of this Thesis

This thesis is divided into three parts along the topic of *applications of knowledge discovery in collaborative tagging systems*. We start by introducing collaborative tagging systems in Part I, focus on two areas of knowledge discovery in Part II, and present three applications in Part III.

The first part lays the groundwork for the following parts by introducing collaborative tagging systems and formalizing their underlying data structure (Chapter 2). An overview of the BibSonomy system – which functions as datasource for the knowledge discovery methods following in Part II as well as one of the applications for the integration and evaluation of results in Part III – is given in Chapter 3.

Two aspects of collaborative tagging systems are investigated in Part II: analyzing the conceptual structure of folksonomies (Chapter 4) and recommending tags in folksonomies (Chapter 5). In Chapter 4, we present triadic Formal Concept Analysis as unsupervised knowledge discovery method for shared conceptualizations in folksonomies. Since in this setting a gold standard is not available, we perform a qualitative evaluation on several datasets. In contrast, Chapter 5 presents tag recommendations as supervised machine learning scenario. A quantitative evaluation compares several methods against the decision of users from three collaborative tagging systems.

A wide range of applications related to collaborative tagging systems is described in Part III. Building upon the research conducted in Chapter 5, we describe, in Chapter 6, the implementation and integration of a tag recommendation framework for BibSonomy. In Chapter 7 we transfer community support methods developed for collaborative tagging systems to the folksonomy found on the social semantic desktop Nepomuk. The part concludes with an adaptation of the folksonomy idea to search engine query logs in Chapter 8 and an outlook on compelling future challenges in Chapter 9.

Part I

Collaborative Tagging Systems

Collaborative tagging systems allow users to share resources and annotate them with freely chosen keywords – so called tags.

In this part we introduce collaborative tagging systems, their underlying datastructure – folksonomies, and give a brief overview on scientific work dealing with such systems.

Furthermore, we present the architecture and features of BibSonomy as an exemplary social bookmarking system that will accompany us throughout this thesis.

Chapter 2

Foundations and State of the Art

The rapid evolvement of the World Wide Web into the ‘Web 2.0’ named diversity of new technologies and services in the last years has brought fascinating new possibilities to web users. One can now easily find encyclopedic articles to very diverse topics, edit and enhance them with one’s own knowledge (*Wikipedia*¹). Thousands of people publish and annotate their photos (*Flickr*²) – one can comment, (often) re-use their works for presentations, artworks, or postcards and add own photos. Similarly, one can arrange a custom ‘tv program’ using short video clips recorded by people all over the world (*YouTube*³) – or, as is more often the case, trail away in the mass of funny clips, homemade music videos, documentations, or true works of art. Saving and sharing links to interesting web sites and discovering which pages users with similar interests found interesting (*Delicious*⁴) never has been easier – with the added value of world wide accessibility of one’s web bookmarks. News from events all over the world are spread in seconds by users of micro blogging platforms (e. g., *Twitter*⁵), as well as experiences from their daily life or snippets of conversation from political discussions or conferences.

Technologies like blogs, wikis, collaborative tagging, RSS feeds, or AJAX are the cornerstone of this development, but the most important characteristic common to all mentioned services is the diminishing distinction between content provider and content consumer. The same people which read the articles in Wikipedia edit and write them. In particular, the principle of *tagging*, i. e., the annotation of resources with freely chosen keywords – so called *tags*, has gained wide popularity. First introduced by *collaborative tagging systems*, it soon became very popular and rapidly spread to other systems. Now it is very common to tag blog posts (e. g., Technorati⁶), news articles (e. g., Slashdot⁷), or software (e. g., Freshmeat⁸). Aside from the web, software packages of the Linux distribution Debian⁹ can now be annotated with keywords, as well as local web bookmarks

¹<http://www.wikipedia.org/>

²<http://www.flickr.com/>

³<http://www.youtube.com/>

⁴<http://www.delicious.com/>

⁵<http://twitter.com/>

⁶<http://technorati.com/>

⁷<http://slashdot.org/>

⁸<http://freshmeat.net/>

⁹<http://www.debian.org/>

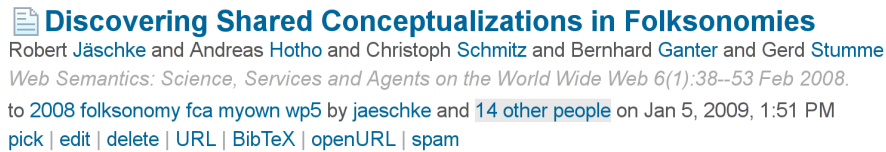


Figure 2.1: A screenshot of a publication reference post in BibSonomy. Below the title and the bibliographic metadata (authors, journal, year, etc.) the tags *2008*, *folksonomy*, *fca*, *myown*, and *wp5*, which user *jaeschke* assigned to this post, are shown. Further, the posting date and the number of other users having stored that resource are depicted.

in the web browser Firefox.¹⁰ This considerably expands the application range of the methods discussed in this work.

2.1 Collaborative Tagging Systems

Collaborative tagging systems allow users to share arbitrary resources in the World Wide Web and to annotate them with freely chosen keywords, so called *tags*. The resources together with the tags are stored on a central server and can be accessed from any computer connected to the web. Although actually meant to designate systems which allow to share web links – *bookmarks* – only, the term *social bookmarking system* often is used interchangeably for such systems.

What type of resource can be saved depends on the particular system. There are services to store photos (Flickr), web links (Delicious), music preferences (Last.fm¹¹) or even goals in life (43 Things¹²). The *BibSonomy* system,¹³ developed by the Knowledge and Data Engineering group of the University of Kassel,¹⁴ supports sharing of both web links (URLs) and publication references. Figure 2.1 shows a screenshot of a publication reference posted to BibSonomy.

With tags, the users can describe in a simple way resources, express their opinions about them, or characterize their importance, origin, usability, etc. (Golder and Huberman, 2005). The tags later facilitate retrieval of resources, navigation in the system, and also allow for serendipitous browsing and discovery of new interesting resources from other users, since in most systems the resources are publicly visible for everybody. Thus, tags provide an immediate benefit to the user. Another advantage of tags over hierarchically organized categories (e.g., folders in a file system) is the multitude of

¹⁰<http://www.mozilla.com/firefox/>

¹¹<http://www.last.fm/>

¹²<http://www.43things.com/>

¹³<http://www.bibsonomy.org/>

¹⁴<http://www.kde.cs.uni-kassel.de/>

implicitly form communities of interest. Making those communities explicit further allows users to discover other users with similar interests. The systems often support this process by listing posts and tags sorted by their popularity. Thus, one can easily see the popular resources for a given topic. Additionally, most systems provide social features like groups, friends or an inbox, which further strengthen interaction and cooperation between users. *Groups* allow users to form explicit communities in which they can share their resources. On the one hand, users can restrict the access to posts to members of a group, on the other hand a group provides an aggregated view of the posts of its members. Moreover, often a user can add other users to his list of *friends*. This allows him to share posts only among his friends and thus keep them informed about links he finds interesting for them. The network built by this friendship links could be further explored, e.g., for computing trust or improving recommendations. In some systems users can ‘send’ interesting posts directly to other users by attaching a special tag. Those posts then appear in the other users’ *inbox* and thereby allow them to store the posts if they find them interesting. Typically, this operation is restricted to friends or group members to anticipate abuse.

2.2 Folksonomies

In their core, collaborative tagging systems are all very similar. Once a user is logged in, he can add a resource to the system, and assign arbitrary tags to it. The collection of all his assignments is his *personomy*, the collection of all personomies constitutes the *folksonomy*. The user can explore his personomy, as well as the personomies of the other users, in all dimensions: for a given user one can see all resources he has uploaded, together with the tags he has assigned to them; when clicking on a resource one sees which other users have uploaded this resource and how they tagged it; and when clicking on a tag one sees who assigned it to which resources.

The three-dimensional data structure of *users*, *resources*, and *tags* underlying collaborative tagging systems is called *folksonomy* – a blend of the words *folk* and *taxonomy* which stands for conceptual structures created by the people (Vander Wal, 2007). Folksonomies are thus a bottom-up complement to more formalized Semantic Web technologies, as they rely on *emergent semantics* (Steels, 1998; Staab et al., 2002; Cattuto et al., 2007b) which result from the converging use of the same vocabulary. The main difference to ‘classical’ ontology engineering approaches is their aim to respect to the largest possible extent the request of non-expert users not to be bothered with any formal modeling overhead. Thomas Vander Wal, who coined the term folksonomy, writes (Vander Wal, 2007):

“Folksonomy is the result of personal free tagging of information and objects (anything with a URL) for one’s own retrieval. The tagging is done in a social environment (usually shared and open to others). Folksonomy is created from



Figure 2.3: A schematic view of a folksonomy, illustrating the three dimensions of users (top), resources (left), and tags (right). Artwork by Dennis Kohlmetz based on author’s sketch.

the act of tagging by the person consuming the information.”

Accordingly, a folksonomy describes the annotation of resources by users with keywords, as outlined in Figure 2.3. Although often used synonymously, a folksonomy denotes the data structure build in a collaborative tagging system, rather than the system itself. Before we formally define this structure, we distinguish its different manifestations.

To this respect we must first differentiate between ‘broad’ and ‘narrow’ folksonomies (Vander Wal, 2005). While broad folksonomies, like in BibSonomy, allow several users to tag the same resource, in narrow folksonomies – the most popular example is YouTube – only the user who created or uploaded the resource can assign tags to it. As a consequence, there exist only links between users when they have used the same tags. Some authors also tend to call only broad folksonomy based systems ‘social’ or ‘collaborative’. To that effect, in this work we will essentially focus on broad folksonomies and corresponding systems.

Compared to ontologies (Gruber, 1993), folksonomies constitute a rather weak semantic structure. Besides a sub-/supertag relation available in some systems (cf. the next section) which allows users to build a tag hierarchy, there are no explicit relations between tags. One can not link plural with singular forms of words or distinguish homonyms or polysemes. The ambiguity of tags is the main drawback of the open vocabulary of folksonomies and is partially addressed by the deployment of tag recommendation methods (cf. Chapter 5).

Beyond collaborative tagging systems, Folksonomy-like structures can be found in various other applications. Tagging file systems (Bloehdorn et al., 2006) augment the directory tree structure of classical file systems with the possibility to assign tags to files. The tags then can be used to search files or for navigation. Since a file can have multiple tags, then several paths to a file exist in contrast to the single path in a directory based file system. Queries issued to search engines can also be regarded as folksonomy (Krause et al., 2008a). The terms of a query then play the role of tags, which the user implicitly adds to resources by clicking on the links in the result list. In Chapter 8 this structure – which we entitled *logsonomy* – is compared to folksonomies.

2.3 A Formal Model

A folksonomy \mathbb{F} describes the users U , resources R , and tags T , and the user-based assignment of tags to resources by the ternary relation $Y \subseteq U \times T \times R$. Our model of a folksonomy was introduced in (Hotho et al., 2006b) and is also the core of the data model of the BibSonomy system described in Chapter 3.

Definition 2.1 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., and
- 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).
- \prec is a user-specific subtag/supertag-relation, i. e., $\prec \subseteq U \times T \times T$, called subtag/-supertag relation.

Definition 2.2 The personomy \mathbb{P}_u of a given user $u \in U$ is the restriction of \mathbb{F} to u , i. e., $\mathbb{P}_u := (T_u, R_u, I_u, \prec_u)$ with $I_u := \{(t, r) \in T \times R \mid (u, t, r) \in Y\}$, $T_u := \pi_1(I_u)$, $R_u := \pi_2(I_u)$, and $\prec_u := \{(t_1, t_2) \in T \times T \mid (u, t_1, t_2) \in \prec\}$, where π_i denotes the projection on the i th dimension.

Users are typically described by their user ID, and tags may be arbitrary strings. What is considered a resource depends on the type of system. For instance, in Delicious, the resources are URLs, in BibSonomy URLs or publication references, and in

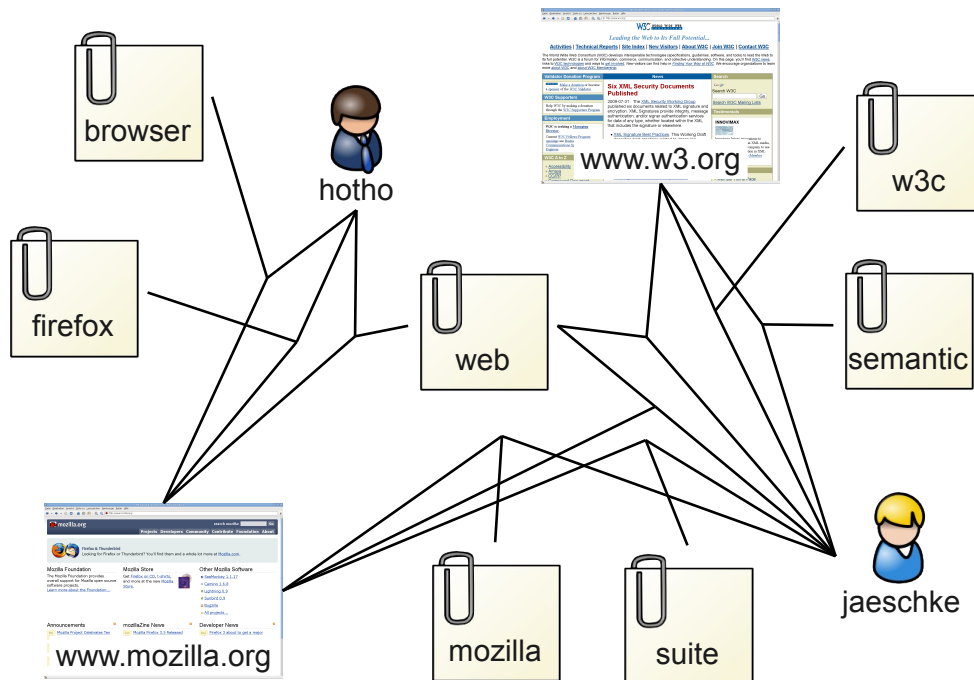


Figure 2.4: Excerpt of a folksonomy hypergraph showing the three posts (hotho, {web, browser, firefox}, <http://www.mozilla.org/>), (jaeschke, {w3c, semantic, web}, <http://www.w3.org/>), (jaeschke, {web, suite, mozilla}, <http://www.mozilla.org/>)

Last.fm, the resources represent artists. Typically, resources can be referenced using URLs, independent of their type.

In Delicious a variant of the subtag/supertag relation is implemented as *bundles*; BibSonomy's implementation is described in Section 3.5.5. Often we do not make use of the relation for sake of simplicity. I. e., we set $\prec := \emptyset$, and we will simply regard a folksonomy as a quadruple $\mathbb{F} := (U, T, R, Y)$. This structure is known in Formal Concept Analysis (Ganter and Wille, 1999) as a *triadic formal context* (Lehmann and Wille, 1995; Stumme, 2005). Another perspective of this structure is that of a tripartite, undirected hypergraph $G = (V, E)$, where $V = U \dot{\cup} T \dot{\cup} R$ is the disjoint union of the sets of users, tags and, resources, and every hyperedge $(u, t, r) \in E$ connects exactly one tag, one user, and one resource. An exemplary visualization of such a hypergraph can be seen in Figure 2.4.

For convenience, we also define, for all $u \in U$ and $r \in R$, $T(u, r) := \{t \in T \mid (u, t, r) \in Y\}$, i. e., $T(u, r)$ is the set of all tags that user u has assigned to resource r . The set of all *posts* of the folksonomy is then $P := \{(u, S, r) \mid u \in U, r \in R, S = T(u, r), S \neq \emptyset\}$. Thus, each *post* consists of a user, a resource and all tags that this user has assigned to that resource.

2.4 Related Work

While the scientific community has only begun to explore folksonomies as a knowledge representation mechanism as well as a source of data which can be mined for different purposes, there is a growing number of publications concerned with the various aspects of this new phenomenon. Overviews of social bookmarking tools with special emphasis on folksonomies are provided by (Hammond et al., 2005) and (Lund et al., 2005), as well as (Mathes, 2004) and (Speller, 2007) who discuss strengths and limitations of folksonomies. Other authors focus on analyzing and visualizing the structure of folksonomies, e. g., (Dubinko et al., 2006) or (Golder and Huberman, 2005) which identified seven types of tags. Since tagging has been widely adopted by other kinds of systems, there is further work investigating aspects of tagging in blogs, e. g., (Brooks and Montanez, 2005), wikis, e. g., (Voss, 2006), or e-learning, e. g., (Bateman et al., 2007). The knowledge discovery, information retrieval, and knowledge engineering communities are currently becoming involved in this development, e. g., by enhancing recommendations given by the systems, improving search and ranking, and structuring the knowledge in a systematic way.

Cattuto et al. (2006) investigate statistical properties of tagging systems and introduce a stochastic model of user behaviour; Halpin et al. (2006) analyze the dynamics and semantics of tagging systems, and Lambiotte and Ausloos (2005) introduce further techniques to structure the tripartite network of folksonomies. Recently, work on more specialized topics such as structure mining on folksonomies – e. g., to visualize trends (Dubinko et al., 2006) has been presented. Steels (2006) considers – looking from a psychological perspective – collaborative tagging as distributed cognition. More practical questions, e. g., if (Heymann et al., 2008a) and how (Yahia et al., 2008) social bookmarking can improve keyword based (web) search, have come up more recently.

Finally, the popularity of tagging caused interest in companies, e. g., at IBM (Millen et al., 2005), which expect better linkage and accessibility of intranet resources. Social bookmarking systems (as well as wikis) are seen as a tool to overcome the knowledge acquisition bottleneck (Hayes-Roth et al., 1983) which has been a problem in many earlier knowledge management systems.

Chapter 3

BibSonomy

To give an example of a collaborative tagging system, this chapter describes the web-based social bookmark and publication sharing system BibSonomy developed by our research group. BibSonomy is one of the folksonomies analyzed in Chapters 4 and 5 and is also home of the tag recommendation framework described in Chapter 6. After an introduction to the user interface, architecture, and features of BibSonomy, we describe ongoing work.

3.1 Introduction

BibSonomy¹ (Hotho et al., 2006a) started as a students project at the Knowledge and Data Engineering Group of the University of Kassel² in spring 2005. We had a three-fold motivation to develop BibSonomy: First, we as researchers needed a tool to organize our publication references; second, interested in the (at that time) upcoming and growing collaborative tagging systems we were looking for a data source to analyze such systems; and finally, we wanted to have a playground to implement and test new ideas in a running system. The goal of the project thus was to implement a system for organizing BIB_TE_X³ entries in a way similar to bookmarks in Delicious – which was at that time becoming more and more popular. We soon decided to integrate bookmarks as a second type of resource into the system. After the student finished the project, we continued development and made huge efforts to tune the SQL queries and to test and improve the usability of the system. Upon the progress made, BibSonomy was opened for public access at the end of 2005 – first announced to colleagues only, later in 2006 to the public.

Since then, the number of users has steadily grown (see Figure 3.1). We implemented several useful features (cf. Section 3.5), we redesigned the architecture to ease future developments (Section 3.4), and we and other researchers used BibSonomy or its data for research (cf. Sections 3.2, 4.4.3, and 5.6.2). Having the opportunity to develop and run BibSonomy, we also used that gift to integrate our research results into the system

¹<http://www.bibsonomy.org/>

²<http://www.kde.cs.uni-kassel.de/>

³BIB_TE_X (Patashnik, 1988) is a popular literature management system for L^AT_EX (Lamport, 1986), which many researchers use for writing scientific articles.

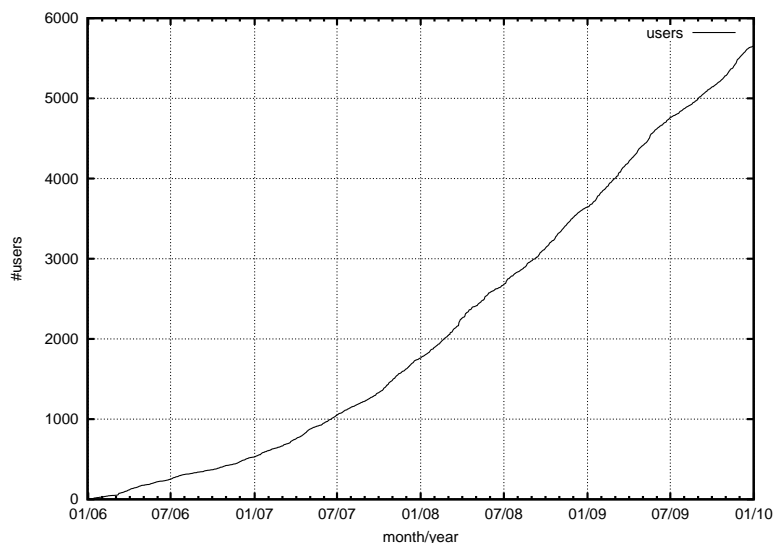


Figure 3.1: The growth of the number of users in BibSonomy which have at least one post and are not flagged as spammer.

(see Section 3.5.9 and Chapter 6) – both for the benefit of the users and to directly measure the performance of our methods.

3.2 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. Until the end of 2009 more than three million users⁴ have contributed bookmarks to Delicious and thereby probably make it one of the largest human-annotated collection of web links.

CiteULike⁵ 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,⁶ operated by the Nature Publishing Group. In its size it is comparable to BibSonomy.

⁴This estimate is based on the author's regular crawling activity. On September 25th 2006, Joshua Schachter announced to have reached the one million mark (Schachter, 2006).

⁵<http://www.citeulike.com/>

⁶<http://www.connotea.org/>

Since its start in 2006, the number of scientific articles referencing BibSonomy or using its data has steadily grown. Here we give a brief overview on some of the works not written by our group.

In (Regulski, 2007) the usability of BibSonomy as a tool for searching and managing scientific literature is discussed. The author describes how it can be used as a supplement to conventional reference databases and compares BibSonomy with Connotea and CiteULike. In particular, the author discusses how BibSonomy can be used as a platform for dissemination of online journal articles. An extensive user study comparing BibSonomy and CiteULike has been organized by the University Library of Amsterdam (van der Graaf, 2007). It started with a discussion round collecting the expectations of the participants, asked them to fill out questionnaires, and also evaluated logbooks which contained records of the user's interaction with the systems. At that time, participants favoured CiteULike over BibSonomy, in particular, since the needed functionality in BibSonomy often was rather hidden.

An analysis of the tagging activity to discover usage patterns in BibSonomy and CiteULike can be found in (Santos-Neto et al., 2007). The authors further evaluate user similarity to find communities of users with particular interests. Therefore they build an *interest-sharing graph* based on different definitions of user interest: *user-item*, *user-tag*, and *directed user-item* similarity. For those measures they investigate the connected components of the graph. As one result, they discover “a significant number of small sub-communities of interests totally separated from each other”.

The evolution of several social bookmarking systems has resulted in non-integrated heterogeneous tag spaces. Oldenburg et al. (2008) analyze the similarity of those tag spaces by comparing the tags over one month from four different services. Their goal is a virtual tag space that provides the user a uniform view over several tagging systems.

Cattuto et al. (2009) analyze the tag-tag-co-occurrence network of BibSonomy and Delicious and build a framework to model social annotation. Based on the assumption that annotating a post with tags is a random walk in a ‘semantic space’, the generated networks are able to reproduce the complex evolution and structure of the empirical data. This is proven by the comparison of several topological and statistical network properties.

Zheleva and Getoor (2009) investigate how publicly available information of users in folksonomies, like group membership and friendship links, allow to infer sensitive private attributes of users' profiles (e. g., age, gender, religion). They regard this as a relational classification task and evaluate several methods on four large datasets. On the BibSonomy data they are able to predict if a user is a spammer or not, using solely the common tags of users as information.

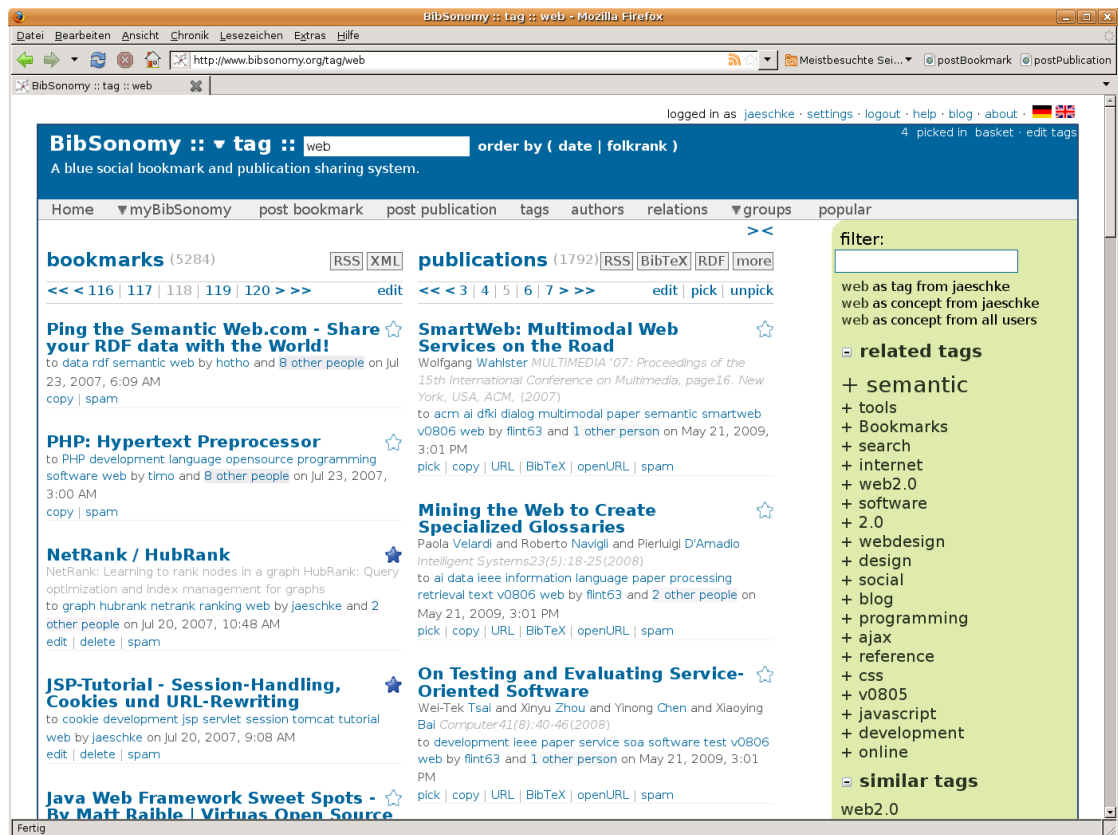


Figure 3.2: A screenshot of the BibSonomy page <http://www.bibsonomy.org/tag/web>. It simultaneously shows bookmarks and BibTeX-based bibliographic references tagged with the keyword *web*.

3.3 User Interface

A typical list of posts is depicted in Figure 3.2 which shows bookmark and publication posts containing the tag *web*. The page is divided into four parts: the header (showing information such as the current page and path, navigation links and a search box), two lists of posts – one for bookmarks and one for publications – each sorted by date in descending order, and a list of tags related to the posts. This scheme holds for all pages showing posts and allows for navigation in all dimensions of the folksonomy. The semantics of those pages is explained in Section 3.4.3. The posts in the lists can also be sorted by FolkRank (see Section 3.5.9), while the tags can be sorted lexicographically or by frequency of usage, depending on the user’s choice.

A detailed view of one bookmark post from the list in Figure 3.2 can be seen in



Figure 3.3: Detailed screenshots of a bookmark and a publication post in BibSonomy.

Figure 3.3(a). The first line shows in bold the title of the bookmark (*NetRank/HubRank*) which has the URL of the web page as underlying hyperlink. On the right side the blue star (★) indicates that the user who is logged in to the system owns this post. A click on the star allows the user to edit the post. The next two lines show an optional description the user can assign to every post. The next two lines belong together and show detailed information: first, all the tags the user has assigned to this post (*graph*, *hubrank*, *netrank*, *ranking* and *web*), second, the user name of that user (*jaeschke*), followed by a note, how many users tagged that specific resource (*2 other people*). These parts have underlying hyperlinks, leading to the corresponding tag pages of the user (*/user/jaeschke/graph*, */user/jaeschke/web*, ...), the users page (*/user/jaeschke*) and a page showing all three posts (i.e., the one of user *jaeschke* and those of the two other people) of this resource (*/url/r*). Section 3.4.3 explains the paths given in brackets further. The last part shows the posting date and time, followed by a line of links for actions the user can do with this post – depending on if this is his own (*edit*, *delete*) or another user’s post (*copy*). The concluding *spam* link allows users to mark posts as spam – those suggestions can be regarded by BibSonomy’s spam framework (Krause et al., 2008b).

The structure of a publication post displayed in BibSonomy is very similar, as seen in Figure 3.3(b). The first two lines show again the title of the post, which equals the title of the publication in $\text{BIB}_{\text{T}}\text{X}$. It has an underlying link leading to a page which shows detailed information on that post. Here, the empty blue star (☆) indicates that the logged in user does not own this post. A click on the icon allows him or her to copy the post. This is followed by the authors or editors of the publication, as well as journal or book title and the year. The next lines show the tags assigned to this post by the user, whose user name comes next, followed by a note how many people tagged this publication. As described for bookmark posts, these parts link to the respective pages. After the date and time the user posted this entry follow the actions the user can do, which in this case include picking the post for later download (cf. Section 3.5.7), copying it, accessing the URL of the resource, viewing the $\text{BIB}_{\text{T}}\text{X}$ source code, looking

for a copy of the article in one's local library using OpenURL,⁷ and suggesting the post as spam.

3.4 Architecture

BibSonomy is a web application, written in Java, which is running in an Apache Tomcat⁸ servlet container. It is based on Java Server Pages⁹ and Java Servlet¹⁰ technology and uses a MySQL¹¹ database as backend (see Section 3.4.2). Several components (e. g., one encapsulating database access, one containing all scrapers, . . .) which are described briefly in the following section provide the building blocks of the system.

Currently, the project has several hundred thousand lines of code (cf. Figure 3.4) and is using the Model View Controller (MVC) (Krasner and Pope, 1988) programming paradigm to separate the logical handling of data from the presentation of the data. This enables us to produce output in various formats (see Section 3.5.2), since adding a new output format is accomplished by implementing a new view for the model.

3.4.1 Components

BibSonomy is built up of several *components* (or *modules*) which encapsulate different functionalities of the system. We here briefly describe the main components of the system which are depicted in Figure 3.5. 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 BIB_TE_X strings and files into Java objects and thus is involved in every incoming publication post request. This component basically encapsulates the *javabib* parser written by Johannes Henkel.¹²

Common* Most enums, exceptions, and utility functions 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 classes of the model, e. g.,

⁷OpenURL is a standard which (among other things) allows to resolve meta information about publications into the user's local library. Further information can be found at <http://www.oclc.org/research/projects/openurl/default.htm>.

⁸<http://tomcat.apache.org/>

⁹<http://java.sun.com/products/jsp>

¹⁰<http://java.sun.com/products/servlets>

¹¹<http://www.mysql.com/>

¹²<http://www.plan.cs.colorado.edu/henkel/stuff/javabib/>

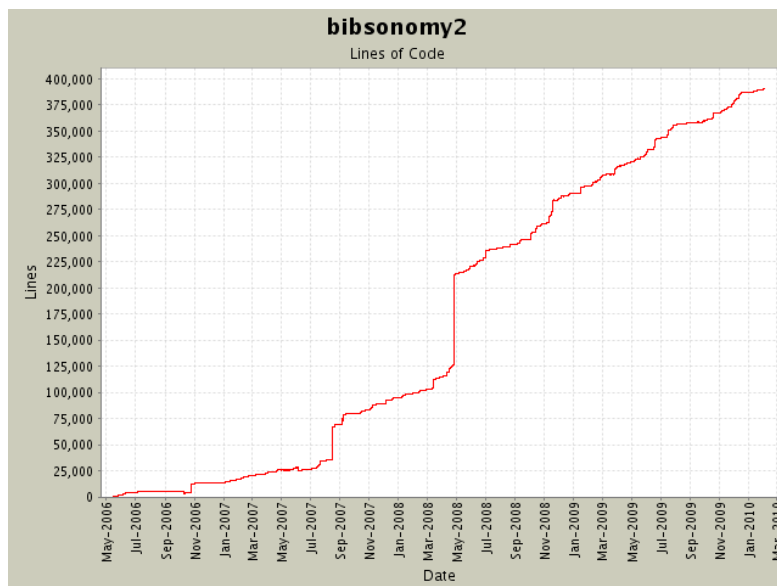


Figure 3.4: The growth of the number of lines of code in the BibSonomy project. The two larger steps show bulk commits. In August 2007 the code of the original implementation was merged into the new project and in April 2008 the training data for the Conditional Random Field of the information extraction scraper (cf. Section 3.5.1) was uploaded. Ignoring the latter, the project contains around 300,000 lines of code – including comments, stylesheets, JavaScript code, and other files.

storePost, *updateUser*, *getGroup*, ... Furthermore, this component contains the request chains which provide lists of posts for each of the URLs described in Section 3.4.3. As backend functions a relational database (cf. Section 3.4.2).

Layout* Implementing the *LayoutInterface*, this component provides rendering of publication posts using export filters from JabRef. For details see Section 3.5.2.

LuceneLogic As a complement of the *DatabaseLogic*, this component provides full-text search over posts (e.g., title, tags, authors, etc.) using Apache Lucene.¹³

Model* The model maps the central elements *Tag*, *User*, *Resource*, and *Post* of the formal folksonomy model introduced in Section 2.3 onto Java classes and thus defines the data model BibSonomy is based on. In particular, all objects one operates on using the *LogicInterface* are defined by these classes. Figure 3.6 gives an overview on the core classes of the model, their attributes and associations. Almost all

¹³<http://lucene.apache.org/>

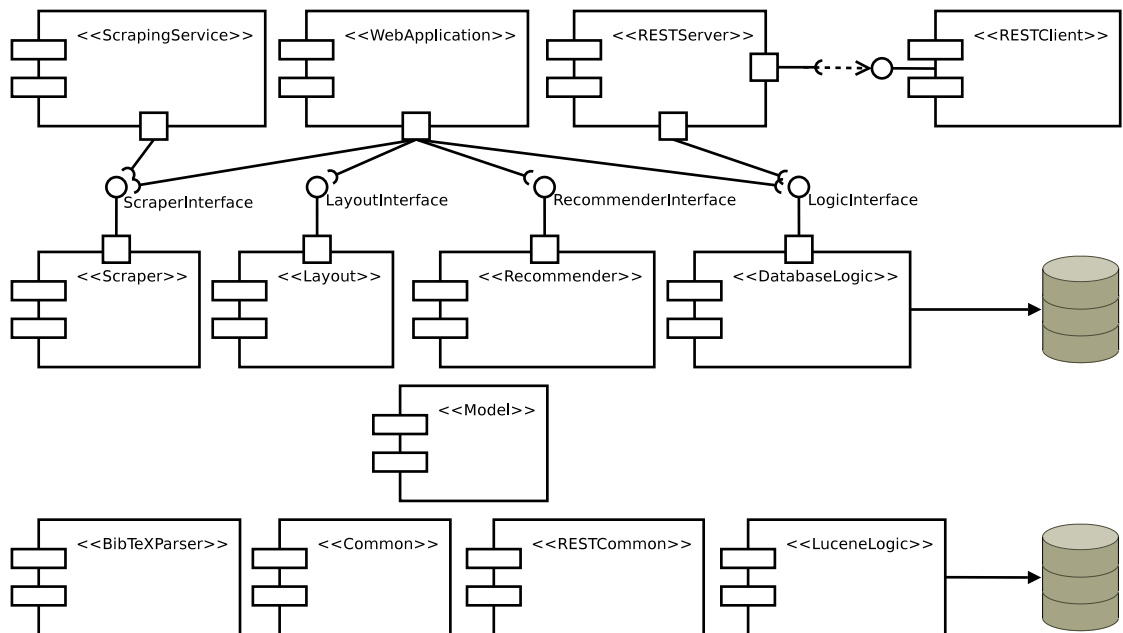


Figure 3.5: The UML component diagram of BibSonomy’s components. All components build upon the *Model* which contains the classes that represent BibSonomy’s data model. The *ScrapingService*, *WebApplication*, and *RESTServer* are the connection to the web.

attributes which can be modified by the user (e.g., *title*, *url*, ...) are modeled as strings to give the user as much freedom as possible in entering values. An important part of the model is the definition of the XML Schema of BibSonomy’s REST API (see Section 3.5.4) which closely resembles the model described here, omitting some attributes necessary for internal use only (e.g., *passwordHash* or *contentId*).

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 6.

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 enums, exceptions, and the XML renderer.

RESTServer* The REST server of BibSonomy offers REST-based access to the *LogicInterface* using XML over HTTP (see Section 3.5.4 for details).

Scraper* More than 60 screen scrapers which allow to extract publication metadata from various digital libraries – see Section 3.5.1. They are assembled into a chain of responsibility which also implements the *ScraperInterface*.

ScrapingService* A stand alone 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,¹⁴ there are controllers for each of the different pages of the web application which are coupled to views by Spring. Typically, each incoming HTTP request is handled by a controller which queries the *DatabaseLogic* for data and subsequently forwards the retrieved results to a view – like a JSP or the *Layout* component – responsible for rendering the output page.

The component structure and in particular the design of the *LogicInterface* and the REST API is based on work of Bork (2006). Since then it has been enhanced by extension of the central components *DatabaseLogic* and *Model* and addition of new components like *Layout* and *Recommender*.

3.4.2 Database

As backend for the *DatabaseLogic* we use a MySQL¹⁵ relational database. The iBATIS¹⁶ data mapper framework is used as an intermediate layer between the *DatabaseLogic* and MySQL. It couples SQL statements (embedded in XML) with the classes of the model and thereby separates the Java code from the SQL code.

The database schema of BibSonomy 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 information regarding *users* and *groups*. In Figure 3.7, the two *posts* tables 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 are very similar – the publication post table has just some additional columns to hold all the $\text{BIB}_{\text{T}}\text{E}_{\text{X}}$ fields. They are separated in the database for efficiency reasons, since these extra columns just need to be stored for publications.

¹⁴<http://www.springsource.org/>

¹⁵<http://www.mysql.com/>

¹⁶<http://ibatis.apache.org/>

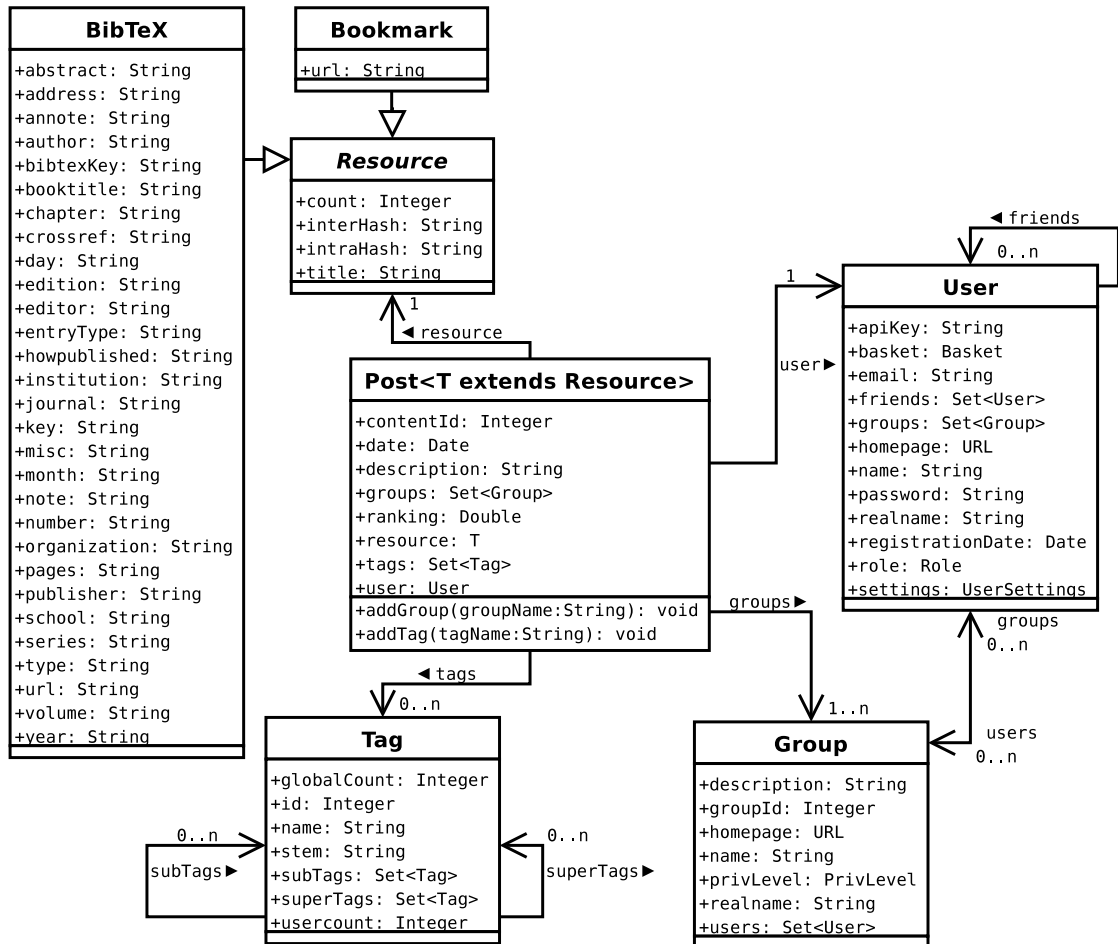


Figure 3.6: UML diagram of BibSonomy's data model.

The *posts* table is connected with the *tags* 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 it in the *tags* table to minimize the rows touched when selecting rows for the various views. Furthermore, several other tables hold counters (i. e., how many people share one resource, how often a tag is used, ...). Finally, several indexes (12 in the *tags* table alone) build the basis for fast answering of queries.

Overall, we spent a significant amount of time investigating and optimizing SQL queries and table schemes and tested both with folksonomy data of up to 8,000,000 posts. At the moment, we need no special caching or physical distribution of the database to get

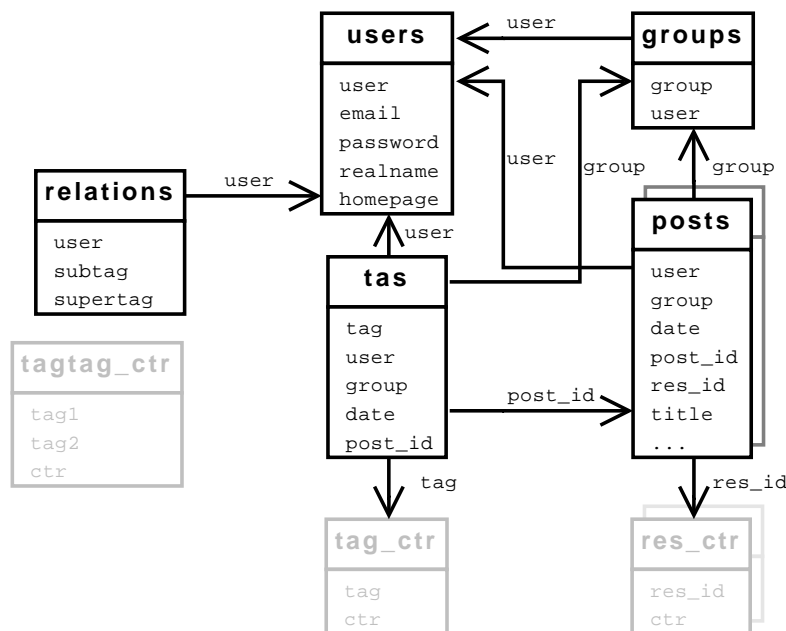


Figure 3.7: Relational schema of the most important tables.

reasonable response times, although 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 accesses (and replicates it to the slave), as well as all non-search engine read accesses; the slave handles a part of the search engine read accesses as well as all reads necessary for offline computations (e. g., to create tag counts, spam predictions, or fulltext search indexes).

3.4.3 Semantics of the URL Scheme

Navigation in BibSonomy's underlying folksonomy is accomplished in the web interface by following links pointing to different pages. Each of the pages shows posts with certain properties, e. g., the `/tag/t` page shows all posts which contain the tag `t`. Here, we describe all pages showing (bookmark and/or publication) posts but omit system pages which are necessary for the usage of BibSonomy like `/help`, `/settings` or `/post_bookmark`, since their semantic is straightforward. All URLs are relative to `http://www.bibsonomy.org/`, i. e., only the path part is given. Since group visibility rights (see Section 3.5.3) make the explanation much more complicated, they are mostly disregarded in this section, as well as in the formal model. The following list describes the contents C of all pages which show posts in BibSonomy:

`/tag/ $t_1 \dots t_n$` Shows every post which has all of the tags t_1, \dots, t_n attached:

$$C_{t_1, \dots, t_n} := \{(u, S, r) \in P \mid \{t_1, \dots, t_n\} \subseteq S\} \quad (3.1)$$

`/user/ u` Shows all posts of user u :

$$C_u := \{(\hat{u}, S, r) \in P \mid \hat{u} = u\} \quad (3.2)$$

`/user/ u / $t_1 \dots t_n$` Shows every post of user u which has all of the tags t_1, \dots, t_n attached:

$$C_{u, t_1, \dots, t_n} := \{(\hat{u}, S, r) \in P \mid \hat{u} = u, \{t_1, \dots, t_n\} \subseteq S\} \quad (3.3)$$

`/concept/ $t_1 \dots t_n$` Shows every post (u, S, r) which has for every tag $t \in \{t_1, \dots, t_n\}$ at least one of its subtags in $\prec \cap (\{u\} \times T \times T)$ or t itself attached (see also Section 3.5.5):

$$C_{t_1, \dots, t_n} := \{(u, S, r) \in P \mid \forall t_i (i = 1, \dots, n) \exists t \in S : (u, t, t_i) \in \prec \vee t = t_i\} \quad (3.4)$$

`/concept/user/ u / $t_1 \dots t_n$` Shows every post of user u which has for every tag $t \in \{t_1, \dots, t_n\}$ at least one of its subtags or t itself attached:

$$C_{u, t_1, \dots, t_n} := \{(\hat{u}, S, r) \in P \mid \hat{u} = u, \forall t_i (i = 1, \dots, n) \exists t \in S : (\hat{u}, t, t_i) \in \prec \vee t = t_i\} \quad (3.5)$$

`/concept/group/ g / $t_1 \dots t_n$` Shows every post which has for every tag $t \in \{t_1, \dots, t_n\}$ 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 :¹⁷

$$C_{g, t_1, \dots, t_n} := \{(u, S, r) \in P \mid u \in g, \forall t_i (i = 1, \dots, n) \exists t \in S : (g, t, t_i) \in \prec \vee t = t_i\} \quad (3.6)$$

`/url/ r` If r is a bookmark: Shows all posts of the resource r :

$$C_r := \{(u, S, \hat{r}) \in P \mid \hat{r} = r\} \quad (3.7)$$

`/url/ r / u` If r is a bookmark: Shows the post of user u of the resource r :

$$C_{r, u} := \{(\hat{u}, S, \hat{r}) \in P \mid \hat{r} = r, \hat{u} = u\} \quad (3.8)$$

¹⁷Each group in BibSonomy is represented as a user and thus can participate in the \prec relation.

`/bibtex/r` If r is a literature reference: Shows all posts of the resource r :

$$C_r := \{(u, S, \hat{r}) \in P \mid \hat{r} = r\} \quad (3.9)$$

`/bibtex/r/u` If r is a literature reference: Shows the post of user u of the resource r :

$$C_{r,u} := \{(\hat{u}, S, \hat{r}) \in P \mid \hat{r} = r, \hat{u} = u\} \quad (3.10)$$

`/group/g` Shows all posts of all users belonging to the group¹⁸ g :

$$C_g := \{(u, S, r) \in P \mid u \in g\} \quad (3.11)$$

`/group/g/t1...tn` Shows every post which has all of the tags t_1, \dots, t_n attached and where the user belongs to group g :

$$C_{g,t_1,\dots,t_n} := \{(u, S, r) \in P \mid u \in g, \{t_1, \dots, t_n\} \subseteq S\} \quad (3.12)$$

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

`/viewable/g/t1...tn` Shows all posts which are set viewable for members of the group g and which have all of the tags t_1, \dots, t_n attached.

`/search/s` Shows all resources, whose metadata matches the search expression s , which is interpreted by the LuceneLogic's full-text search capability.

`/basket` Shows all publication posts which the user has picked in his basket, as described in Section 3.5.7.

`/popular` Shows the three most often posted resources of the last 7, 30, and 120 days. (Note that these numbers are subject to change.)

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

`/author/a1...an` Shows all publication posts whose author or editor field contains all of the names a_1, \dots, a_n .

`/bibtexkey/k` Shows all publication posts which have the $\text{BIB}_{\text{T}}\text{E}_\text{X}$ key k .

An interesting feature, described in Section 3.5.2, 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 3.2), but this feature allows the user to get her output in formats like $\text{BIB}_{\text{T}}\text{E}_\text{X}$ or as an RSS feed.

¹⁸More on groups can be found in Section 3.5.3

3.5 Features

In this section we highlight some of the features of BibSonomy that go beyond the basic collaborative tagging functionality and make it a valuable tool in particular for scientists. We focus on import and export options for bookmarks and publication references, ways to better organize one's tags with the help of tag relations and tag editing, the different aspects in which groups can support researchers, and briefly introduce the API, bibliographic hash keys, the basket page and the FolkRank.

3.5.1 Importing Resources

The most simplistic but also most laborious way to add posts to BibSonomy is by entering their metadata manually into form fields. To lower the efforts to get data into BibSonomy, it supports various ways to import resources from files and web pages which we describe in this section. Nevertheless, forms are still used to edit posts.

Importing Bookmarks

Users can import posts from Delicious very easily by supplying their credentials. The system then copies all posts from Delicious to BibSonomy. This functionality also takes into account the Delicious *bundles*. They are mapped to BibSonomy's \prec relation in the following way: for every bundle B (which is a set of tags) with name b we add $\{b\} \cup B$ to T and set

$$\prec := \prec \cup (\{u\} \times B \times \{b\})$$

where $u \in U$ is the user these bundles belong to.

It is also possible to import bookmark files of the Firefox¹⁹ web browser, where the typical folder hierarchy of the bookmarks can be added to the user's \prec relation. That means that, for every folder a and every subfolder b of a in Firefox, we add (u, b, a) to the user u 's \prec relation, if the user chooses to do so. Permanent synchronization of Firefox bookmarks with BibSonomy is possible using a Firefox add-on (see Section 3.5.4).

Importing Publication References from Files

Import of existing BIB_TE_X or EndNote²⁰ files is simple: after uploading the file, the user can tag the references or automatically assign them the tag *imported*. If an entry contains a *keywords* or *tags* field, its contents is attached as tags to the resource and added to the system. BIB_TE_X fields unknown to BibSonomy are saved in an extra *misc* field and will not get lost.

¹⁹<http://www.mozilla.com/firefox/>

²⁰<http://www.endnote.com/>

Importing Publication References from Digital Libraries

BibSonomy contains more than 60 so called ‘scrapers’ which allow to automatically extract publication metadata from digital libraries like SpringerLink²¹ or IEEE Xplore.²² 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.²³ The scrapers extract all available information and transform it into BIB_TE_X format (if necessary). The complete list of supported digital libraries can be found at <http://www.bibsonomy.org/scrapperinfo>; we also provide a web service²⁴ which only extracts the metadata from a given URL and returns it in BIB_TE_X or RDF format. Finally, the source code of all scrapers is available online in a Maven repository.²⁵

Information Extraction for Publication References Import

Typically, literature references can be imported only from services known by BibSonomy (i. e., supported by a scraper) or in well-defined formats like BIB_TE_X. This is a strong restriction, since many literature references in the web are neither available in BIB_TE_X format nor does BibSonomy offer an appropriate scraper. Those references rather appear in the form of human readable publication lists as shown in Figure 3.8. Hence, our efforts to enhance import focused on techniques to allow for the import of such resources. We integrated the MALLET (McCallum, 2002) system in BibSonomy which uses information extraction techniques (Peng and McCallum, 2004) to fill BIB_TE_X 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.

3.5.2 Exporting Bookmarks and Publication References

Exporting publication references in BIB_TE_X 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 BIB_TE_X format. For example, the page <http://www.bibsonomy.org/bib/search/text+clustering> returns a BIB_TE_X file containing all literature references which contain the words ‘text’ and ‘clustering’ in their fulltext.

²¹<http://www.springerlink.de/>

²²<http://ieeexplore.ieee.org/>

²³The corresponding buttons are available at <http://www.bibsonomy.org/buttons> and can easily be added to the browser’s toolbar.

²⁴<http://scraper.bibsonomy.org/>

²⁵<http://dev.bibsonomy.org/>

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 3.8: 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.

More general, every page which shows posts (see Section 3.4.3) 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 format

`/xml` bookmarks in an XML format compatible to the Delicious bookmark export

`/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 ontology²⁶

`/burst` publications as RSS feed containing the complete metadata according to the SWRC ontology, embedded according to the BuRST specification²⁷

²⁶<http://ontoware.org/projects/swrc/>

²⁷<http://www.cs.vu.nl/~pmika/research/burst/BuRST.html>

`/json` bookmarks and publications in JSON format²⁸ (for an integration example using Simile Exhibit²⁹ see http://www.kde.cs.uni-kassel.de/hotho/publication_json.html)

`/layout/l` publications in one of the available JabRef³⁰ layouts. Valid values for *l* include `html`, `simplehtml`, `tablerefs`, `tablerefsabsbib`, `tablerefsabsbibsort` (different HTML layouts), `harvard` (RTF; can be edited by Microsoft Word and OpenOffice), or `docbook` (DocBook³¹ XML).

`/layout/custom` publications in the custom layout of the user. Users can upload custom JabRef layouts³² 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.

3.5.3 Supporting Research Groups

In many situations it is desirable to share resources only among certain people. If the resources can be public, then one could agree to tag them with a special tag and use that tag to find the shared resources. The disadvantage is that this could be undermined by other users (or spammers) by using the same tag. To solve this problem and also to allow resources to be visible only for certain users, we introduced *groups* in BibSonomy which give users more options to decide with whom they share their resources.

It is thus possible to have private posts, which only the owner of the post can see, as well as posts which can be seen only by group members. Overall, there are several aspects of groups in BibSonomy:

²⁸<http://www.json.org/>

²⁹<http://code.google.com/p/simile-widgets/>

³⁰<http://jabref.sourceforge.net/>

³¹<http://www.docbook.org/>

³²<http://jabref.sourceforge.net/help/CustomExports.php>

1. One can get an aggregated view of all posts of the group members. For example, the URL <http://www.bibsonomy.org/group/kde/seminar2006> represents all posts the members of the *kde* group have tagged with *seminar2006*.
2. It is possible to use groups for privacy so that certain posts can only be seen by group members.
3. Users can explicitly mark posts to be *relevant for* a group. Those posts of the group *kde*, for example, can be shown by the URL <http://www.bibsonomy.org/relevantfor/group/kde>.
4. Resources can be copied directly to the group so that they're persistent, even if a user leaves the group. This is possible, since groups are implemented as a special user who has the name of the group and is also the group admin. He owns the copied references. This feature is in particular useful where the donator has to commit to the resource, e. g., for project deliverables or student projects.
5. While documents attached to publication posts are only visible to the post owner, for groups this restriction can be loosened to share documents among group members.

3.5.4 A 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 (Bork, 2006) based on the idea of REST (Fielding, 2000) 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 four 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 (cf. Section 3.4.1). For a detailed description of the available methods we refer to the online documentation available at <http://www.bibsonomy.org/help/doc/api.html>.

There is an abundant amount of applications using the API, most of them are probably private developments of individuals and institutions using BibSonomy and thus not known to the public. Prominent exceptions are the spam framework of BibSon-

omy (Krause et al., 2008b), a plugin³³ for the Java based bibliographic reference manager JabRef,³⁴ and an add-on for the Firefox web browser.³⁵ Another important example is the Library of the University of Cologne, which has made various efforts for close interaction of its catalogue³⁶ with BibSonomy. Search results from the catalogue can easily be added to a user’s personomy in BibSonomy by just clicking on an icon. The computation of the bibliographic hash keys (cf. Section 3.5.6) allows to match publications in the library with posts in BibSonomy and thus provides related tags which are shown besides the publication metadata. The integration goes further, since one can use the tags for browsing BibSonomy’s posts directly in the system. If a post’s publication is available in the library, the user can look at its metadata (e. g., at which particular library it is available, how many copies there are, etc.) and see where he can get it.

3.5.5 Tag Relations

Tagging gained so much popularity in the past years because it is simple and no specific skills are needed for it. Nevertheless, the longer people use systems like BibSonomy, the more often they ask for options to structure their tags. A user specific binary relation \prec between tags as described in our model of a Folksonomy (see Section 2.3) is an easy way to arrange tags. Therefore we included this possibility in BibSonomy.

To enable the addition of elements to the relation already during tagging, we decided to reserve the character sequences $\prec-$ and $->$. That means, if the user u enters $t_1->t_2$ on annotating a resource, we attach the tags t_1 and t_2 to the respective resource and add the triple (u, t_1, t_2) to the relation \prec . The tag $t_2\prec-t_1$ is interpreted as $t_1->t_2$. Consequently, it is not possible to have tags which contain the strings ‘ $\prec-$ ’ or ‘ $->$ ’. The semantics of the \prec relation is as described in Section 2.3 and can be read as ‘ t_1 is a t_2 ’ or ‘ t_1 is a *subtag* of the *supertag* t_2 ’. There are also other ways to add elements to \prec , in particular a relation editor.

Usage of this relation is made in several situations. First, the user can structure his tag cloud by showing all subtags of a certain supertag and therefore can see the tags in a hierarchy. Second, BibSonomy offers the option to show on a user’s tag page not only posts which contain a certain tag, but also posts which contain one of the subtags of the specific tag. This works also for tag intersections: given t_1, \dots, t_n and a user $u \in U$, then this page contains the set C of posts which is described in Equation 3.5 in Section 3.4.3. In general, all pages whose URL path starts with `/concept` make use of the \prec relation. An overview of user u ’s part of \prec (i. e., $\prec \cap (\{u\} \times T \times T)$) gives the page `/relations/u`.

³³<http://www.bibsonomy.org/help/doc/jabref-plugin/index.html>

³⁴<http://jabref.sourceforge.net/>

³⁵<https://addons.mozilla.org/de/firefox/addon/8855>

³⁶“Kölner Universitäts-Gesamtkatalog”, <http://kug.ub.uni-koeln.de/>

3.5.6 Bibliographic Hash Keys for Duplicate Detection

In particular for literature references there is the problem of detecting duplicate entries, because there are big variations in how users enter fields such as journal name or authors. On the one hand it is desirable to allow a user to have several entries which differ only slightly (e. g., a workshop publication and a journal article with the same title). On the other hand one might want to find other users' entries which refer to the same paper or book even if they are not completely identical.

To fulfill both goals we implemented two hashes to compare publication entries (Voss et al., 2009). One is for comparing the posts of a single user (*intra* user hash) and one for comparing the posts of different users (*inter* user hash). Comparison is accomplished by normalizing and concatenating $\text{BIB}_{\text{T}}\text{E}_{\text{X}}$ fields, hashing the result with the MD5 message digest algorithm (Rivest, 1992) and comparing the resulting hashes. MD5 hashing is done for efficiency reasons only, since this allows for a fixed length storage in the database. Storing the hashes along with the resources in the posts table enables fast comparison and search of posts.

The intra user hash is relatively strict and takes into account the fields *title*, *author*, *editor*, *year*, *entrytype*, *journal*, *booktitle*, *volume*, and *number*. This allows one to have articles with the same title from the same authors in the same year but in different volumes (e. g., a technical report and the corresponding journal article). In contrast, the inter user hash is less specific and includes only the *title*, *year* and *author* or *editor* (depending on what the user has entered).

In both hashes all fields which are taken into account are normalized, i. e., certain special characters are removed, whitespace and author/editor names normalized. The latter is done by concatenating the first letter of the first name by a dot with the last name, both in lower case. Persons are then sorted alphabetically by this string and concatenated by a colon. More details can be found on the web page <http://www.bibsonomy.org/help/doc/inside.html> together with an online computation service and source code examples.³⁷

The current duplicate detection is very simple and fails to detect spelling errors, differences in how special characters (like German umlauts) are entered or additional $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ commands. This is ongoing work; our implementation allows for simple addition of new hashes.

Currently, duplicate detection is used on the one hand to warn the user when she wants to add an already existing resource to her personomy and on the other hand to show how many users tagged a certain resource. A step beyond detecting duplicates could be providing the user with additional fields found in other entries referring to the same publication so that she can complete her own entries with additional information. Furthermore, the inter hash is used to identify resources in the chapters of Part II. There,

³⁷There is also an ongoing discussion on how to improve the hashing as well as implementations and variants listed on http://www.gbv.de/wikis/cls/Bibliographic_Hash_Key.

two resources are regarded as equal, if they have the same inter hash. This outweighs variations between resources and ensures sufficient overlap.

For bookmark entries in BibSonomy, their URLs are hashed with MD5 and this hash is used as both intra and inter hash. As can be seen from discussion with users, opinions on if and how to normalize URLs in such systems differ. On the one hand, URLs like `http://www.w3c.org/`, `http://w3c.org/` and `http://www.w3c.org/index.html` might denote the same resource, on the other hand they are different URLs and it is not obvious whether they really mean the same resource.

3.5.7 Collecting Publication Posts with the ‘Basket’

Every publication post can be ‘picked’ by the user and is then available in a ‘shopping basket’-like download area on the `/basket` page. This is useful for collecting references one needs for a publication. Since all publication related export options mentioned in Section 3.5.2 apply to the basket page, it is straightforward to get the collected posts in BIBTEX, EndNote or another supported format.

3.5.8 Tag Editing

Besides changing the tags of a post by editing it, BibSonomy offers at the moment two other ways of changing the tags of several posts at once. Both methods support the user in creating and maintaining a consistent tag vocabulary.

First, by preceding the path part of a URL with `/bediturl` (or `/beditbib`) one can edit the tags of all own bookmarks (or publications) on the corresponding page at once. This function is also available through links on the respective pages.

Second, BibSonomy has an $m:n$ tag editor which allows a user to exchange m tags by n other tags. More precisely: given two sets A and B of tags³⁸ and a user $u \in U$, then the $m:n$ tag editor sets iteratively for every $r \in R_u$ with $A \subseteq T(u, r)$:

$$Y := (Y \setminus (\{u\} \times A \times \{r\})) \cup (\{u\} \times B \times \{r\}).$$

3.5.9 FolkRank

In (Hotho et al., 2006b) we introduced the FolkRank algorithm for search and ranking in Folksonomies (see Section 5.3.2). An offline computed version is integrated into BibSonomy and provides posts for the `/tag/t1...tn` page in ranked order – as an alternative to the usual ordering by posting date. 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’. An example of

³⁸If B contains tags not already included in T , then T is adjusted in the obvious way.

the top five bookmarks and publication references for the tag *folksonomy* can be seen in Figure 3.9, the corresponding related users and tags in Figure 3.10.

To allow for efficient retrieval of the FolkRank ordered posts, we regularly pre-compute for each tag $t \in T$ the ranking $\vec{w}(t)$ of all resources, users, and tags – as defined by Equation 5.3 in Section 5.3.2 – and store the top 100 elements of each dimension in the database. Retrieval for queries with only one tag (i. e., $/\mathbf{tag}/t$) then simply returns those elements for the tag t . However, due to the combinatorial explosion, the offline computation for queries with more than one tag, i. e., $/\mathbf{tag}/t_1 \dots t_n$ is not possible. Thus, we create a merged ranking for the tags t_1, \dots, t_n using the function

$$\vec{w}[x](t_1, \dots, t_n) := \sum_{i=1}^n \vec{w}[x](t_i)$$

which aggregates the FolkRank weights $\vec{w}[x](t_i)$ for each x of the top 100 elements of each dimension that are stored in the database.

3.6 Outlook

From an administrative point of view, the future operation of BibSonomy is assured by funding from at least two DFG (Deutsche Forschungsgemeinschaft) projects which use BibSonomy as research platform (“Info 2.0 – Informationelle Selbstbestimmung im Web 2.0”) or aim to extend its features for academic publication management (“PUMA – Akademisches Publikationsmanagement”). In particular the PUMA project will pave the way for better interaction with classical library systems.

Ongoing work is the implementation and integration of new features based on our research results. One such example is shown in Chapter 6, where we describe BibSonomy’s tag recommendation framework which is a result of our work on analysing tag recommendations for folksonomies (cf. Chapter 5). More recently, tag similarity measures (Cattuto et al., 2008) have been integrated to improve navigation by presenting similar tags to the users. In the future we plan to further support the social aspects of BibSonomy with the help of community detection methods and by extending features exclusively available for groups (e. g., pre-defined tag spaces, or collective editing of posts).

BibSonomy::home ☆
to bibtex bibliography by mobileink and [126 other people](#) on Jan 12, 2006, 6:59 PM
[copy](#) | [spam](#)

Folksonomies - Cooperative Classification and Communication Through Shared Metadata ☆
to article folksonomy discussion definition by hotho and [40 other people](#) on Nov 18, 2005, 9:50 AM
[copy](#) | [spam](#)

Shirky: Ontology is Overrated -- Categories, Links, and Tags ☆
Ways of organization: Why categorization is (often) not so good and tagging is (usually) much better - long article but very worth reading!
to article folksonomy ontology tagging by dolefulrabbit and [31 other people](#) on Nov 30, 2005, 5:22 PM
[copy](#) | [spam](#)

Social Bookmarking Tools (I): A General Review ☆
to social folksonomy tagging bookmarking by dolefulrabbit and [38 other people](#) on Nov 22, 2005, 5:06 PM
[copy](#) | [spam](#)

Folksonomies: Tidying up Tags? ☆
to folksonomy ontology tag tagging tags delicious Reading flickr by jintan and [19 other people](#) on Feb 6, 2006, 9:14 PM
[copy](#) | [spam](#)

Information Retrieval in Folksonomies: Search and Ranking ☆
Andreas Hotho and Robert Jäschke and Christoph Schmitz and Gerd Stumme *Proceedings of the 3rd European Semantic Web Conference \emph(accepted for publication), Budva, Montenegro, June2006.*
to informationretrieval tagging by nichtich and [66 other people](#) on Apr 29, 2006, 1:56 AM
[pick](#) | [copy](#) | [BibTeX](#) | [openURL](#) | [spam](#)

BibSonomy: A Social Bookmark and Publication Sharing System ☆
Andreas Hotho and Robert Jäschke and Christoph Schmitz and Gerd Stumme *Proceedings of the First Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures, page87-102. Aalborg, Aalborg Universitetsforlag, (2006)*
to 2006 FCA OntologyHandbook bibsonomy bookmarking folksonomy iccs I3s myown nepomuk social tagorapub by stumme and [47 other people](#) on Sep 20, 2006, 6:19 PM
[pick](#) | [copy](#) | [URL](#) | [BibTeX](#) | [openURL](#) | [spam](#)

The Structure of Collaborative Tagging Systems ☆
Scott Golder and Bernardo A. Huberman *Aug2005.*
to clustering folksonomy folksonomies delicious self-organization social-networks information_organization socialtagging collaborative_tagging emergence tagging tag tags no-tag by lkl_kss and [53 other people](#) on Mar 25, 2006, 1:26 AM
[pick](#) | [copy](#) | [URL](#) | [BibTeX](#) | [openURL](#) | [spam](#)

Ontologies Are Us: A Unified Model of Social Networks and Semantics. ☆
Peter Mika *International Semantic Web Conference, page522-536. Springer, (2005)*
to web2.0 folksonomy ontology by motte and [51 other people](#) on Jan 19, 2006, 1:30 PM
[pick](#) | [copy](#) | [URL](#) | [BibTeX](#) | [openURL](#) | [spam](#)

Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems ☆
Paul Heymann and Hector Garcia-Molina *Computer Science Department, April2006.*
to tagging taxonomy algorithm collaboration by itst and [37 other people](#) on Jul 8, 2006, 12:56 AM
[pick](#) | [copy](#) | [URL](#) | [BibTeX](#) | [openURL](#) | [spam](#)

Figure 3.9: 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.



Figure 3.10: 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=folkcrank> on June 8th, 2009.

Part II

Knowledge Discovery

According to Fayyad et al. (1996), "Knowledge Discovery in Databases is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data."

In this part we introduce the Trias algorithm for discovering frequent closed itemsets in triadic data (like folksonomies) and evaluate several methods for tag recommendations in collaborative tagging systems.

Chapter 4

Formal Concept Analysis

The folksonomies that the users of collaborative tagging systems are setting up represent lightweight conceptual structures. Unlike ontologies, these shared conceptualizations are not formalized, but rather implicit. Discovering such hidden conceptualizations can be regarded as a first step for learning ontologies from folksonomies and thereby make the knowledge represented by folksonomies more explicit. In this chapter we present a new data mining task, the *mining of all frequent tri-concepts*, together with an efficient algorithm, for discovering these implicit shared conceptualizations. Our approach extends the data mining task of discovering all closed itemsets to three-dimensional data structures to allow for mining folksonomies. We provide a formal definition of the problem, and present an efficient algorithm for its solution. Finally, we show the applicability of our approach on three large real-world examples. The work in this chapter has been published in (Jäschke et al., 2006, 2007a, 2008a).

4.1 Introduction

First, we briefly introduce the notion of a *tri-concept* and then discuss the relation of our work to the fields of Closed Itemset Mining and Formal Concept Analysis.

4.1.1 Discovering Shared Conceptualizations

Unlike ontologies, folksonomies do not suffer from the knowledge acquisition bottleneck (Hayes-Roth et al., 1983), as the significant provision of content by many people shows. On the other hand, folksonomies – unlike ontologies (Gruber, 1993) – do not explicitly state shared conceptualizations, nor do they force users to use the same tags. However, the usage of tags of users with similar interests tends to converge to a shared vocabulary. Our intention is to discover these shared conceptualizations that are hidden in a folksonomy. To this end, we present the TRIAS algorithm for discovering subsets of folksonomy users who implicitly agree (on subsets of resources) on a common conceptualization.

Our algorithm will return a tri-ordered¹ set of triples, where each triple (A, B, C) consists of a set A of users, a set B of tags, and a set C of resources. These triples

¹See Section 4.2.3 for details.

– called *tri-concepts* in the sequel – have the property that each user in A has tagged each resource in C with all tags from B , and that none of these sets can be extended without shrinking one of the other two dimensions. Each retrieved triple indicates thus a set A of users who (implicitly) share a conceptualization, where the set B of tags is the intension of the concept, and the set C of resources is its extension. We can additionally impose minimum support constraints on each of the three dimensions ‘users’, ‘tags’, and ‘resources’, to retrieve the most significant shared concepts only.

4.1.2 The Problem of Closed Itemset Mining in Triadic Data

From a data mining perspective, the discovery of shared conceptualizations opens a new research field which may prove interesting also outside the folksonomy domain: ‘Closed itemset mining in triadic data’, which is located on the confluence of the research areas of Association Rule Mining (Agrawal et al., 1993) and Formal Concept Analysis.

Formal Concept Analysis (FCA) (Wille, 1982; Ganter and Wille, 1999) is a mathematical theory that formalizes the concept of ‘concept’, and allows for computing concept hierarchies out of data tables. At the end of last century, one discovered that it also provides an elegant framework for significantly reducing the effort of mining association rules (Pasquier et al., 1999a; Zaki and Hsiao, 1999; Stumme, 1999). A new research area emerged which became known as *closed itemset mining* in the data mining community and as *iceberg concept lattices* (Stumme et al., 2002) in FCA.

Independent of this development, Formal Concept Analysis has been extended more than ten years ago to deal with three-dimensional data (Lehmann and Wille, 1995). This line of *Triadic Concept Analysis* did not receive a broad attention up to now. With the rise of *folksonomies* as core data structure of collaborative tagging systems, however, the interest in Triadic Concept Analysis increased again.

Here, we initiate the confluence of both lines of research, Triadic Concept Analysis and Closed Itemset Mining (see Figure 4.1). In particular, we give a formal definition of the *problem of mining all frequent tri-concepts* (in other terms: the three-dimensional version of mining all frequent closed itemsets), and present our algorithm TRIAS for mining all frequent tri-concepts of a given dataset.

With its sets of users, tags, and resources, folksonomies have one additional dimension compared to typical basket analysis datasets (which consist of the two dimensions ‘items’ and ‘transactions’). Informally spoken, the task of mining all frequent *tri-sets* is to discover all triples of sets of users, tags, and resources, resp., such that, for each triple of sets, all users in the first set have assigned all tags in the second set to all resources in the third set, and that the cardinalities of the three sets are above predefined minimum support thresholds.²

²In classical association rule mining, the thresholds equal the minimum support and minimal length thresholds.

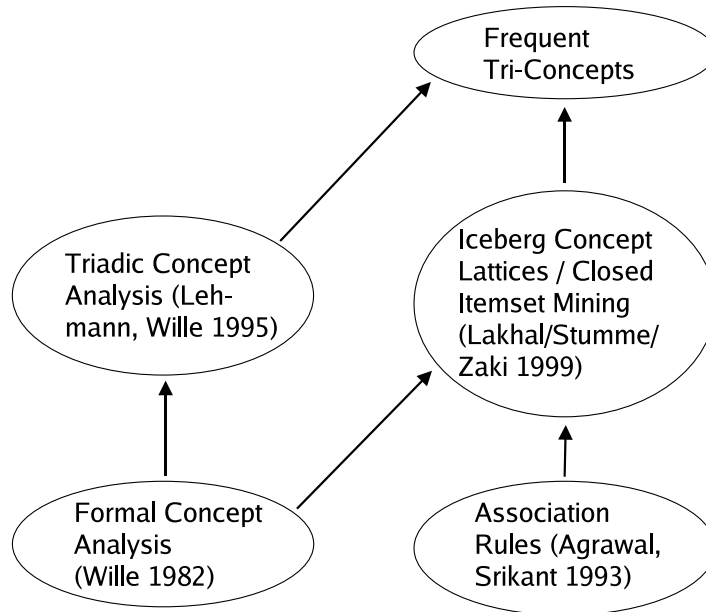


Figure 4.1: The history of iceberg tri-lattices.

As in the classical case, the resulting set of all frequent tri-sets is usually too large, and can be condensed without any loss of information. To this end, we adapt the notion of iceberg concept lattices (aka closed itemsets) to the three-dimensional nature of folksonomies. With our TRIAS algorithm, we provide an efficient method for computing all frequent tri-concepts.

4.1.3 Contribution and Organization of this Chapter

In this chapter, we present the following contributions:

- a formal definition of the problem of mining frequent tri-concepts,
- TRIAS, an efficient algorithm for solving the problem,
- and a conceptual analysis of two social bookmarking systems and an IT security manual by means of this algorithm.

The chapter is organized as follows. In the next section, we discuss the state of the art and related work in the research areas of Ontology Learning, Formal Concept Analysis, and Closed Itemset Mining. In Section 4.3.1, we provide the formal definition of the problem of mining all frequent tri-concepts; in Section 4.3.2, we introduce our TRIAS algorithm; and in Section 4.3.3, we evaluate its performance. In Section 4.4, we apply our approach

on three large-scale real-world applications: the folksonomy of the popular bookmark sharing system Delicious, a manual for protecting IT infrastructure, and the collection of publications in our social reference management system BibSonomy (introduced in Chapter 3). Section 4.6 concludes with an outlook on future work.

4.2 Basic Notions and State of the Art

In this section, we recall the basic notions and discuss the state of the art of the research areas relevant to this chapter: Ontology Learning, Formal Concept Analysis and its triadic version, and the mining of closed itemsets.

4.2.1 Ontology Learning

The term *ontology learning* was first introduced by Maedche and Staab (2001). It stands for the task of (semi-)automatically constructing an ontology or a domain model. Usually machine learning or data mining algorithms are applied mostly on textual data to extract the hidden conceptualization from the data and to make it explicit. Revealing the hidden conceptualization of an author partially written in a text document can be seen as a kind of reverse engineering task (cf. (Cimiano, 2006)). All ontology learning approaches try to support the knowledge engineer in setting up the ontology. Recent advances in ontology learning are described in (Buitelaar et al., 2005).

In this chapter, we describe one step for learning ontologies from folksonomies. Other approaches are discussed in the next paragraph.

Related Work. Approaches trying to analyze the weakly structured information of folksonomies and use this to learn conceptualization or ontologies are still rare. Among them is the work of Mika (2005), who defines a model of semantic-social networks for extracting lightweight ontologies from Delicious. Besides calculating measures like the clustering coefficient, (local) betweenness centrality or the network constraint on the extracted one-mode network, Mika uses co-occurrence techniques for clustering the folksonomy. Schmitz (2006) proposes the construction of a subsumption tree consisting of Flickr tags based on the tag co-occurrence network of tags. Both approaches are showing ways to construct an ontology, but both are using only parts of the information of a folksonomy as they are based on an aggregated graph rather than the full folksonomy.

Heymann and Garcia-Molina (2006) propose an algorithm to construct a tag hierarchy from tagging data. First, they represent each tag as a vector which for each resource counts how often the tag has been used to annotate that resource. Then, they build a graph with tags as nodes and an edge between two tags, if the (cosine) similarity between the vectors of those tags is above a certain threshold. Finally, using betweenness centrality to measure the generality of tags they are able to construct a hierarchical

taxonomy of tags. Benz and Hotho (2007) extend this work by using different underlying tag graphs based on user and resource co-occurrence and by employing degree instead of betweenness centrality which allows for a more efficient computation. As a further extension, they consider tag compounds and synonyms.

4.2.2 Formal Concept Analysis

Formal Concept Analysis (FCA) is a conceptual clustering technique that formalizes the concept of ‘concept’ as established in the international standard ISO 704: a concept is considered as a unit of thought constituted of two parts: its extension and its intension (Wille, 1982; Ganter and Wille, 1999). This understanding of ‘concept’ is first mentioned explicitly in the Logic of Port Royal (Arnauld and Nicole, 1668). To allow a formal description of extensions and intensions, FCA starts with a (*formal*) *context*:

Definition 4.1 (Wille, 1982) *A formal context is a triple $\mathbb{K} := (G, M, I)$ which consists of a set G of objects [German: *Gegenstände*], a set M of attributes [Merkmale], and a binary relation $I \subseteq G \times M$. $(g, m) \in I$ is read as “object g has attribute m ”.*

This data structure equals the set of transactions used for association rule mining, if we consider M as the set of items and G as the set of transactions.

Definition 4.2 (Wille, 1982) *For $A \subseteq G$, let*

$$A^I := \{m \in M \mid \forall g \in A: (g, m) \in I\} ;$$

and dually, for $B \subseteq M$, let

$$B^I := \{g \in G \mid \forall m \in B: (g, m) \in I\} .$$

Now, a formal concept is a pair (A, B) with $A \subseteq G$, $B \subseteq M$, $A^I = B$ and $B^I = A$. Then A is called extent and B is called intent of the concept.

This is equivalent to saying that $A \times B \subseteq I$ such that neither A nor B be can be enlarged without violating this condition.

Definition 4.3 (Wille, 1982) *The set $\mathfrak{B}(\mathbb{K})$ of all concepts of a formal context \mathbb{K} together with the partial order $(A_1, B_1) \leq (A_2, B_2) :\Leftrightarrow A_1 \subseteq A_2$ (which is equivalent to $B_1 \supseteq B_2$) is a complete lattice, called the concept lattice of \mathbb{K} .*

The concept lattice is a hierarchical conceptual clustering of the data which can be visualized by a Hasse diagram. This visualization technique has been used in many applications for qualitative data analysis (Ganter et al., 2005). An example of a Hasse diagram is given in Figure 4.6 and described in more detail in Section 4.4.1.

Related Work. FCA has grown over the years to a powerful theory for data analysis, information retrieval, and knowledge discovery (Stumme and Wille, 2000). In Artificial Intelligence (AI), FCA is used as a knowledge representation mechanism (Stumme, 2003) and as conceptual clustering method (Strahringer and Wille, 1993; Carpineto and Romano, 1993; Mineau et al., 1985). In database theory, FCA has been extensively used for class hierarchy design and management (Missikoff and Scholl, 1989; Yahia et al., 1996; Dicky et al., 1996; Waiyama et al., 1997; Schmitt and Saake, 1998; Godin et al., 1998).

The amount of publications on Formal Concept Analysis is abundant. A good starting point for the lecture are the textbooks (Ganter and Wille, 1999; Carpineto and Romano, 2004; Ganter et al., 2005), the collection of FCA publications in BibSonomy,³ and the proceedings of the International Conference on Formal Concept Analysis⁴ and the International Conference on Conceptual Structures⁵ series.

4.2.3 Triadic Concept Analysis

Inspired by the pragmatic philosophy of Charles S. Peirce with its three universal categories (Peirce, 1931–1935), Rudolf Wille and Fritz Lehmann extended Formal Concept Analysis in 1995 with a third category:

Definition 4.4 (Lehmann and Wille, 1995) *A triadic formal context is a quadruple $\mathbb{F} := (G, M, B, Y)$ where G , M , and B are sets, and Y is a ternary relation between G , M , and B , i. e., $Y \subseteq G \times M \times B$. The elements of G , M , and B are called (formal) objects, attributes, and conditions, resp, and $(g, m, b) \in Y$ is read “object g has attribute m under condition b ”.*

A triadic formal context models exactly the structure of a folksonomy $\mathbb{F} := (U, T, R, Y)$ without the sub-/supertag relation \prec , as introduced in Section 2.3.

Definition 4.5 (Lehmann and Wille, 1995) *A triadic concept of the context \mathbb{F} is a triple (A_1, A_2, A_3) with $A_1 \subseteq G$, $A_2 \subseteq M$, and $A_3 \subseteq B$ with $A_1 \times A_2 \times A_3 \subseteq Y$ such that none of its three components can be enlarged without violating this condition.*

From each of the three dimensions one obtains a quasi-order \lesssim_1 , \lesssim_2 , and \lesssim_3 , resp., on the set of all tri-concepts: For $i = 1, 2, 3$, let $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3)$ iff $A_i \subseteq B_i$.

The definition of a triadic concept is the natural extension of the definition of a formal concept to the triadic case. Alternatively the definition can be described with \cdot^I operators similar to the dyadic case, but as there are now three dimensions involved, the notation (which we omit here, cf. (Lehmann and Wille, 1995)) becomes more complex.

³<http://www.bibsonomy.org/tag/fca>

⁴<http://www.informatik.uni-trier.de/~ley/db/conf/icfca/>

⁵<http://www.informatik.uni-trier.de/~ley/db/conf/iccs/>

Lemma 4.1 (Lehmann and Wille, 1995) For two tri-concepts \mathbf{a} and \mathbf{b} , and for $i \neq j \neq k \neq i$, $\mathbf{a} \lesssim_i \mathbf{b}$ and $\mathbf{a} \lesssim_j \mathbf{b}$ implies $\mathbf{b} \lesssim_k \mathbf{a}$.

This implication is the triadic version of the dyadic proposition that for two dyadic concepts (A_1, A_2) and (B_1, B_2) holds $A_1 \subseteq B_1$ iff $B_2 \subseteq A_2$. In the dyadic case, the two orders induced by the concept extents and the concept intents, resp. are thus dually isomorphic. This allows for visualizing the concept lattice in just one diagram and is at the same time the justification for the famous support pruning strategy in the Apriori algorithm. In the triadic case, the relationship between the three quasi-orders is unfortunately weaker (as seen above), which makes both the mining (see Section 4.3.2) and the visualization (see Section 4.4) more complex. Figures 4.7–4.9 show examples of diagrams of triadic concept lattices; they are discussed in detail in Section 4.4.

Lehmann and Wille (1995) present an extension of the theory of ordered sets and (concept) lattices to the triadic case, and discuss structural properties. This approach initiated research on the theory of *concept trilattices*.

Whereas there have been some significant publications on the mathematical properties of trilattices (see below), this approach had no large impact on real-world applications up to now. This is mainly due to its above-mentioned resistance to scalable visualizations. With the rise of collaborative tagging systems on the web, triadic data move again in the focus of many researchers. In this setting, one needs – beside a more scalable visualization paradigm – knowledge discovery and information retrieval methods and algorithms that are able to handle very large datasets.

Related Work. Following the initial paper by Lehmann and Wille (1995), several researchers started to analyze the mathematical properties of trilattices, e.g., (Biedermann, 1997b,a, 1998b; Dau and Wille, 2001; Ganter and Obiedkov, 2004; Wille, 1995; Wille and Zickwolff, 2000). Lehmann and Wille (1995) and Dau and Wille (2001) present several ways to project a triadic context to a dyadic one. Stumme (2005) presents a model for navigating a triadic context by visualizing concept lattices of such projections. In (Schmitz et al., 2006) is discussed, how to compute association rules from a triadic context, based on these (and other) projections. A first step towards truly ‘triadic association rules’ has been done in (Ganter and Obiedkov, 2004).

4.2.4 Closed Itemset Mining

In terms of Formal Concept Analysis, the task of mining frequent itemsets (Agrawal et al., 1993) can be described as follows: Given a formal context $\mathbb{K} = (G, M, I)$ and a threshold $\text{minsupp} \in [0, 1]$, determine all subsets B of M where the *support* $\text{supp}(B) := \frac{|B^I|}{|G|}$ (with B^I as defined above) is larger than the threshold minsupp . In warehouse basket analysis, M is the set of items and G is the set of transactions.

The set of these so-called *frequent itemsets* itself is usually not considered as a final result of the mining process, but rather an intermediate step. Its most prominent use are association rules (Agrawal et al., 1993). Association rules are for instance used in warehouse basket analysis, where the warehouse management is interested in learning about products that are frequently bought together.

Since determining the frequent itemsets is the computationally most expensive part, most research has focused on this aspect. Most algorithms follow the way of the well-known Apriori algorithm (Agrawal and Srikant, 1994), which is traversing iteratively the set of all itemsets in a levelwise manner. Algorithms based on this approach have to extract the supports of *all* frequent itemsets from the database. However, this is by no means necessary.

It turned out that FCA can significantly improve both the efficiency and the effectiveness of frequent itemset mining. (Pasquier et al., 1999a; Zaki and Hsiao, 1999; Stumme, 1999) discovered independently that it is sufficient to consider the intents of those concepts where the cardinality of their extent is above the minimum support threshold. These frequent concept intents are called *closed itemsets* in association rule mining, because the set of all concept intents is a closure system (i. e., it is closed under set intersection). The corresponding closure operator is the consecutive application of the two \cdot^I operators defined in Definition 4.2. I. e., for an itemset B , the set B^{II} is the smallest concept intent containing B . This closure operator will be used in the TRIAS algorithm in Section 4.3.2.

In FCA, the equivalent notion is that of an *iceberg concept lattice* (Stumme et al., 2002), which is the \vee -semi-lattice $\{(A, B) \in \mathfrak{B}(\mathbb{K}) \mid \frac{|A|}{|G|} \geq \text{minsupp}\}$ with the order defined in Section 4.2.2. The iceberg concept lattice visualizes the most frequent concepts of a dataset (Stumme et al., 2002), and allows for an efficient visualization of a basis (condensed set) of association rules (Stumme et al., 2001; Pasquier et al., 2005). These bases allow to reduce the number of rules significantly without losing any information.

Related Work. The problem of mining frequent itemsets arose first as a sub-problem of mining association rules (Agrawal et al., 1993), but it then turned out to be present in a variety of problems: mining sequential patterns (Agrawal and Srikant, 1995), episodes (Mannila, 1997), association rules (Agrawal and Srikant, 1994), correlations (Silverstein et al., 1998), multi-dimensional patterns (Kamber et al., 1997; Lent et al., 1997), maximal itemsets (Bayardo, 1998; Zaki et al., 1997; Lin and Kedem, 1998), closed itemsets (Taouil, 2000; Pasquier et al., 1999a,b; Pei et al., 2000).

The first algorithm based on the combination of association rule mining with FCA was Close (Pasquier et al., 1999a), followed by A-Close (Pasquier et al., 1999b), ChARM (Zaki and Hsiao, 1999), Pascal (Bastide et al., 2000), Closet (Pei et al., 2000), and Titanic (Stumme et al., 2002), each having its own way to exploit the closure operator which is hidden in the data. Many algorithms can be found at the Frequent Itemset

Mining Implementations Repository.⁶

Beside closed itemsets, other condensed representations have been studied: key sets (Bastide et al., 2000)/free sets (Boulicaut et al., 2000), δ -free sets (Boulicaut et al., 2000), non-derivable itemsets (Calders and Goethals, 2002), disjunction free sets (Bykowski and Rigotti, 2001), and k -free sets (Riout, 2005). Closed itemsets and other condensed representations can be used for defining bases of association rules (Stumme et al., 2001; Pasquier et al., 2005).

4.3 Mining Frequent Tri-Concepts of a Folksonomy

In this section we formalize the problem of mining all frequent tri-concepts of a folksonomy, present the TRIAS algorithm for its efficient solution, and discuss its performance.

4.3.1 The Problem of Mining all Frequent Tri-Concepts

We will now formalize the problem of mining all frequent tri-concepts. We start with an adaptation of the notion of ‘frequent itemsets’ to the triadic case.

Definition 4.6 *Let $\mathbb{F} := (U, T, R, Y)$ be a folksonomy/triadic context. A tri-set of \mathbb{F} is a triple (A, B, C) with $A \subseteq U$, $B \subseteq T$, $C \subseteq R$ such that $A \times B \times C \subseteq Y$.*

As folksonomies have three dimensions which are completely symmetric, one can establish minimum support thresholds on all of them. The general problem of mining frequent tri-sets is then the following:

Problem 4.1 (Mining all frequent tri-sets) *Let $\mathbb{F} := (U, T, R, Y)$ be a folksonomy/triadic context, and let $u\text{-minsup}, t\text{-minsup}, r\text{-minsup} \in [0, 1]$. The task of mining all frequent tri-sets consists in determining all tri-sets (A, B, C) of \mathbb{F} with $\frac{|A|}{|U|} \geq u\text{-minsup}$, $\frac{|B|}{|T|} \geq t\text{-minsup}$, and $\frac{|C|}{|R|} \geq r\text{-minsup}$.*

This is actually a harder problem than the direct adaptation of frequency to one more dimension: In classical frequent itemset mining, one has a constraint – the frequency – only on one dimension (the number of transactions). Thus the equivalent triadic version of the problem would need two minimum support thresholds only (say $u\text{-minsup}$ and $t\text{-minsup}$). However, this seems not natural as it breaks the symmetry of the problem. Hence we decided to go for the harder problem directly (which equals in the dyadic case the addition of a minimal length constraint on the itemsets). The lighter version with only two constraints is then just a special case (e. g., by letting $r\text{-minsup} = 0$).

⁶<http://fimi.cs.helsinki.fi/>

As in the dyadic case, our thresholds are monotonic/antimonotonic constraints: If (A_1, B_1, C_1) with A_1 maximal for $A_1 \times B_1 \times C_1 \subseteq Y^7$ is not u -frequent, then all (A_2, B_2, C_2) with $B_1 \subseteq B_2$ and $C_1 \subseteq C_2$ are not u -frequent either. The same holds symmetrically for the other two dimensions.

With the step from two to three dimensions, however, the direct symmetry between monotonicity and antimonotonicity (which results in the dyadic case from the dual order isomorphism between the set of concept extents and the set of concept intents) breaks. All we have in the triadic case is the following lemma which results (via the three quasi-orders defined in Section 4.2.3) from the triadic Galois connection (Biedermann, 1997a) induced by a triadic context.

Lemma 4.2 (Lehmann and Wille, 1995) *Let both (A_1, B_1, C_1) and (A_2, B_2, C_2) be tri-sets with A_i being maximal for $A_i \times B_i \times C_i \subseteq Y$, for $i = 1, 2$.⁸ If $B_1 \subseteq B_2$ and $C_1 \subseteq C_2$ then $A_2 \subseteq A_1$. The same holds symmetrically for the other two directions.*

As the set of all frequent tri-sets is highly redundant, we will in particular consider a specific condensed representation, i. e., a subset which contains the same information, namely the set of all frequent tri-concepts.

Definition 4.7 *A tri-set is a frequent tri-concept if it is both a tri-concept and a frequent tri-set.*

Problem 4.2 (Mining all frequent tri-concepts) *Let $\mathbb{F} := (U, T, R, Y)$ be a folksonomy/triadic context, and let u -minsup, t -minsup, r -minsup $\in [0, 1]$. The task of mining all frequent tri-concepts consists in determining all tri-concepts (A, B, C) of \mathbb{F} with $\frac{|A|}{|U|} \geq u$ -minsup, $\frac{|B|}{|T|} \geq t$ -minsup, and $\frac{|C|}{|R|} \geq r$ -minsup.*

Sometimes it is more convenient to use absolute rather than relative thresholds. For this case we let $\tau_u := |U| \cdot u$ -minsup, $\tau_t := |T| \cdot t$ -minsup, and $\tau_r := |R| \cdot r$ -minsup.

Once Problem 4.2 is solved, we obtain the answer to Problem 4.1 in a straightforward enumeration as $\{(A, B, C) \mid \exists \text{ frequent tri-concept } (\hat{A}, \hat{B}, \hat{C}): A \subseteq \hat{A}, B \subseteq \hat{B}, C \subseteq \hat{C}, |A| \geq \tau_u, |B| \geq \tau_t, |C| \geq \tau_r\}$.

4.3.2 The Trias Algorithm for Mining all Frequent Tri-Concepts

Our algorithm for mining all frequent tri-concepts of a folksonomy $\mathbb{F} := (U, T, R, Y)$ is listed as Algorithm 4.1. A prior version was used for analysing psychological studies (Krolak-Schwerdt et al., 1994). That application varied from TRIAS as it aimed at

⁷In the dyadic case this condition is implicitly covered by the use of B^I in the definition of the support since, for any given $B \subseteq M$, the set B^I is always maximal with $B^I \times B \subseteq I$.

⁸This holds in particular if the tri-sets are tri-concepts, see Lemma 4.1.

an iterative pruning of the data set. Furthermore, it did not take into account any frequency constraints.

We let $\tilde{Y} := \{(u, (t, r)) \mid (u, t, r) \in Y\}$, and we identify the elements of U , T , and R with natural numbers, i. e., $U = \{1, \dots, |U|\}$ (and similarly for T and R). In both its outer and its inner loop, TRIAS calls the pairs of subroutines *FirstFrequentConcept* $((G, M, I), \tau)$ and *NextFrequentConcept* $((A, B), (G, M, I), \tau)$. These two routines provide an enumeration of all frequent dyadic concepts (A, B) of the formal (dyadic) context (G, M, I) . The context is passed over as input parameter. *FirstFrequentConcept* returns in (A, B) the first concept of the enumeration. *NextFrequentConcept* takes the current concept (A, B) and modifies it to the next concept of the enumeration. This way, we compute all frequent maximal cuboids in the relation Y by consecutively computing maximal rectangles in the binary relations \tilde{Y} and I , resp, where the condition in line 8 of Algorithm 4.1 checks if the rectangle layers form a maximal cuboid. Note that $A \subseteq (B \times C)^{\tilde{Y}}$ trivially holds, because of $A = I^{\tilde{Y}}$ and $(B \times C) \subseteq I$. Hence only ‘ \supseteq ’ has to be checked.

Algorithm 4.1 The TRIAS algorithm for mining all frequent tri-concepts.

Require: $U, T, R, Y, \tau_u, \tau_t, \tau_r$

- 1: $\tilde{Y} := \{(u, (t, r)) \mid (u, t, r) \in Y\}$
- 2: $(A, I) := \text{FirstFrequentConcept}((U, T \times R, \tilde{Y}), \tau_u)$
- 3: **repeat**
- 4: **if** $|I| \geq \tau_t \cdot \tau_r$ **then**
- 5: $(B, C) := \text{FirstFrequentConcept}((T, R, I), \tau_t)$
- 6: **repeat**
- 7: **if** $|C| \geq \tau_r$ **then**
- 8: **if** $A = (B \times C)^{\tilde{Y}}$ **then**
- 9: **print** A, B, C
- 10: **end if**
- 11: **end if**
- 12: **until not** $\text{NextFrequentConcept}((B, C), (T, R, I), \tau_t)$
- 13: **end if**
- 14: **until not** $\text{NextFrequentConcept}((A, I), (U, T \times R, \tilde{Y}), \tau_u)$

For computing all (frequent) maximal rectangles in a binary relation, one can resort to any algorithm for computing (iceberg) concept lattices. The enumeration can be done in any convenient way. For the inner and the outer loop, one could use different algorithms for that task.

In our implementation we equipped the NEXT CLOSURE algorithm (Ganter, 1987; Ganter and Wille, 1999) with frequency pruning for implementing the *FirstFrequentConcept* and *NextFrequentConcept* routines (see Algorithms 4.2 and 4.3, resp.) for both the outer and the inner loop. This algorithm has the advantage that it needs almost no

Algorithm 4.2 The *FirstFrequentConcept* method of the TRIAS algorithm.

Require: $(G, M, I), \tau$

- 1: $A := \emptyset^I$
 - 2: $B := A^I$
 - 3: **if** $|A| < \tau$ **then**
 - 4: $\text{NextFrequentConcept}((A, B), (G, M, I), \tau)$
 - 5: **end if**
 - 6: **return** (A, B)
-

Algorithm 4.3 The *NextFrequentConcept* method of the TRIAS algorithm.

Require: $(A, B), (G, M, I), \tau$

- 1: **while** $\text{defined}(i)$ **do**
 - 2: $A := (B \oplus i)^I$
 - 3: **if** $|A| \geq \tau$ **then**
 - 4: $D := A^I$
 - 5: **if** $B <_i D$ **then**
 - 6: $B := D$
 - 7: **return true**
 - 8: **end if**
 - 9: **end if**
 - 10: $i := \max(M \setminus B \cap \{1, \dots, i-1\})$
 - 11: **end while**
 - 12: **return false**
-

space in main memory.

NEXT CLOSURE computes the concepts of a dyadic formal context (G, M, I) in a particular order, starting with the concept $(\emptyset^I, \emptyset^{II})$. For a given concept (A, B) , NEXT CLOSURE computes the concept (C, D) whose intent D is the next set after B in the so-called *lectic* order. The lectic order on sets is a total order and is equivalent to the lexicographic order of bit vectors representing those sets.

To find the next concept we define, for $B \subseteq M$ and $i \in M$,

$$B \oplus i := (B \cap \{1, \dots, i-1\}) \cup \{i\}.$$

By applying the closure operator \cdot^I to $B \oplus i$, the algorithm computes, for a given B , the set $D := (B \oplus i)^I$. This is the lectically next intent, if $B <_i D$ holds, meaning that i is the smallest element in which B and D differ, and $i \in D$.

The method *NextFrequentConcept* adopts this idea and additionally checks if the computed extent $A := (B \oplus i)^I$ fulfills the minimal support criterion before computing the intent $D := A^I$. This is done in line 3 of Algorithm 4.3 by considering the extent A

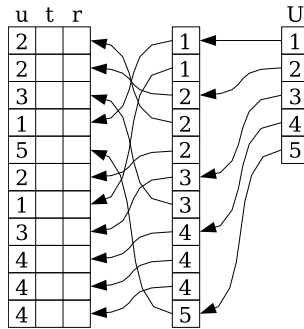


Figure 4.2: Example arrays which provide access to the triples of Y in sorted order of the user dimension.

only if it is large enough.

Taking a closer look on the operator \cdot^I revealed that it demands the computation of several set intersections at a time to gather all common attributes of the given objects (or all common objects of the given attributes, resp.). Since profiling showed that this is the main computational bottleneck of the algorithm, we optimized the computation by first ordering the sets to be intersected by size (with the smallest set first). Then the algorithm recursively intersects the sets with a procedure used for merge-sort (Knuth, 1998). This is possible, since every itemset of the binary context can be accessed as ordered list in the data structure described in the following.

Because two sortings of Y are needed, instead of storing both, we just store the permutations for every order and an additional offset array which provides constant time access to the triples of a given tag, user, or resource. The chosen approach is exemplified for a sorting of the user dimension in Figure 4.2. The array on the left contains the unsorted triples Y of which only the values from U are shown here. The array in the middle describes the permutation which allows us to access the triples in lexicographic order. Finally, the right array contains, for every element $u \in U$, an offset which points to the position in the second array, which points to the first triple of that user in the Y list. Together, all this provides constant time access to the sorted tag-resource pairs of every user.

4.3.3 Performance of the Trias Algorithm

As in the dyadic case, the number of (frequent) tri-concepts may grow exponentially in the worst case. Biedermann (1998b) has shown that the concept tri-lattice of the triadic context of size $n \times n \times n$ where only the main diagonal is empty has size 3^n . In typical applications, however, one is far from this theoretical boundary. Therefore, we focus on empirical evaluations on a large scale real-world dataset.

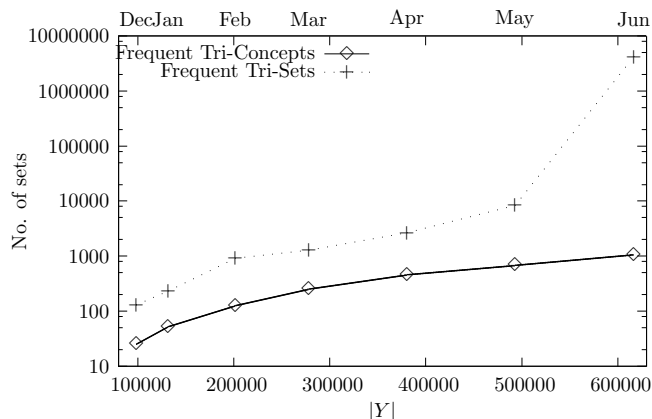


Figure 4.3: The number of frequent tri-sets vs. the number of frequent tri-concepts.

For measuring the runtime and the number of frequent concepts we have evaluated the performance of TRIAS on a snapshot of the Delicious system (which is described in more detail in Section 4.4.1). It consists of all users, tags, resources and tag assignments we could download that were entered to the system on or before June 15, 2004. From this base set we created triadic contexts of monthly snapshots as follows. \mathbb{F}_0 contains all tag assignments performed on or before December 15, 2003, together with the involved users, tags, and resources; \mathbb{F}_1 all tag assignments performed on or before January 15, 2004, together with the involved users, tags, and resources; and so on until \mathbb{F}_6 which contains all tag assignments performed on or before June 15, 2004, together with the involved tags, users, and resources. This represents seven monotonously growing contexts describing the Delicious folksonomy at different points in time. For mining frequent tri-sets and frequent tri-concepts we used minimum support values of $\tau_u := \tau_t := \tau_r := 2$ and measured the run-time of our Java implementation on a dual-core Opteron system with 2 GHz and 8 GB RAM.

Figure 4.3 shows the number of frequent tri-concepts versus the number of frequent tri-sets on the logarithmically scaled y -axis, whereas the x -axis depicts the number of triples in Y – which grows from 98,870 triples in December 2003 to 616,819 in June 2004. It shows a massive increase of frequent tri-sets in June 2004 with only a modest growth of the number of frequent tri-concepts. This difference results from the fact that more and more users appear and start to agree on a common vocabulary, which leads to more frequent tri-concepts with larger volumes from June 2004 on. Such large concepts (like those shown in Table 4.5) contain combinatorially many frequent tri-sets.

One can observe that the number of frequent tri-sets of every snapshot is always at least one magnitude of size larger than the number of frequent tri-concepts. Consequently, computing frequent tri-sets is much more demanding than computing frequent tri-concepts – without providing any additional information.

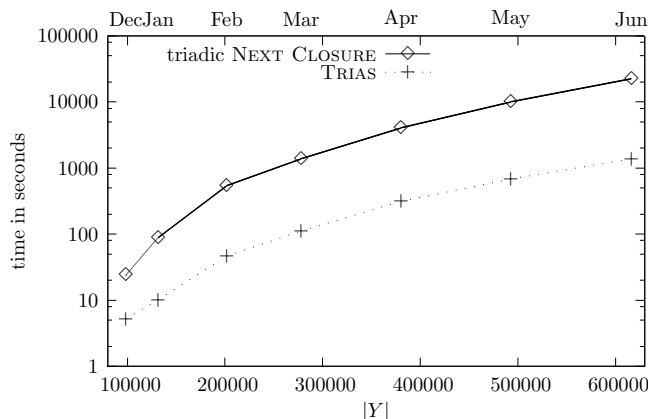


Figure 4.4: A comparison of the runtime of the triadic NEXT CLOSURE and the TRIAS algorithm on the Delicious datasets.

A comparison of the speed improvement gained from not computing all tri-concepts with an algorithm like the triadic version of NEXT CLOSURE and afterwards pruning the non-frequent concepts but using the TRIAS algorithm for directly mining frequent tri-concepts is shown in Figure 4.4. The logarithmically scaled y -axis depicts the runtime of the algorithms in seconds while the x -axis shows again the size of the Y relation. One can see that computing all tri-concepts is more than one magnitude more expensive than mining only the frequent tri-concepts one is interested in.

With these observations we conclude that the TRIAS algorithm provides an efficient method to mine frequent tri-concepts in large scale conceptual structures.

4.4 Qualitative Evaluation

To evaluate the usefulness of TRIAS for discovering shared conceptualizations, we have applied the algorithm on three real-world data sets: the social bookmarking system Delicious, the IT Baseline Security Manual of the German Federal Office for Information Security, and the collection of publications in our social reference management system BibSonomy.

4.4.1 Conceptual Clustering of the Delicious Folksonomy

First, we have analyzed the popular social bookmarking system Delicious⁹ with our approach. Delicious is a server-based system with a simple-to-use interface that allows users to organize and share bookmarks on the web. It enables its users to store for each URL, in addition to the tags assigned to it, a description and a note.

⁹<http://www.delicious.com/>

For detecting communities of users which have the same tagging behaviour (and thus share their conceptualizations), we ran the TRIAS algorithm on a Delicious snapshot consisting of all users, resources, tags and tag assignments we could download that were entered to the system on or before June 15, 2004 (Hotho et al., 2006b). The resulting folksonomy consists of $|U| = 3,301$ users, $|T| = 30,416$ different tags, $|R| = 220,366$ resources (URLs), which are linked by $|Y| = 616,819$ triples.

As a first step, we ran TRIAS on the dataset without restricting the minimum supports (i. e., $\tau_u := \tau_t := \tau_r := 0$). The resulting concept tri-lattice consists of 246,167 tri-concepts. We then investigated the concepts which contain two or more users, tags and resources, i. e., with minimal support values of $\tau_u := \tau_t := \tau_r := 2$. There were 1,062 such tri-concepts.¹⁰

Figure 4.5 shows three examples. The first of them shows that the two users *bibi* and *poppy* have assigned the three tags *women*, *cinema*, and *film* to all the ten listed web pages, which are all about women in movies or women in the movie industry.

The two lower tri-concepts show that different tri-concepts with the same extent can co-exist.¹¹ The first of them shows that the two users *fischer* and *gnat* agree (implicitly) in their assignments of the tags *css*, *web*, and *design* to the four listed URLs, while the users *angusf* and *carlomazza* agree in assigning the same tags to five completely different URLs. When inspecting the corresponding web pages, one finds out that the content of all resources is indeed very much related. These two related tri-concepts may be exploited further for extracting relations between tags or for recommending to all of the four users to study the posts of the other three.

Next, we wanted to study in more detail shared conceptualizations around the tags *css*, *web*, and *design*. To this end, we computed the concept lattice that is shown in Figure 4.6. Its formal context (G, M, I) was constructed as follows. Its set G of objects was extracted from the set of all resources by selecting all those resources which were tagged with at least one of these three tags by at least $k_1 \in \mathbb{N}$ users. The set M contains all tags. A tag $t \in M$ is defined to be related to a resource $r \in G$ (i. e., $(r, t) \in I$) iff

$$\frac{|\{u \in U \mid (u, t, r) \in Y\}|}{|\{u \in U \mid \exists r' \in R: (u, t, r') \in Y\}|} \geq k_2,$$

for a given $k_2 \in [0, 1]$.

For our analysis, we have set $k_1 = 5$. This means that a resource was considered only if at least five users assigned it to at least one of the tags *css*, *web*, and *design*. This resulted in 575 resources. The second pruning parameter was set to $k_2 = 0.5$, i. e., at

¹⁰Larger thresholds did not provide any results any more. This comes from the fact that we took a rather early snapshot of Delicious, where the numbers of users, tags, and resources were still rather small. See also Section 4.3.3.

¹¹This is in contrast to the situation in the dyadic case, where equality in one dimension implies equality in the other one.

<i>A</i>	bibi poppy
<i>B</i>	women cinema film
<i>C</i>	http://www.reelwomen.org/ http://www.people.virginia.edu/~pm9k/libsci/womFilm.html http://www.lib.berkeley.edu/MRC/womenbib.html http://www.beaconcinema.com/womfest/ http://www.widc.org/ http://www.wftv.org.uk/home.asp http://www.feminist.com/resources/artspeech/media/femfilm.htm http://www.duke.edu/web/film/pioneers/ http://www.womenfilmnet.org/index.htm#top http://208.55.250.228/
<i>A</i>	fischer gnat
<i>B</i>	css design web
<i>C</i>	http://www.quirksmode.org/ http://webhost.bridgew.edu/etribou/layouts/ http://www.picment.com/articles/css/funwithforms/ http://www.alistapart.com/articles/sprites/
<i>A</i>	angusf carlomazza
<i>B</i>	css design web
<i>C</i>	http://www.positioniseverything.net/index.php http://www.fu2k.org/alex/css/layouts/3Col_NN4_FMFM.mhtml http://glish.com/css/home.asp http://www.maxdesign.com.au/presentation/process/index.cfm http://unraveled.com/projects/css_tabs/

Figure 4.5: Examples of frequent tri-concepts of Delicious.

least half of the users who considered a resource had to use a particular tag, otherwise the tag was not assigned to the resource. This resulted in a relatively sparse assignment which reflects only rather strong shared conceptualizations. This way, only 22 tags were assigned to at least one resource; and only 297 out of the 575 resources received at least one tag.

The resulting concept lattice is displayed in Figure 4.6. For better readability, we pruned from it the tags *rest*, *cms*, *wiki*, *xml*, *fonts*, *wordpress*, *google*, *search*, *color*, *art*, and *music*. These tags formed singletons (i. e., separate nodes that were connected only to the top and to the bottom element of the lattice) with one or two resources each.

Each node in the diagram is a formal concept according to the definition in Sec-

tion 4.2.2, i. e., a pair (A, B) where A is its extent (all resources belonging to it), and B is its intent (all tags belonging to it). In the diagram, the extent of a concept consists of all resources attached to the concepts or to any of its sub-concepts; and the intent consists of all tags that are attached to the concept or to any of its super-concepts. The left-most concept, for instance, has the two URLs starting with `http://www.fiftyfoureleven...` as extent, and the set $\{php, css\}$ of tags as intent. The top node represents the concept (G, G^I) , and the bottom node the concept (M^I, M) .

The diagram shows that most agreement exists for the usage of the tag *css*, as it was assigned (according to our majority vote with the k_2 threshold) to 235 resources, while *web* was assigned to only 14 resources, and *design* to 31 resources. Apparently, the latter are too general or polysemous terms to reach a large agreement about their usage.

The resulting concept lattice could now be used for building a concept hierarchy. It suggests to the ontology engineer, e. g., to model *architecture* as a sub-concept of *design*. Another use of the concept lattice is a collaborative filtering approach to web search. When a user is, for instance, searching for ‘*web design*’, the system could recommend him the web pages `http://www.alistapart.com/articles/elastic` and `http://9rules.com/version2/`.

4.4.2 IT Baseline Protection Manual

To illustrate another use of iceberg tri-lattices, we now focus on a non-folksonomy application. The IT Baseline Security Manual (‘Grundschutzhandbuch’) of the German Federal Office for Information Security (Ger, 2003) provides a description of a threat scenario and standard security measures for typical IT systems, and detailed descriptions of safeguards to assist with their implementation.¹²

Unlike a folksonomy, this manual has not been set up by an open group of users, but by a closed group of experts of the federal office. The manual has thus carefully been designed by domain specialists, and can be considered as an ontology (a formal specification of the shared conceptualization of the experts of the federal office) – structured in form of a triadic context. Here, we use our knowledge discovery approach not for discovering a shared conceptualization, but for analysing it. Even though the manual is smaller than a typical folksonomy resulting from a social bookmarking system, it is still by far too large to be analyzed without technical support.

The core data of the manual forms a triadic context (U, T, R, Y) . We consider as objects U the 66 IT components, as attributes T the 377 listed threats, and as conditions R the 912 safeguards. They are related by 5,680 triples.¹³

From this dataset, we have computed the iceberg concept lattice for $\tau_u = \tau_t = \tau_r = 3$. Its visualization in Figure 4.7 follows the conventions introduced in (Lehmann and Wille, 1995). The five nodes in the middle are the five resulting frequent tri-concepts. The sets

¹²The online version of the manual is available at `http://www.bsi.de/gshb/`.

¹³See (Dau and Wille, 2001; Söll, 1998; Wille and Zickwolff, 2000) for other analyses of this dataset.

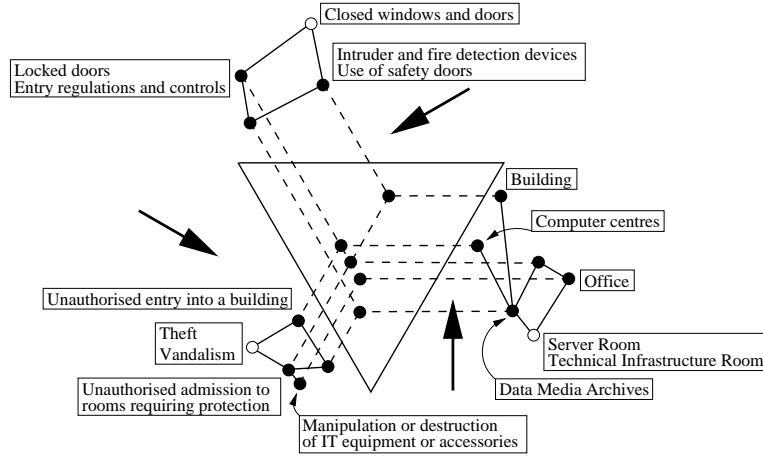


Figure 4.7: All frequent tri-concepts of the IT Baseline Security Manual for $\tau_u = \tau_t = \tau_r = 3$.

of users, tags, and resources (in this example IT components, threats, and safeguards) composing a tri-concept can be read off the three sides of the triangle. There, three Hasse diagrams display the three quasi-orders \lesssim_1 , \lesssim_2 , and \lesssim_3 as introduced in Section 4.2.3. The arrows guide the reader to the larger elements of each quasi-order. Each node in a hierarchy represents the set containing the labels attached to it plus all labels below. The empty nodes are not part of the quasi-order. They are just used to be able to place each label once only. In the IT components hierarchy on the right, for instance, the leftmost node represents the set $\{\text{Computer Centres}, \text{Data Media Archives}, \text{Server Room}, \text{Technical Infrastructure Room}\}$.

A node in the middle of the diagram represents then the tri-concept consisting of the three components it projects to. The left-most tri-concept, for instance, is the tri-concept $(\{\text{Computer Centres}, \text{Server Room}, \text{Data Media Archives}, \text{Technical Infrastructure Room}\}, \{\text{Unauthorised entry into a building}, \text{Theft}, \text{Vandalism}\}, \{\text{Locked doors}, \text{Entry regulations and controls}, \text{Closed windows and doors}\})$.

The three corners of the inner triangle are not realized (as there are no nodes on them). They stand for the tri-sets (\emptyset, T, R) , (U, \emptyset, R) , and (U, T, \emptyset) , resp., and are only realized if the first, second, or third threshold is set to zero.

The manual distinguishes seven classes of IT components, like *Networked Systems* and *Telecommunications*. The fact that all components that occur in the most frequent tri-concepts (i. e., the six components in the right-most hierarchy) are of the *Infrastructure* class indicates that this class was modeled with the highest level of detail. Surprisingly, it surpasses more typical IT classes like the two mentioned above.

For having a closer look, we decrease the minimum thresholds, e. g., to $\tau_u = 3, \tau_t = 2, \tau_r = 3$. The resulting tri-lattice is shown in Figure 4.8. It contains the previous five

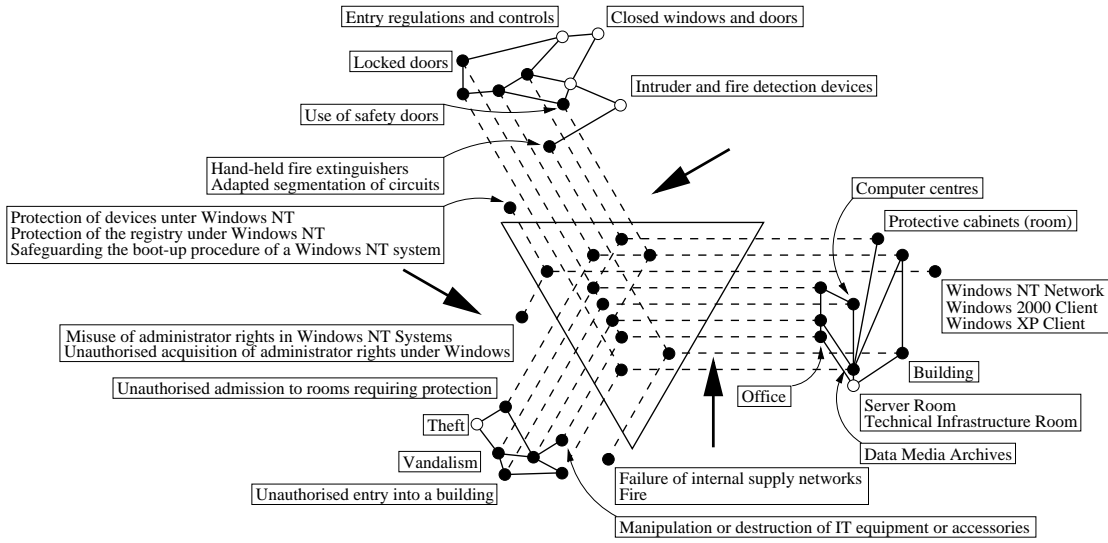


Figure 4.8: All frequent tri-concepts of the IT Baseline Security Manual for $\tau_u = 3$, $\tau_t = 2$, $\tau_r = 3$.

tri-concepts plus five new ones. We see that again the major contribution comes from the *Infrastructure* class, which is now extended by *Protective cabinets*. Additionally, some more of the combinations of these components became frequent, indicated by the additional nodes in the right hierarchy.

With the decreasing thresholds, the lower left hierarchy grew as well. It contains now additionally four threats in two separated nodes. These nodes are not comparable (in terms of set inclusion) with the already existing nodes. The threats in the lower one of them – *Failure of internal supply networks*, *Fire* – are extending the list of threats against the *Infrastructure* class via the IT component *Building*. The upper hierarchy shows the safeguards against these new threats: *Hand-held fire extinguishers* and *Adapted segmentation of circuits*.

The threats in the uppermost isolated node of the lower left hierarchy – *Misuse of administrator rights [...]* and *Unauthorised acquisition [...]* – belong to a new class of IT components, as they are related to the new isolated node with three Windows operating systems in the right diagram. The safeguards against these threats are listed at the isolated node in the upper diagram. The IT components that seem to be endangered secondmost are thus – after IT infrastructure rooms – Windows operating systems. At least they are modeled with greater detail as other operating systems that show up when decreasing the thresholds further.

If we decrease the minimum thresholds further, we can discover more and more details, until we finally reach with $\tau_u = \tau_t = \tau_r = 0$ all 3,751 tri-concepts of this dataset.

4.4.3 Conceptual Analysis of the BibSonomy Publication Data

We conclude the list of applications with an analysis of the publication management part of BibSonomy (cf. Chapter 3). We first made a snapshot of BibSonomy’s publication posts, including all posts made until November 23, 2006 at 13:30 CET. From the snapshot we excluded the publication posts from the DBLP computer science bibliography¹⁴ 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 about shared conceptualizations. Similarly we excluded all tag assignments with the tag *imported* and all publication posts which exclusively have this tag, because it is automatically assigned to all posts which were added by one of the import functions. The resulting snapshot contains $|Y| = 44,944$ tag assignments built by $|U| = 262$ users, containing $|R| = 11,101$ publication references tagged with $|T| = 5,954$ distinct tags.

TRIAS needed 75 minutes on a 2 GHz AMD Opteron machine to compute all 13,992 tri-concepts of this dataset. Among those there are 12,659 tri-concepts which contain only one user, representing the individual conceptualizations of the users. (These could be used to present personal concept hierarchies by means of dyadic Hasse diagrams.) The remaining 1,333 tri-concepts thus all contain at least two users and therefore represent shared concepts. To further analyze these concepts, we take a closer look on the tri-concepts which contain at least three users, two tags and two publication entries (i. e., with minimal support values $\tau_u = 3$, $\tau_t = 2$, $\tau_r = 2$). Each of these 21 tri-concepts expresses the fact that all of its users tagged all its publications with all its tags.

The diagram in Figure 4.9 shows the triadic concept lattice of all these 21 tri-concepts. The titles of the publications in the figure are substituted by numbers for space reasons. The corresponding titles can be found in Table 4.1, the full bibliographic information was tagged in BibSonomy (after the evaluation) with the tag *trias_example*.¹⁵ As in Figures 4.7 and 4.8, the 21 nodes in the center of the triangle represent the 21 frequent tri-concepts. The sets of users, tags, and resources composing a tri-concept can be read off the three sides of the triangle.

Table 4.1: The mapping of publication IDs to publication titles.

ID	Publication Title
1	A Finite-State Model for On-Line Analytical Processing in Triadic Contexts
2	Annotation and Navigation in Semantic Wikis
3	A Semantic Wiki for Mathematical Knowledge Management
4	BibSonomy: A Social Bookmark and Publication Sharing System
5	Bringing the “Wiki-Way” to the Semantic Web with Rhizome
6	Building and Using the Semantic Web

¹⁴<http://www.informatik.uni-trier.de/~ley/db/>

¹⁵http://www.bibsonomy.org/group/kde/trias_example?items=50

Table 4.1: The mapping of publication IDs to publication titles.

ID	Publication Title
7	Conceptual Clustering of Text Clusters
8	Content Aggregation on Knowledge Bases using Graph Clustering
9	Creating and using Semantic Web information with Makna
10	Emergent Semantics in BibSonomy
11	Explaining Text Clustering Results using Semantic Structures
12	Harvesting Wiki Consensus - Using Wikipedia Entries as Ontology Elements
13	Information Retrieval in Folksonomies: Search and Ranking
14	KAON – Towards a Large Scale Semantic Web
15	Kaukolu: Hub of the Semantic Corporate Intranet
16	Kollaboratives Wissensmanagement
17	Learning with Semantic Wikis
18	Mining Association Rules in Folksonomies
19	On Self-Regulated Swarms, Societal Memory, Speed and Dynamics
20	Ontologies improve text document clustering
21	Proceedings of the First Workshop on Semantic Wikis – From Wiki To Semantics
22	Proc. of the European Web Mining Forum 2005
23	Semantic Network Analysis of Ontologies
24	Semantic Resource Management for the Web: An ELearning Application.
25	Semantic Web Mining
26	Semantic Web Mining and the Representation, Analysis, and Evolution of Web Space
27	Semantic Web Mining for Building Information Portals (Position Paper)
28	Social Bookmarking Tools (I): A General Review
29	Social Bookmarking Tools (II). A Case Study – Connotea
30	Social Cognitive Maps, Swarm Collective Perception and Distributed Search on Dynamic Landscapes
31	SweetWiki : Semantic Web Enabled Technologies in Wiki
32	Text Clustering Based on Background Knowledge
33	The ABCDE Format Enabling Semantic Conference Proceedings
34	The Courseware Watchdog: an Ontology-based tool for Finding and Organizing Learning Material
35	Towards a Wiki Interchange Format (WIF) – Opening Semantic Wiki Content and Metadata
36	Towards Semantic Web Mining
37	TRIAS - An Algorithm for Mining Iceberg Tri-Lattices
38	Usage Mining for and on the Semantic Web (Book)
39	Usage Mining for and on the Semantic Web (Workshop)
40	Wege zur Entdeckung von Communities in Folksonomies
41	WordNet improves text document clustering

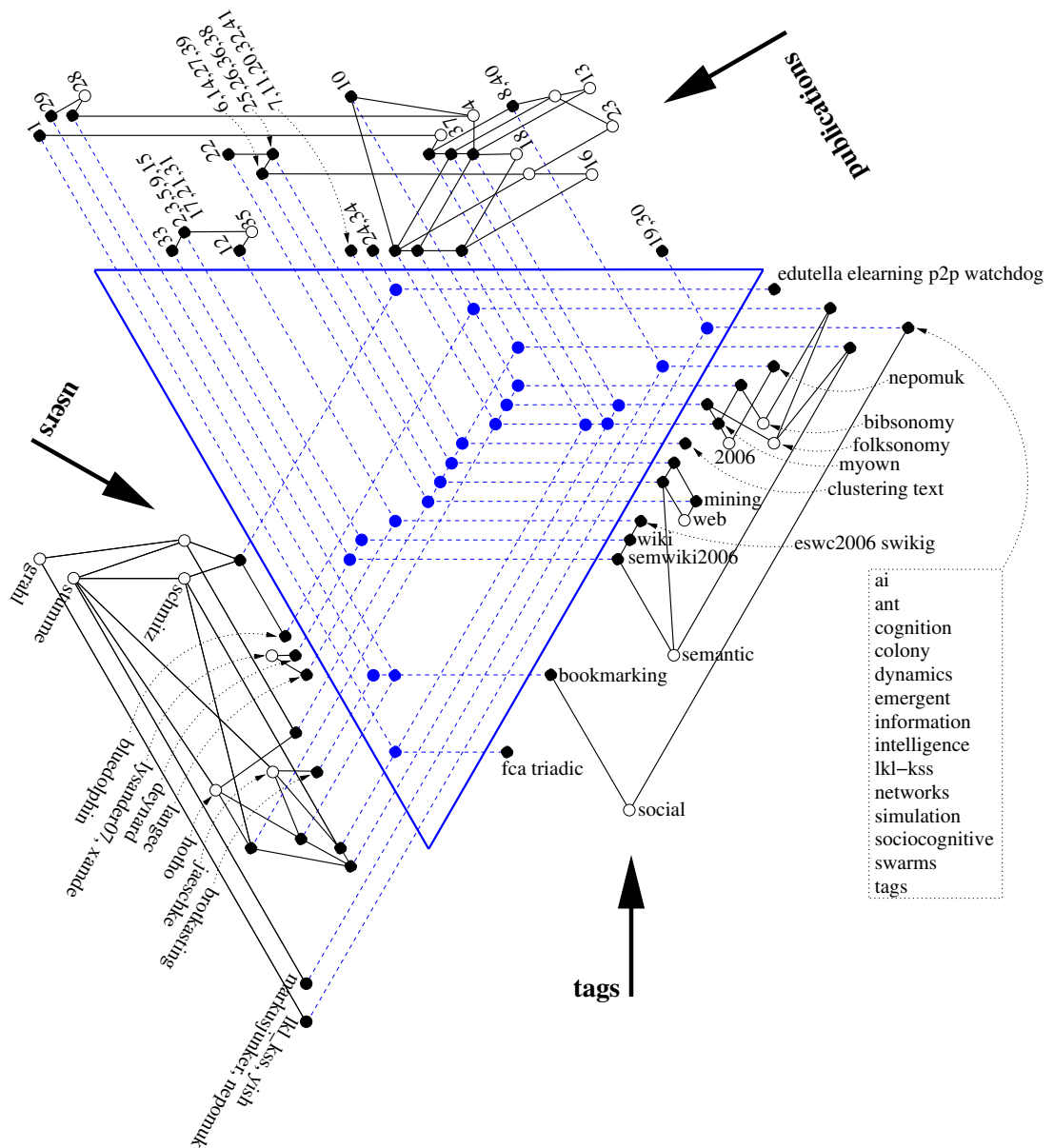


Figure 4.9: All frequent tri-concepts of the BibSonomy publications for $\tau_u = 3$, $\tau_t = 2$, $\tau_r = 2$.

For instance, the lower-most node in the triangle represents the tri-concept consisting of the set $\{jaeschke, schmitz, stumme\}$ of users, the set $\{fca, triadic\}$ of tags, and the set $\{1, 37\}$ of resources. Similarly, the node in the user hierarchy labeled *brotkasting*

represents not only the user *brotkasting* but also all users in nodes laying below this node. Therefore the users *jaeschke* and – since it is located below both *brotkasting* and *jaeschke – stumme* also belong to this node. Note that it fulfills thus the minimal support constraint $\tau_u = 3$ for the users.

A closer look on the tag hierarchy reveals the content of the most central publications in the system. The tag *social* co-occurs with most of the tags. On the level of generality defined by the τ thresholds, this tag is (together with the tags *ai* (meaning Artificial Intelligence), ..., *tags*) assigned by the users *lklkss* and *yish* to the publications 19 and 30, (together with the tag *bookmarking*) by the users *hotho*, *jaeschke*, *stumme* to the publications 4 and 28, and (again together with the tag *bookmarking*) by the users *brotkasting*, *jaeschke*, *stumme* to the publications 28 and 29. The tags as well as the corresponding publication titles indicate that the two sets of users $\{lklkss, yish\}$ and $\{brotkasting, hotho, jaeschke, stumme\}$ form two sub-communities which both work on social phenomena in the Web 2.0, but from different perspectives.

A second topical group is spanned by the tag *semantic*, which occurs in three different contexts. The first is on semantic wikis, which correlates with the isolated group $\{2, \dots, 31, 12, 33, 35\}$ of publications, and the – equally isolated – group $\{lysander07, xamde, deynard, langec\}$ of users. The second context in which the tag *semantic* occurs is on Semantic Web Mining, being connected by the users $\{grahl, hotho, stumme\}$ with different combinations of the additional tags *web* and *mining* to the publications 6, 14, 22, 25, 26, 27, 36, 38, and 39. These assignments are witnessed by the three tri-concepts in the very middle of the diagram. On the same line are two more tri-concepts, which indicate that these users are also interested in *text clustering* and in *nepomuk* (the European project Nepomuk¹⁶). The third context in which the tag *semantic* occurs is in combination with *folksonomy*. This provides a link to the group $\{2006, myown, nepomuk, bibsonomy, folksonomy\}$ of tags which are used by researchers to describe their own publications.

Two more topical groups can be found at the top and bottom of the tags quasi-order. One is related to a Peer-to-Peer eLearning application, and the other to triadic Formal Concept Analysis.

Since the diagram shows the *frequent* tri-concepts only, we cannot deduce from the absence of a relationship that two objects are not related at all. When the thresholds are lowered, links between the topical islands discussed above will show up.

Concluding, we see that iceberg tri-concept lattices provide a means for exploring the flat structure of folksonomies – just as iceberg concept lattices in the dyadic case. One may be surprised by the relatively small numbers of frequent tri-concepts. This shows – just as in the dyadic case – that the closeness condition provides a strong criterion for pruning the result set without loss of information.

¹⁶<http://nepomuk.semanticdesktop.org/>

4.5 Neighborhoods of Triadic Concepts

The line diagrams used in the previous section to visualize (iceberg) tri-lattices (cf. Figures 4.7 to 4.9) have been drawn manually since there is no tool support for (semi)-automatic drawing. Unfortunately, drawing the diagrams is a tricky task which requires a lot of time and aptitude. For larger tri-lattices it is mostly infeasible to draw such a diagram, in particular, if the triadic context satisfies the *tetrahedron condition* (Biedermann, 1997b) or violates the *Thomson condition* (Wille and Zickwolff, 2000). Thus, there is a need for simpler visualization metaphors.

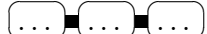
As a first step to allow for automatic visualization of tri-lattices, we here define a binary relation which can be directly visualized as an undirected graph with standard tools like Graphviz (Ellson et al., 2004) or Pajek (de Nooy et al., 2005).

Definition 4.8 *Given a tri-context $\mathbb{F} := (U, T, R, Y)$ and its corresponding (iceberg) tri-lattice $\mathfrak{B}(\mathbb{F})$, two tri-concepts $\mathbf{a}_1 = (A_1, B_1, C_1)$ and $\mathbf{a}_k = (A_k, B_k, C_k)$ are direct neighbors, if $A_1 = A_k$ or $B_1 = B_k$ or $C_1 = C_k$ holds. We then write $\mathbf{a}_1 \sim \mathbf{a}_k$ and say \mathbf{a}_1 is a direct neighbor of \mathbf{a}_k .*

\mathbf{a}_1 and \mathbf{a}_k are neighbors, if there exist tri-concepts $\mathbf{a}_2, \dots, \mathbf{a}_{k-1} \in \mathfrak{B}(\mathbb{F})$ with $\mathbf{a}_i \sim \mathbf{a}_{i+1}$ (for $i = 1, \dots, k-1$). We then write $\mathbf{a}_1 \sim \mathbf{a}_k$ and say \mathbf{a}_1 is a neighbor of \mathbf{a}_k .

It is easy to see that $\sim \subseteq \sim$ holds and that \sim is an equivalence relation on $\mathfrak{B}(\mathbb{F})$. Thus, we can consider the equivalence class $[\mathbf{a}] := \{\mathbf{b} \mid \mathbf{a} \sim \mathbf{b}\}$ of a tri-concept $\mathbf{a} \in \mathfrak{B}(\mathbb{F})$ which we call the *neighborhood* of \mathbf{a} . The neighborhood contains all tri-concepts one can ‘reach’ in the tri-lattice diagram by following the lines from the tri-concept.

We now define the *neighborhood graph* (V, E) of a neighborhood, which has as its vertices $v \in V$ the tri-concepts of the neighborhood. Two concepts are connected by an edge $e \in E$, if they are direct neighbors. I. e., for a tri-concept \mathbf{a} , its neighborhood graph is $([\mathbf{a}], \sim \cap ([\mathbf{a}] \times [\mathbf{a}]))$. Looking at neighborhoods from a graph theoretical perspective, they are exactly the connected components of the graph $(\mathfrak{B}(\mathbb{F}), \sim)$. Because (V, E) is a standard graph, it can be automatically drawn by tools like Graphviz.

In Figure 4.10 the neighborhood graph of the tri-concept $(\{hotho, schmitz, stumme\}, \{2006, bibsonomy, myown\}, \{4, 10\})$ from Figure 4.9 is shown. The darkness of an edge (and additionally its orientation) depicts the dimension in which the endpoint concepts equal: black (0°) for the tag dimension, light grey (60°) for the user dimension, and dark grey (120°) for the resource dimension. Thus, tri-concepts which are located on the same (horizontal) black line share the same tags – in this case the two concepts $(\{hotho, schmitz, stumme\}, \{2006, folksonomy, myown\}, \{4, 13, 16, 18\})$ and $(\{hotho, jaeschke, schmitz, stumme\}, \{2006, folksonomy, myown\}, \{4, 13, 18\})$, for example. Some edges which can be inferred by reflexivity or transitivity “along one dimension” were not drawn to simplify the graph. E. g., the edge connecting the leftmost concept and the rightmost concept from the three tri-concepts on the black line  is omitted.

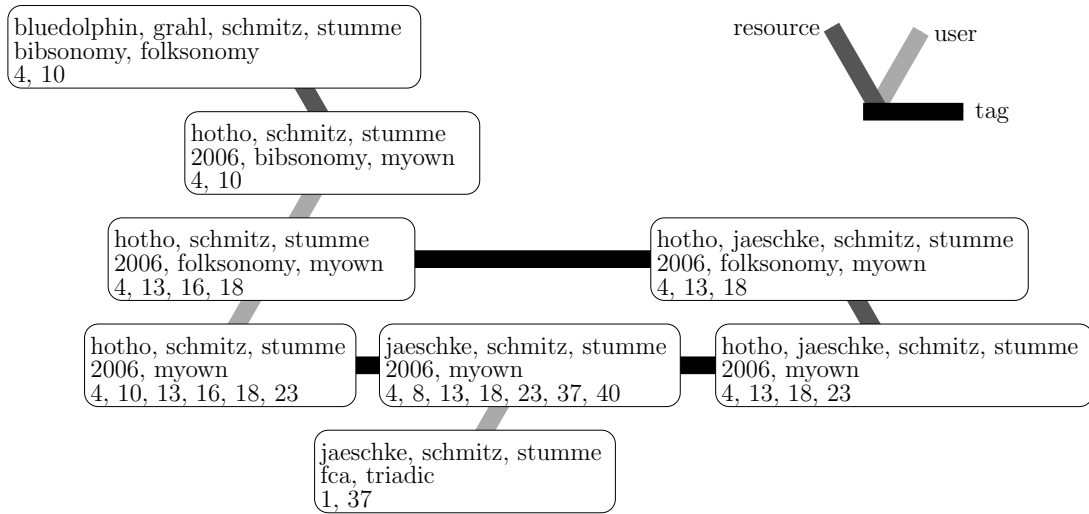


Figure 4.10: The neighborhood graph of the neighborhood $[(\{hotho, schmitz, stumme\}, \{2006, bibsonomy, myown\}, \{4, 10\})]$.

The main benefit of the visualization metaphor for tri-lattices presented here over the usual diagrams is the ability to automatically draw neighborhood graphs using standard tools for graph visualization, even for large tri-lattices. However, one loses the depiction of the hierarchical nature of the lattice as given by the quasi-orders. This must not be a disadvantage, since the simpler diagrams might be easier to read and understand for people without a background in triadic FCA. As a consequence, one could embed such diagrams in collaborative tagging systems to visualize shared conceptualizations and aid their navigation.

To conclude this section with a second example, we again look at the IT Baseline Security Manual context introduced in Section 4.4.2. With minimal support values of $\tau_u = \tau_t = \tau_r = 2$, we obtain an iceberg tri-lattice with 151 tri-concepts, arranged in 13 neighborhoods. The largest neighborhood contains 77, the second largest neighborhood 31 tri-concepts. The complete neighborhood graph is shown in Figure 4.12. Since this graph is too large to be properly readable in print, we here focus on the second largest component of that tri-lattice (Figure 4.11).

Both graphs were drawn automatically using Pajek, only minimal manual work was necessary to improve the labeling. Again, the darkness of an edge depicts which of the dimensions *IT components* (black), *threats* (dark grey), and *safeguards* (light grey) in adjacent concepts are equal. Additionally, the width of an edge is equivalent to the size of the corresponding sets, e.g., the vertical black edge in the center of Figure 4.11 connecting the two tri-concepts $(\{Gebäude, Serverraum, Datenträgerarchiv, Raum für technische Infrastruktur\}, \{Unbefugter Zugriff zu schutzbedürftigen Räumen,$

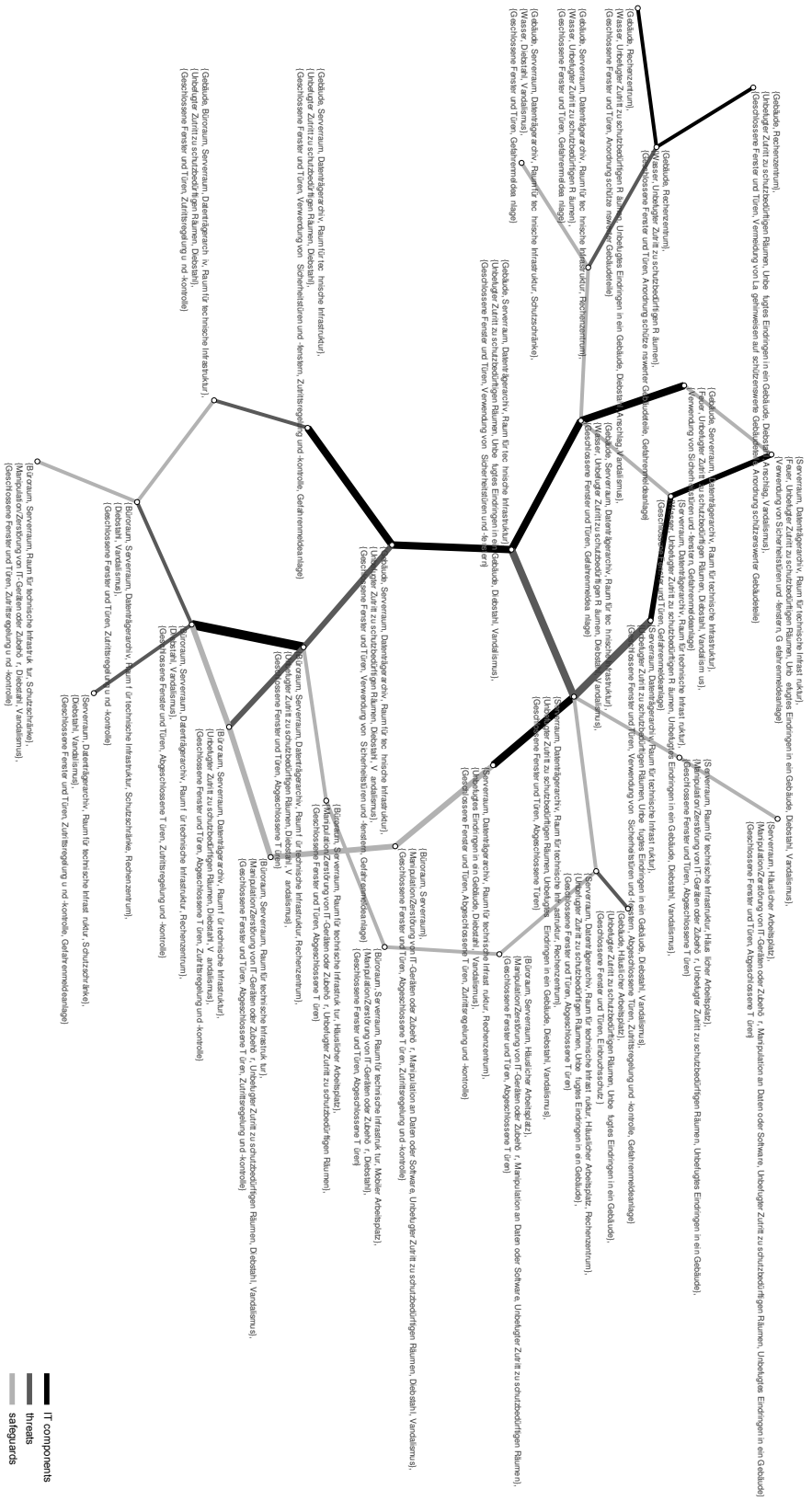


Figure 4.11: The second largest neighborhood of the IT Baseline Security Manual iceberg concept lattice with minimal support $\tau_u = \tau_r = \tau_s = 2$.



Figure 4.12: The neighborhood graph of the IT Baseline Security Manual iceberg concept lattice with minimal support $\tau_u = \tau_t = \tau_r = 2$. It consists of 151 concepts in 13 neighborhoods, with the largest neighborhood (top left) mostly modeling threats and safeguards around IT components for the *Windows* operating system. The second largest neighborhood (top right) is focused on equipment from the *Infrastructure* IT components class like *server rooms* or *offices*.

Unbefugtes Eindringen in ein Gebäude, Diebstahl, Vandalismus}, {*Geschlossene Fenster und Türen, Verwendung von Sicherheitstüren und -fenstern*}) and ({*Gebäude, Serverraum, Datenträgerarchiv, Raum für technische Infrastruktur*}, {*Unbefugter Zutritt zu schutzbedürftigen Räumen, Diebstahl, Vandalismus*}, {*Geschlossene Fenster und Türen, Verwendung von Sicherheitstüren und -fenstern, Gefahrenmeldeanlage*}) is black with a relative width of four because the two concepts have the same four IT components as extent.

The IT components in this neighborhood are all from the *Infrastructure* class, e. g., *server room*, or *office*. The graph shows a remarkable variety in the differences between two tri-concepts. Often, there are only subtle changes between adjacent concepts like in the ‘triangle’ in the bottom center of the graph, where each of the three edges has a different darkness. The three concepts vary only in one element at a time. Thus, this triangle might suggest to the ontology engineer to adapt the ontology such that the three concepts are merged into one, since the safeguard *Zutrittsregelung und -kontrolle* might help against the threat of *Unbefugter Zutritt zu schutzbedürftigen Räumen* also for the IT component *Rechenzentrum*.

4.6 Conclusion

In this chapter, we have presented a formal definition of the problem of mining all frequent tri-concepts, and an efficient algorithm for its solution. We have empirically studied the performance of the algorithm, and have presented three real-world applications. This work opens a series of challenging tasks for future research.

1. An important issue for the presentation of the results is the development of a visualization metaphor to display small, medium, and large (frequent) concept tri-lattices, and to provide efficient means for navigating and browsing them. First steps in this direction have been taken in Section 4.5 with the introduction of neighborhoods of triadic concepts.
2. Continuing the research on association rules, a natural next step would be the exploitation of ‘triadic association rules’, combining thus the developments in triadic FCA and association rule mining. Some ideas in this direction are sketched in Section 9.3.3.
3. The natural next step after discovering shared conceptualizations would be to formalize them in an ontology. Therefore, our approach could be extended to an ontology learning application.
4. TRIAS, when using NEXT CLOSURE as underlying algorithm to compute binary concepts, can be parallelized in a natural way. Since NEXT CLOSURE computes concepts in lexic order, we can divide the search space and handle the parts in

parallel. Therefore, one needs to exchange line 2 of Algorithm 4.1 on page 51 with a call to $\text{NextFrequentConcept}((A, I), (U, T \times R, \tilde{Y}), \tau_u)$ with I set to the element before the first element of the search interval. How the division into small tasks can be done in a clever way is an open question.

Chapter 5

Tag Recommendations

Recommender systems suggest new or potentially interesting items to users. In the context of collaborative tagging systems, this could be other users or resources or – as discussed in this chapter – tags to annotate a resource. We quantitatively evaluate the tag recommendation performance of three classes of algorithms, each on three real world folksonomy datasets. This chapter is based on work published in (Jäschke et al., 2007b) and (Jäschke et al., 2008c).

5.1 Introduction

To support users in the tagging process and to expose different facets of a resource, most collaborative tagging systems offered some kind of tag recommendations already at an early stage. Delicious, for instance, had a tag recommender in June 2005 at the latest,¹ and also included resource recommendations.² However, no algorithmic details were published. We assume that these recommendations basically provide those tags which were most frequently assigned to the resource (called *most popular tags by resource* in the sequel).

Most recommender systems are typically used to call users' attentions to new objects they do not know yet and have not rated already in the past. This is often due to the fact that there is no repeat-buying in domains like books, movies, music etc. in which these systems typically operate. In social bookmarking systems, on the contrary, re-occurring tags are an essential feature for structuring the knowledge of a user or a group of users, and have to be considered by a tag recommender. This means that the fact that a tag already has been used to annotate a resource does not exclude the possibility of recommending the same tag for a different resource of the same user.

Overall, recommending tags can serve various purposes, such as: increasing the chance of getting a resource annotated, reminding a user what a resource is about and consolidating the vocabulary across the users. Furthermore, as Sood et al. (2007) point out, tag recommenders lower the effort of annotation by changing the process from a *generation* to a *recognition* task, i. e., rather than “inventing” tags the user only needs to find and click a recommended tag.

¹http://www.socio-kybernetics.net/saurierduval/archive/2005_06_01_archive.html

²http://blog.delicious.com/blog/2005/08/people_who_like.html

In this chapter, we evaluate a tag recommender based on Collaborative Filtering (introduced in Section 5.3.1), a graph-based recommender using our ranking algorithm FolkRank (see Section 5.3.2), and several simpler approaches based on tag counts (Section 5.3.3). In Section 5.4, we discuss the computational costs of the different algorithms. The quality of the resulting recommendations is evaluated on three real world folksonomy datasets from Delicious, BibSonomy and Last.fm (Sections 5.5 and 5.6). In the following we start with recalling the basics and discussing related work.

5.2 Problem Definition and Related Work

In this section we define the problem of tag recommendation in folksonomies and briefly discuss the used relevance criterion. We close with an overview on related work in the field of tag recommendation.

5.2.1 Problem Definition

Recommender systems (RS) in general recommend interesting or personalized information objects to users based on explicit or implicit ratings. Usually RS predict ratings of objects or suggest a list of new objects that the user hopefully will like the most. 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 $\tilde{T}(u, r)$ of tags.³ 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. Thus, in Section 5.5 we perform a *quantitative evaluation* which measures how good the recommended tags are compared to the choice of the user.

5.2.2 Related Work

The topic of tag recommendations in social bookmarking systems has attracted quite a lot of attention in the last years. The existent approaches usually lay in the collaborative filtering and information retrieval areas. In (Mishne, 2006; Byde et al., 2007), algorithms for tag recommendations are devised based on content-based filtering techniques. Xu et al. (2006b) identify properties of good tag recommendations like high coverage of multiple facets, high popularity, or least-effort and introduce a collaborative tag suggestion

³We are using the notation introduced in Section 2.3.

approach based on the HITS algorithm (Kleinberg, 1999). A goodness measure for tags, derived from collective user authorities, is iteratively adjusted by a reward-penalty algorithm. Benz et al. (2006) present a collaborative approach for bookmark classification based on a combination of nearest-neighbor-classifiers. There, a keyword recommender plays the role of a collaborative tag recommender, but it is just a component of the overall algorithm, and therefore there is no information about its effectiveness alone. Basile et al. (2007) suggest an architecture of an intelligent recommender tag system; Vojnovic et al. (2007), try to imitate the learning of the true popularity ranking of tags for a given resource during the assignment of tags by users. In (Firan et al., 2007; Xu et al., 2006a; Tso-Sutter et al., 2008) the problem of tag-aware resource recommendations is investigated. The standard tag recommenders, in practice, are services that provide the most-popular tags used for a particular resource. This is usually done by means of tag clouds where the most frequent used tags are depicted in a larger font or otherwise emphasized.

The approaches described above address important aspects of the problem, but they still diverge on the notion of tag relevance and evaluation protocol used. In (Xu et al., 2006b; Basile et al., 2007), e. g., no quantitative evaluation is presented, while in (Mishne, 2006), the notion of tag relevance is not entirely defined by the users but partially by experts. Furthermore, most of them make use of some content information which is specific to the particular type of resource of the system. It is certainly interesting to exploit content information, but since folksonomies can support different types of resources, e. g., audio, image, text, or video, one would need to write specific recommenders suited for each distinct content type. In this chapter we are particularly interested in generic algorithms that can be applied to folksonomies disregarding the domain and kind of resource supported.

Heymann et al. (2008b) model the tag prediction task as a binary classification problem for each tag with the web pages being the objects to classify. Besides the content of web pages, they also incorporate the anchor texts of links pointing to the page and host names of in-/outlinks as features for a support vector machine (SVM). They try to answer questions like “What precision can we get with low recall?”, “Which page information is best for predicting tags?”, or “What makes a tag predictable?”. Additionally, they apply association rules between tags to expand tag-based queries. Another analysis of the application of classification methods to the tag recommendation problem can be found in (Illig et al., 2009).

One task of the ECML PKDD Discovery Challenge 2008 (Hotho et al., 2008) also addressed the problem of tag recommendations in folksonomies. M. Tatu and D’Silva (2008) base their suggestions on normalized tags from posts and normalized concepts from textual content of resources. This includes user added text like title or description as well as the document content. Using natural language processing (NLP) tools they extract important concepts from the textual metadata and normalize them using Wordnet (Fellbaum, 1998). Lipczak (2008) developed a three step approach which utilizes

words from the title expanded by a folksonomy driven lexicon, personalized by the tags of the posting user. I. Katakis and Vlahavas (2008) consider the recommendation task as a multilabel text classification problem with tags as categories.

5.3 Algorithms

In this section we present three classes of recommendation algorithms we evaluate in the following sections: a straight-forward adaptation of Collaborative Filtering (Breese et al., 1998; Resnick et al., 1994) based on user-tag and user-resource projections, two adaptations of PageRank (Page et al., 1999) for folksonomies, and various methods based on counting the most popular tags.

5.3.1 Collaborative Filtering

Due to its simplicity and promising results, Collaborative Filtering (CF) has been one of the most dominant methods used in recommender systems. In the next section we recall the basic principles and then present the details of the adaptation to folksonomies.

Basic Collaborative Filtering Principle

The main idea of Collaborative Filtering is to suggest new objects or to predict the utility of a certain object based on the opinion of like-minded users (Sarwar et al., 2001). In CF, for m users and n objects, the user profiles are represented in a user-object matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$. The matrix can be decomposed into row vectors:

$$\mathbf{X} := [\vec{x}_1, \dots, \vec{x}_m]^T \text{ with } \vec{x}_u := [x_{u,1}, \dots, x_{u,n}], \text{ for } u := 1, \dots, m,$$

where $x_{u,o}$ indicates that user u rated object o by $x_{u,o} \in \mathbb{R}$. Each row vector \vec{x}_u corresponds thus to a user profile representing the object ratings of a particular user. This decomposition leads to user-based CF – in contrast to item-based algorithms (see (Deshpande and Karypis, 2004)).⁴

Now, one can compute, for a given user u , the recommendation as follows. First, based on the matrix \mathbf{X} and for a given k , the set N_u^k of the k users that are most similar to user $u \in U$ are computed:

$$N_u^k := \underset{v \in U \setminus \{u\}}{\operatorname{argmax}}^k \operatorname{sim}(\vec{x}_u, \vec{x}_v)$$

⁴We also measured the performance of item-based CF. Since precision and recall of its recommendations were for all datasets worse than those of user-based CF, we decided to present the later type of CF as the baseline for CF algorithms.

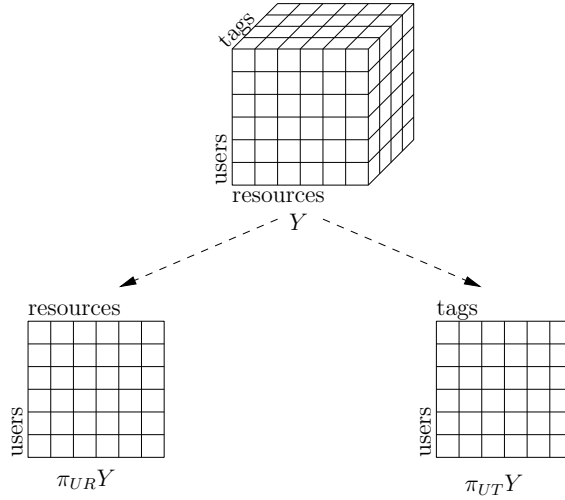


Figure 5.1: Projections of Y into the user's resource and user's tag spaces.

where the superscript in the argmax function indicates the number k of neighbors to be returned, and sim is regarded (in our setting) as the cosine similarity measure, i. e., $\text{sim}(\vec{x}_u, \vec{x}_v) := \frac{\langle \vec{x}_u, \vec{x}_v \rangle}{\|\vec{x}_u\| \|\vec{x}_v\|}$.

Then, for a given $n \in \mathbb{N}$, the top n recommendations consist of a list of objects ranked by decreasing frequency of occurrence in the ratings of the neighbors (see Eq. 5.1 below for the folksonomy case).

Collaborative Filtering for Recommending Tags in Folksonomies

Because of the ternary relational nature of folksonomies, traditional Collaborative Filtering cannot be applied directly, unless we reduce the ternary relation Y to a lower dimensional space (Balby Marinho and Schmidt-Thieme, 2007). To this end we consider as 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 (cf. Figure 5.1).

The projections preserve the user information, and lead to recommender systems based on occurrence or non-occurrence of resources or tags, resp., with the users. This approach is similar to recommenders that are based on web log data. Notice that here we have two possible setups in which the k -neighborhood N_u^k of a user u can be formed, by considering either the resources or the tags as objects.

Having defined matrix \mathbf{X} , and having decided whether to use $\pi_{UR}Y$ or $\pi_{UT}Y$ 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:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T} \sum_{v \in N_u^k} \operatorname{sim}(\vec{x}_u, \vec{x}_v) \delta(v, t, r) \quad (5.1)$$

where $\delta(v, t, r) := 1$ if $(v, t, r) \in Y$ and 0 else.

5.3.2 A Graph-Based Approach

The web search algorithm PageRank (Page et al., 1999) reflects the idea that a web page is important if there are many pages linking to it, and if those pages are important themselves.⁵ In (Hotho et al., 2006b), 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 we use it for generating tag recommendations.

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 *undirected* 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., $V = U \cup T \cup R$). 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. This results in the following definition.

⁵This idea was extended in a similar fashion to bipartite subgraphs of the web in HITS (Kleinberg, 1999) and to n -ary directed graphs in (Xi et al., 2004).

The rank of the vertices of the graph is computed with the weight spreading computation

$$\vec{w}_{t+1} \leftarrow dA^T\vec{w}_t + (1-d)\vec{p}, \quad (5.2)$$

where \vec{w} is a weight vector with one entry for each node in V , A is the row-stochastic version of the adjacency matrix⁶ of the graph $G_{\mathbb{F}}$ defined above, \vec{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 \vec{p} . By normalization of the vector \vec{p} , we enforce the equality $\|\vec{w}\|_1 = \|\vec{p}\|_1$. This⁷ ensures that the weight in the system will remain constant. The rank of each node is its value in the limit $\vec{w} := \lim_{t \rightarrow \infty} \vec{w}_t$ of the iteration process.

For a global ranking, one will choose $\vec{p} = \mathbf{1}$, i.e., the vector composed by 1's. In order to generate recommendations, however, \vec{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.

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 the experiments presented below, we will see that this version performs reasonable, but not exceptional. This is in line with our observation (Hotho et al., 2006b) that the topic-specific rankings are biased by the global graph structure. As a consequence, we developed in (Hotho et al., 2006b) 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. In our case, the topic is determined by the user/resource pair (u, r) for which we intend to compute the tag recommendation.

1. Let $\vec{w}^{(0)}$ be the fixed point from Equation (5.2) with $\vec{p} = \mathbf{1}$.
2. Let $\vec{w}^{(1)}$ be the fixed point from Equation (5.2) with $\vec{p} = \mathbf{1}$, but $\vec{p}[u] = 1 + |U|$ and $\vec{p}[r] = 1 + |R|$.
3. The final weight vector is

$$\vec{w} := \vec{w}^{(1)} - \vec{w}^{(0)}. \quad (5.3)$$

⁶ $a_{ij} := \frac{1}{\text{deg}(i)}$ if $\{i, j\} \in E$ and 0 else

⁷ ... together with the condition that there are no rank sinks – which holds trivially in the undirected graph $G_{\mathbb{F}}$.

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 $\vec{w}[x]$ of an element x of the folksonomy the *FolkRank* of x .⁸

For generating a tag recommendation for a given user/resource pair (u, r) , we compute the ranking as described and then restrict the result set $\tilde{T}(u, r)$ to the top n tag nodes.

5.3.3 Most Popular Tags

In this section we introduce methods based on tag counts. In the sequel we will see that these methods are particularly cheap to compute and therefore might be good candidates for online computation of recommendations.

For convenience, we define, for a user $u \in U$, the set of all his tag assignments $Y_u := Y \cap (\{u\} \times T \times R)$. The sets Y_r (for any resource $r \in R$) and Y_t (for any tag $t \in T$) are defined accordingly. Similarly, we define, for $t \in T$ and $r \in R$, $Y_{t,u} := Y \cap (\{u\} \times \{t\} \times R)$; and define $Y_{t,r}$ accordingly. Finally, we define, for a user $u \in U$, the set of all his tags $T_u := \{t \in T \mid \exists r \in R: (u, t, r) \in Y\}$. The set T_r (for any resource $r \in R$) is defined accordingly.

Variants of ‘Most Popular Tags’

1. Recommending the *most popular tags* of the folksonomy is the most simplistic approach. It recommends, for any user $u \in U$ and any resource $r \in R$, the same set:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T}^n (|Y_t|).$$

This approach suffers only minimally from cold-start problems.

2. Tags that globally are most specific to the resource will be recommended when using the *most popular tags by resource*:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T}^n (|Y_{t,r}|) .$$

3. Since users might have specific interests for which they already tagged several resources, using the *most popular tags by user* is another option:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T}^n (|Y_{t,u}|) .$$

⁸In (Hotho et al., 2006b) we showed that \vec{w} provides indeed valuable results on a large-scale real-world dataset while $\vec{w}^{(1)}$ provides an unstructured mix of topic-relevant elements with elements having high edge degree. In (Hotho et al., 2006c), we applied this approach for detecting trends over time in folksonomies.

As we will later see (cf. Section 5.6), none of the aforementioned methods alone will in general provide the best recommendations. Nevertheless, the simplicity and cost efficiency of algorithms based on tag counts make them a favored approach for use in existing folksonomy systems. Therefore, we experimented with a *mix* of the recommendations generated by variants 2 and 3 which we call *most popular tags mix* in the following sections.

Mix of ‘Most Popular Tags’ Recommenders

The main idea of this approach is to recommend a mix of the most popular tags of the user with the most popular tags of the resource. The simplest way to mix the tags is to add their counts and then sort them by their count:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T}^n (|Y_{t,r}| + |Y_{t,u}|) .$$

This way of mixing will be called *most popular tags mix 1:1*, since we just add the counts as they are. For instance, if the resource has been tagged four times with *web* by other users and the user has used the tag *web* six times on other resources, the tag *web* would get a count of ten.

Although this method already yields good results (as we will show in Section 5.6), the influence of the user-based recommendation will be very small compared to the resource-based recommendation if many people have tagged this resource. Vice versa, if a user has tagged many resources, his most popular tags might have counts that are much higher than the counts provided by the resources. Hence, we introduced another mix variant, where the tag counts of the two participating sets are *normalized* and *weighted* before they are added. We define as normalization function, for each tag $t \in T_r$:

$$\operatorname{norm}_r(t) := \frac{|Y_{t,r}| - \min_{t' \in T} |Y_{t',r}|}{\max_{t' \in T} |Y_{t',r}| - \min_{t' \in T} |Y_{t',r}|} . \quad (5.4)$$

For $t \in T_u$, the normalization $\operatorname{norm}_u(t)$ is defined in an analogue fashion. After normalization the weights of all tags in T_r and T_u lie between zero and one – with the most popular tag(s) having weight 1 and the least important tag(s) having weight 0. A pre-defined factor $\rho \in [0, 1]$ allows us to balance the influence of the user and the resource:

$$\tilde{T}(u, r) := \operatorname{argmax}_{t \in T}^n (\rho \operatorname{norm}_r(t) + (1 - \rho) \operatorname{norm}_u(t)) .$$

We call this method the *most popular tags ρ -mix*.

Note that the *most popular tags 0-mix* is just the *most popular tags by user* strategy, since the normalization does not change the order of the tags. Similarly, the *most popular tags 1-mix* is just the *most popular tags by resource* strategy. However, due to

normalization, the *most popular tags 0.5-mix* is not identical to the *most popular tags mix 1:1*.

In Section 5.6 we analyze how well different values of ρ perform and find the best value for the examined datasets.

5.4 Computational Costs

In an online scenario, where tag recommendations should be given to the user while he tags a resource, one must consider the computational costs of the used algorithm. Hence, in this section we want to discuss briefly the costs of the algorithms proposed so far. Where possible, we assume that the methods use efficient index structures (e. g., to access a user's tag assignments) or sparse data structures are used (i. e., for the matrix A in FolkRank). We will see that the methods described in the preceding section are especially cheap to compute and therefore might be good candidates for real-time computation of recommendations, if they can provide useful recommendations. Here we want to estimate the complexity of recommending n tags for a given user-resource tuple (u, r) using the proposed solutions.

5.4.1 Collaborative Filtering

The computational complexity of the CF algorithm depends on three steps:

1. Computation of projections: In order to compose the projections, we need to determine only the resources' and tags' co-occurrences with the set of users $V \subseteq U$ that have tagged the active resource $r \in R$. For that, we need to do a linear scan in Y resulting in a complexity of $\mathcal{O}(|Y|)$. However, with appropriate index structures, which allow to access the tag assignments of u (or r) efficiently, this reduces to $\mathcal{O}(\log(|R|) + |Y_u||V| \log(|U|))$.
2. Neighborhood formation: In traditional user-based CF algorithms, the computation of the neighborhood N_u is usually linear on the number of users as one needs to compute the similarity of the active user with all the other users in the database. However, in CF-based tag recommendations we are only interested in the subset V of users that tagged the active resource. Thus, the upper bound on the complexity of this step would be $\mathcal{O}(|V|Z)$, as we need to compute $|V|$ similarities each requiring Z operations. In the worst case $|V| = |U|$ but this rarely occurs in practice. In addition, we need to sort the similarities to compute the N_u nearest users. Therefore the complexity of this step is $\mathcal{O}(|V|(Z + \log(|N_u|)))$.
3. Recommendations: In order to compute the top- n recommendations for a given (u, r) pair, we need to: (i) count the tag occurrences of nearest users N_u similarities

(see Eq. 5.1), and (ii) sort the tags based on their weight, which results in a complexity of $\mathcal{O}(|Y_u||N_u| \log(n))$.

Hence, the whole complexity given the three steps above is $\mathcal{O}(\log(|R|) + |Y_u||V| + |V|(Z + \log(|N_u|)) + |Y_u||N_u| \log(n))$ and can be simplified to $\mathcal{O}(|V|(2|Y_u| + \log(|V|) + |Y_u| \log(|n|)) \in \mathcal{O}(|V||Y_u|)$ since $|N_u| \leq |V|$ and $Z \leq |Y_u|$.

5.4.2 The Graph-Based Approach

One iteration of the adapted PageRank requires the computation of $dA^T \vec{w} + (d-1)\vec{p}$, with $A \in \mathbb{R}^{s \times s}$ where $s := |U| + |T| + |R|$. If t marks the number of iterations, the complexity would therefore be $(s^2 + s)t \in \mathcal{O}(s^2 t)$. However, since A is sparse, it is more efficient to go linearly over all tag assignments in Y to compute the product $A^T \vec{w}$. Together with the costs of adding the preference vector $\vec{p} \in \mathbb{R}^s$ this results in a complexity of $\mathcal{O}((|Y| + s)t)$. After rank computation we have to sort the weights of the tags to collect the top n tags, thus the final complexity of the adapted PageRank for top- n tag recommendation is $\mathcal{O}((|Y| + s)t + |T| \log(n))$.

For FolkRank, one has to compute the baseline $\vec{w}^{(0)}$ once (and update it on a regular basis) – hence, these costs do not really add up to the costs for computing one recommendation. However, the baseline $\vec{w}^{(0)}$ has to be subtracted from $\vec{w}^{(1)}$, which costs at most $|T|$ iterations (since we are only interested in the weights of the tags). Thus, the costs of FolkRank are $\mathcal{O}((|Y| + s)t + |T| \log(n) + |T|)$, which can be simplified to $\mathcal{O}((|Y| + s)t)$, since $|T|$ is small compared to $|Y|$.

5.4.3 Most Popular Tags

If we want to compute, for a given pair (u, r) , the most popular tags of the user u (or the resource r), we need to linearly scan Y to calculate the occurrence counts for u 's tags (or r 's tags) and afterwards sort the tags we gathered by their count. This would result in a complexity of $\mathcal{O}(|Y| + |T_u| \log(n))$ (or $\mathcal{O}(|Y| + |T_r| \log(n))$). Nevertheless (as for CF), with efficient index structures to access T_u (or T_r) this reduces to $\mathcal{O}(\log(|U|) + |Y_u| + |T_u| \log(n))$ (or $\mathcal{O}(\log(|R|) + |Y_r| + |T_r| \log(n))$).

For the *most popular tags mixes* we have to consider both of the costs and additionally add the costs to normalize the tags, which includes finding the tags with the highest and lowest counts. This results in a complexity of $\mathcal{O}(\log(|U|) + |Y_u| + \log(|R|) + |Y_r| + |T_u| + |T_r| + (|T_u| + |T_r|) \log(n))$. With $|T_u| \leq |Y_u|$ the costs are at most $\mathcal{O}(4|Y_u| + 2|Y_u| \log(n)) \in \mathcal{O}(|Y_u|)$.

5.4.4 Comparison

Since Y_u is only a small part of Y , *CF* and the most popular methods are much cheaper to compute than *FolkRank*, which in each iteration has to scan Y . Additionally, both

methods do not need any iteration. Comparing *CF* and the *most popular mixes* requires to estimate the size of the set V of users, which have tagged a particular resource. This certainly depends on the resource at hand, but on average the factor $|V|$ of the *CF* costs will be larger than the constant factors of $|Y_u|$ in the *most popular mix* costs. In general, both methods have similar costs with some advantage on the side of the mixes.

5.5 Quantitative Evaluation

In order to evaluate the quality of the recommendations of the different algorithms, we have run experiments on three real-world datasets. In this section we first describe the datasets we used, how we prepared the data, the methodology deployed to measure the performance, and which algorithms we used, together with their specific settings. The results will be discussed in Section 5.6.

5.5.1 Datasets

To evaluate the proposed recommendation techniques we have chosen datasets from three different collaborative tagging systems: *Delicious*, *BibSonomy*, and *Last.fm*. They have different sizes, different resources, and are probably used by different people. Therefore we assume that our observations will also be significant for other collaborative tagging systems. Table 5.1 gives an overview on the datasets. For all datasets we disregarded if the tags had lower or upper case, since this is the behaviour of most systems when querying them for posts tagged with a certain tag (although often they store the tags as entered by the user).

Delicious. One of the first and most popular social bookmarking systems is Delicious⁹ which exists since the end of 2003. It allows users to tag bookmarks (URLs) and had according to its blog around 1.5 Mio. users in February 2007. We used a dataset from Delicious we obtained from July 27 to 30, 2005 (Hotho et al., 2006b).

BibSonomy. As described in Chapter 3, BibSonomy allows users to manage and annotate bookmarks and publication references simultaneously. 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 bibliography¹⁰ 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.

⁹<http://www.delicious.com/>

¹⁰<http://www.informatik.uni-trier.de/~ley/db/>

Table 5.1: Characteristics of the used datasets.

dataset	$ U $	$ T $	$ R $	$ Y $	$ P $	date	k_{\max}
Delicious	75,245	456,697	3,158,435	17,780,260	7,698,653	2005-07-30	77
BibSonomy	1,037	28,648	86,563	341,183	96,972	2007-04-30	7
Last.fm	3,746	10,848	5,197	299,520	100,101	2006-07-01	20

Last.fm. Audioscrobbler¹¹ is a “database that tracks listening habits”. The user profiles are built through the use of the company’s flagship product, Last.fm,¹² a system that provides personalized music streams for its users and updates their profiles using the music they listen to. Audioscrobbler exposes large portions of its data through its web services API. The data was gathered during July 2006, partly through the web services API (collecting user nicknames), partly crawling the Last.fm site. Here the resources are artist names, whose spellings are already normalized by the system.

5.5.2 Core Computation

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 (see, e. g., Sarwar et al., 2001) and restrict the evaluation to the ‘dense’ part of the folksonomy. To this end, we adapt the notion of a p -core (Batagelj and Zaversnik, 2002) 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*. For the Delicious dataset we will later see (cf. Section 5.6.1) that using the core will (except for the adapted PageRank) not change the relative performance differences of the algorithms.

To construct the p -core, recall that a folksonomy (U, T, R, Y) can be formalized equivalently as undirected tri-partite hypergraph $G = (V, E)$ with $V = U \cup T \cup R$ and $E = \{(u, t, r) \mid (u, t, r) \in Y\}$. First we define, for a subset V' of V (with $V' = U' \cup T' \cup R'$ and $U' \subseteq U, T' \subseteq T, R' \subseteq R$), the function

$$P(v, V') = \begin{cases} \{(v, S, r) \mid r \in R', S = T_{V'}(v, r)\} & \text{if } v \in U' \\ \{(u, v, r) \mid u \in U', r \in R'\} & \text{if } v \in T' \\ \{(u, S, v) \mid u \in U', S = T_{V'}(u, v)\} & \text{if } v \in R' \end{cases} \quad (5.5)$$

which assigns to each $v \in V'$ the set of all posts in which v occurs. Here, $T_{V'}(u, r)$ is defined as in Section 2.3, but restricted to the subgraph (V', E') , with E' containing all edges from E whose nodes are contained in V' . Let $p(v, V') := |P(v, V')|$. The p -core at

¹¹<http://www.audioscrobbler.net/>

¹²<http://www.last.fm/>

Table 5.2: Characteristics of the p -cores at level k .

dataset	k	$ U $	$ T $	$ R $	$ Y $	$ P $
Delicious	10	37,399	22,170	74,874	7,487,319	3,055,436
BibSonomy	5	116	412	361	10,148	2,522
Last.fm	10	2,917	2,045	1,853	219,702	75,565

level $k \in \mathbb{N}$ is then the subgraph of (V, E) induced by V' , where V' is a maximal subset of V such that, for all $v \in V'$, $p(v, V') \geq k$ holds.

Since $p(v, V')$ is, for all v , a monotone function in V , the p -core at any level k is unique (Batagelj and Zaversnik, 2002), and we can use the algorithm presented in (Batagelj and Zaversnik, 2002) for its computation.

An overview on the p -cores we used for our datasets is given in Table 5.2. For BibSonomy, we used $k = 5$ instead of 10 because of its smaller size. The largest k for which a p -core exists is listed, for each dataset, in the last column of Table 5.1.

Although the p -core as defined above breaks the symmetry of the hypergraph structure (contrary to tags, for users and resources the p -degree is not the same as the natural degree in the graph) it is the natural definition for our recommender scenario. We have also performed the evaluation on the symmetric variant of p (with lines 1 and 3 in Equation 5.5 modified similar to line 2), with rather similar results.

5.5.3 Evaluation Measures

To evaluate the recommenders we used a variant of the leave-one-out hold-out estimation (Herlocker et al., 2004) which we call *LeavePostOut*. In all datasets, we picked randomly, for each user u , one resource r_u , which the user had posted before. The task of the recommender was then to predict the tags the user assigned to r_u , based on the folksonomy (U, T, R, Y') with $Y' := Y \setminus (\{u\} \times T(u, r_u) \times \{r_u\})$.

As performance measures we use precision and recall which are standard in such scenarios (Herlocker et al., 2004). For (U, T, R, Y') , u , and r_u as defined above, precision and recall of a recommendation $\tilde{T}(u, r_u)$ are defined as follows

$$\text{recall}(\tilde{T}(u, r_u)) = \frac{|T(u, r_u) \cap \tilde{T}(u, r_u)|}{|T(u, r_u)|} \quad (5.6)$$

$$\text{precision}(\tilde{T}(u, r_u)) = \frac{|T(u, r_u) \cap \tilde{T}(u, r_u)|}{|\tilde{T}(u, r_u)|} . \quad (5.7)$$

For each dataset, we averaged these values over all its users:

$$\text{recall} = \frac{1}{|U|} \sum_{u \in U} \text{recall}(\tilde{T}(u, r_u)) \quad (5.8)$$

$$\text{precision} = \frac{1}{|U|} \sum_{u \in U} \text{precision}(\tilde{T}(u, r_u)) . \quad (5.9)$$

This process was repeated ten times for each dataset, each time with a randomly chosen resource per user, to further minimize the variance. In the sequel, the listed recall and precision values are thus always the averages over all ten runs.

5.5.4 Settings of the Algorithms

It is important to notice that not all algorithms necessarily have maximal coverage, i. e., can always recommend n tags. Since *FolkRank* and *most popular tags* are the only algorithms with maximal coverage, the evaluation can be perturbed if the other algorithms cannot fill the list up to the given n . In this sense, whenever the recommendation list of an algorithm is not filled up to n , we complete the remaining entries with tags taken from the *most popular tags* that are not already in the list.

For each of the algorithms of our evaluation, we will now describe briefly the specific settings used to run it.

Collaborative Filtering UT. For this Collaborative Filtering variant the neighborhood is computed based on the user-tag matrix $\pi_{UT}Y$. The only parameter to be tuned in the CF based algorithms is the number k of nearest neighbors. For that, multiple runs were performed where k was successively incremented in steps of 10 until a point where no more improvements in the results were observed. The best values for k were 80 for Delicious, 20 for BibSonomy and 60 for the Last.fm dataset.

Collaborative Filtering UR. Here the neighborhood is computed based on the user-resource matrix $\pi_{UR}Y$. For this approach the best values for k were 100 for Delicious, 30 for BibSonomy and 100 for the Last.fm dataset.

Adapted PageRank. With the parameter $d = 0.7$ we stopped computation after 10 iterations. In \vec{p} , we gave higher weights to the user u and the resource r_u at hand: While each user, tag and resource got a preference weight of 1, u and r_u got a preference weight of $1 + |U|$ and $1 + |R|$, resp.

FolkRank. The same parameters and preference weights were used as in the adapted PageRank.

Most Popular Tags / Most Popular Tags by Resource / Most Popular Tags by User. These three approaches have no parameters. They were applied as described in Section 5.3.3.

Most Popular Tags ρ -Mix. Initially, we computed tag recommendations for all $\rho \in \{0, 0.1, \dots, 0.9, 1\}$. In Section 5.6.1 we show that $\rho = 0.6$ is the most suitable of these values (at least on Delicious and BibSonomy), so that the comparison with the other algorithms will be done with this setting only.

5.6 Results

In this section we present and describe the results of the evaluation. We will see that all three datasets show the same overall behavior: *most popular tags* is outperformed by all other approaches; the *CF-UT* algorithm performs slightly better than and the *CF-UR* approach approximately as good as the *most popular tags by resource*, and *FolkRank* uniformly provides significantly better results. The results for *most popular tags by user* and the *most popular tags 0.6-mix* are different among the datasets, however. We will further elaborate on this later.

There are two types of diagrams. The first type of diagram (e. g., Figure 5.3) shows in a straightforward manner how the recall depends on the number of recommended tags. The other diagrams are usual precision-recall plots. Here 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). Hence, the steady decrease of all curves in those plots means that the more tags of the recommendation are regarded, the better the recall and the worse the precision will be.

Since we averaged for each dataset over ten runs, we added error bars showing the standard deviation to the plots. However, except for the BibSonomy dataset, the standard deviation is so small that the error bars are mostly hidden by the datapoint symbols.

5.6.1 Delicious

Due to the fact that the dataset from Delicious is by far the largest of the three we considered, we will discuss the results in more detail.

Determining ρ for the Most Popular ρ -Mix

Before comparing the different algorithms described in the previous sections, we focus on finding an appropriate ρ for the *most popular ρ -mix* recommender on the Delicious *p*-core at level 10. Therefore, we varied ρ in 0.1-steps from 0 to 1 and plotted the resulting precision and recall; for comparison purposes we also added the plot of the *most popular mix 1:1* recommender.

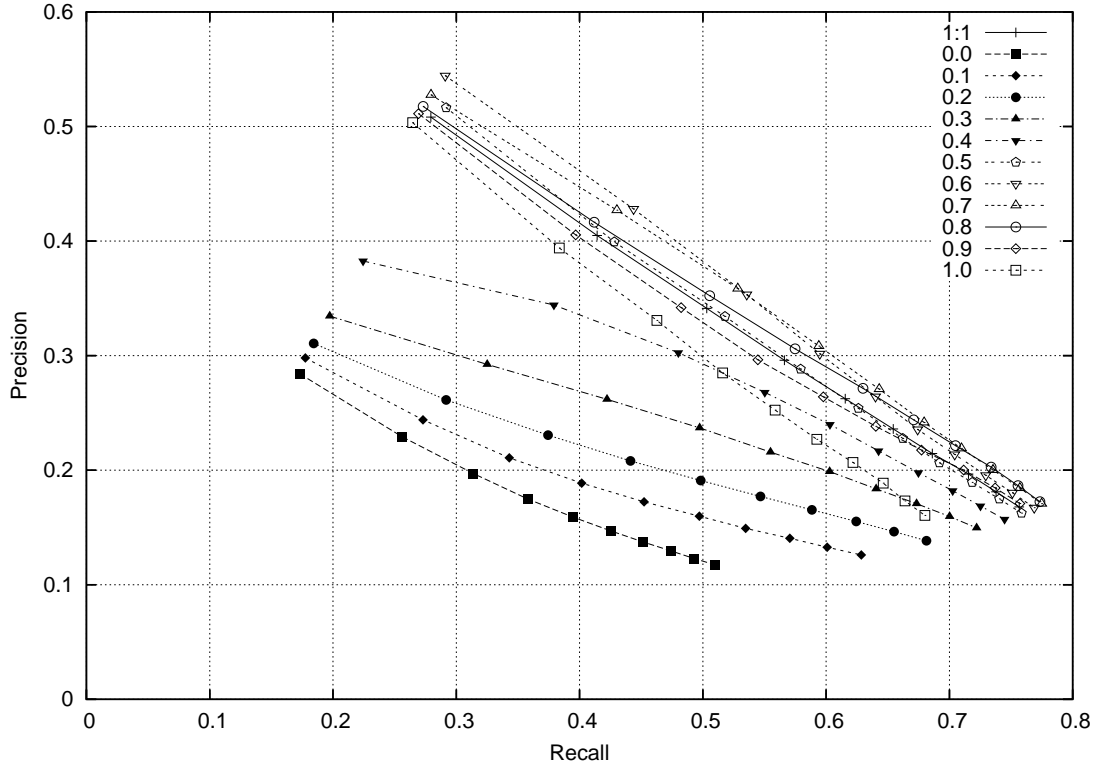
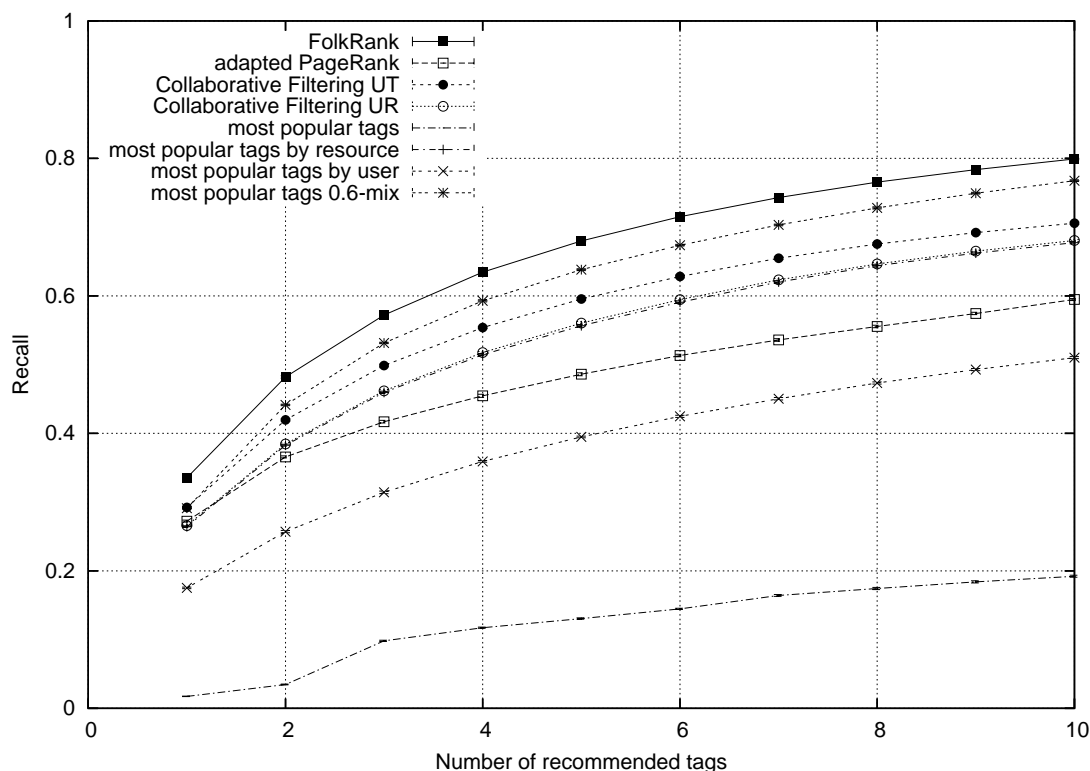


Figure 5.2: Recall and precision of the *most popular tags mix 1 : 1* and the *most popular tags ρ -mix* for $\rho \in \{0, 0.1, \dots, 0.9, 1\}$ on the Delicious p -core at level 10.

As can be seen in Figure 5.2, the *most popular tags by user* ($\rho = 0$) recommender performs worse than the *most popular tags by resource* ($\rho = 1$) recommender for all numbers of recommended tags. All mixed versions perform better than *most popular tags by user* and all mixed versions with $\rho \geq 0.5$ perform better than *most popular tags by resource*. The best performance is obtained for $\rho = 0.6$ for the top three recommendations and $\rho = 0.7$ for more than three recommendations. We conclude, that the tags which other users used for that resource are better recommendations than the most popular tags of the user. Nevertheless, adding a small amount of popular tags of the user to the tags from the resource increases both precision and recall.

We observed a similar precision/recall behaviour for the different values of ρ on the original Delicious data as well as on the BibSonomy dataset. For the following evaluations we decided therefore to include the results of the *most popular tags 0.6-mix* recommendations only, since for the top recommendations they have the best recall and precision and for more tags are still very close to the best results.

Figure 5.3: Recall for Delicious p -core at level 10.

Comparison of the Algorithms on the p -core at Level 10

Figure 5.3 shows how the recall increases, when more tags of the recommendation are used. All algorithms perform significantly better than the baseline *most popular tags* and the *most popular tags by user* strategy – whereas it is much harder to beat the *most popular tags by resource*. The most apparent result is that the graph-based *FolkRank* recommendations have superior recall – independent of the number of regarded tags. The top 10 tags given by *FolkRank* contained on average 80% of the tags the users decided to attach to the selected resource. The second best results come from the *most popular tags 0.6-mix*, followed by the *Collaborative Filtering* approach based on user’s tag similarities.

The idea to suggest the *most popular tags by resource* results in a recall which is very similar to using the *CF* recommender based on user’s resource similarities – both perform worse than the aforementioned approaches. Between *most popular tags by resource* and *most popular tags* are the *adapted PageRank* which is biased towards the high degree nodes, as discussed in Section 5.3.2, and the *most popular tags by user* recommendations,

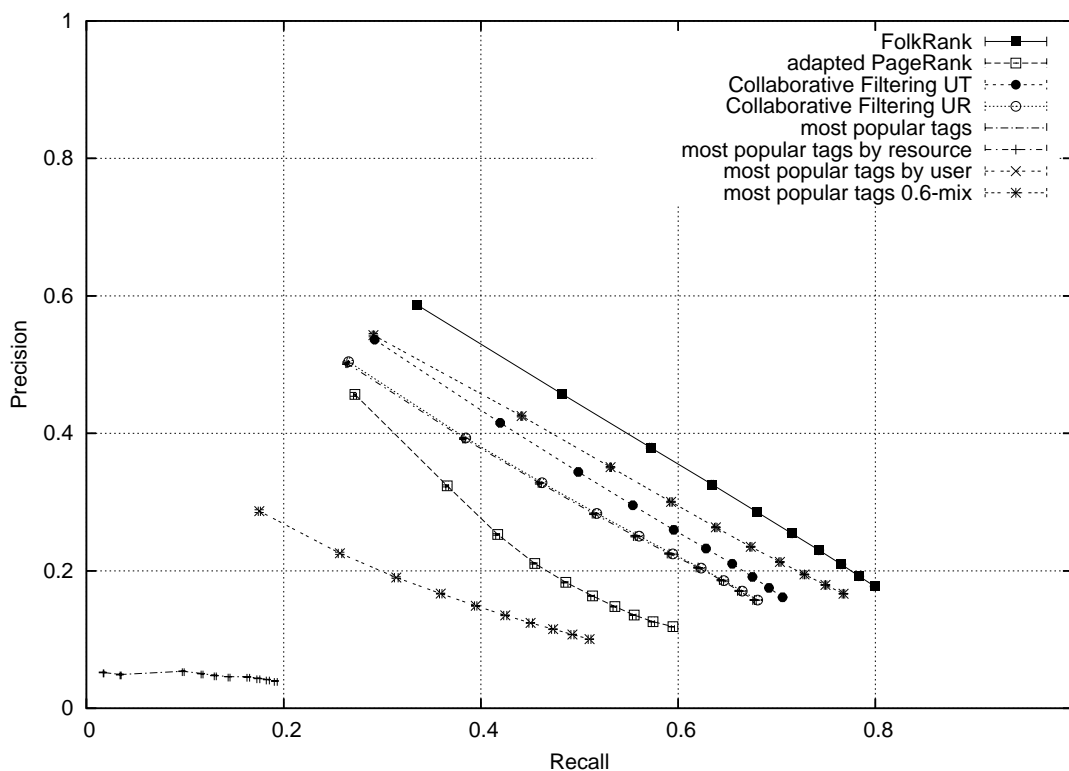


Figure 5.4: Recall and precision for Delicious p -core at level 10.

which again perform not so well.

The precision-recall plot in Figure 5.4 extends Figure 5.3 with the precision measure. It again reveals clearly the quality of the recommendations given by *FolkRank* compared to the other approaches. Its precision values are systematically above those of the other approaches. For its top recommendations, *FolkRank* reaches precisions of 58.7%.

A post in Delicious contains only 2.45 tags on average. A precision of 100% can therefore not be reached when recommending ten tags. This justifies the poor precision of less than 20% for all approaches when recommending ten tags. However, from a subjective point of view, the additional ‘wrong’ tags may even be considered as highly relevant, as the following example shows, where the user *tnash* has tagged the page <http://www.ariadne.ac.uk/issue43/chudnov/> with the tags *semantic*, *web*, and *webdesign*. Since that page discusses the interaction of publication reference management systems in the web by the OpenURL standard, the tags recommended by *FolkRank* (*openurl*, *web*, *webdesign*, *libraries*, *search*, *semantic*, *metadata*, *social-software*, *sfx*, *seo*) are adequate and capture not only the user’s point of view that this is a webdesign re-

lated issue in the semantic web, but also provide him with more specific tags like *libraries* or *metadata*. The *CF* based on user's tag similarities recommends very similar tags (*openurl*, *libraries*, *social-software*, *sfx*, *metadata*, *me/toread*, *software*, *myndsi*, *work*, *2read*). The additional tags may thus animate users to use more tags and/or tags from a different viewpoint for describing resources, and thus lead to converging vocabularies.

The essential point in this example is, however, that *FolkRank* is able to predict – additionally to globally relevant tags – the exact tags of the user which *CF* could not. This is due to the fact that *FolkRank* considers, via the hypergraph structure, also the vocabulary of the user himself, which *CF* does not do. It was this observation that motivated the creation of the *most popular tags ρ -mix*-recommender, where we – in contrast to *CF* – include also the user's tags in the recommendations. As the diagrams show, we succeeded and could gain results better than those of *CF* and only slightly worse than those of *FolkRank*.

The standard deviation for the ten runs of all algorithms on this dataset is for both precision and recall below 3 %.

Comparison of the Algorithms on the Original Dataset

We conclude the evaluation on Delicious with results on the original Delicious dataset, i. e., the dataset as it was before applying the core computation as described in Section 5.5.2. Figure 5.5 shows the recall and precision of the algorithms for this dataset. Due to the long tail of users and resources which occur in only one post, we regarded only resources and users with at least two posts. Otherwise, most of the algorithms would not be able to produce recommendations. Apart from the *adapted PageRank*, the results are similar to the results on the the *p*-core at level 10, with an overall decrease of both precision and recall. The only algorithm which seems to profit from the remaining long tail is the *adapted PageRank*. This is likely due to the fact that the many tags in the long tail together are able to outbalance to a certain degree the strong influence of the nodes with high edge degree. Nevertheless, it can not reach the performance of *FolkRank* or the *most popular tags 0.6-mix*.

The standard deviation for the ten runs of all algorithms on this dataset is for both precision and recall below 2 %.

5.6.2 BibSonomy

For this dataset, the results have a much larger standard deviation, as can be seen by the error bars in Figure 5.6. This is due to the fact that every run is averaging over 116 users only (cf. Table 5.2) and thus the performance of the ten runs differs more. Nevertheless, the tendency of the performance of the different methods is similar to the performance on the other datasets. *FolkRank* provides on average best precision and recall, followed by the *most popular tags 0.6-mix* recommender. Both *Collaborative Filtering* algorithms

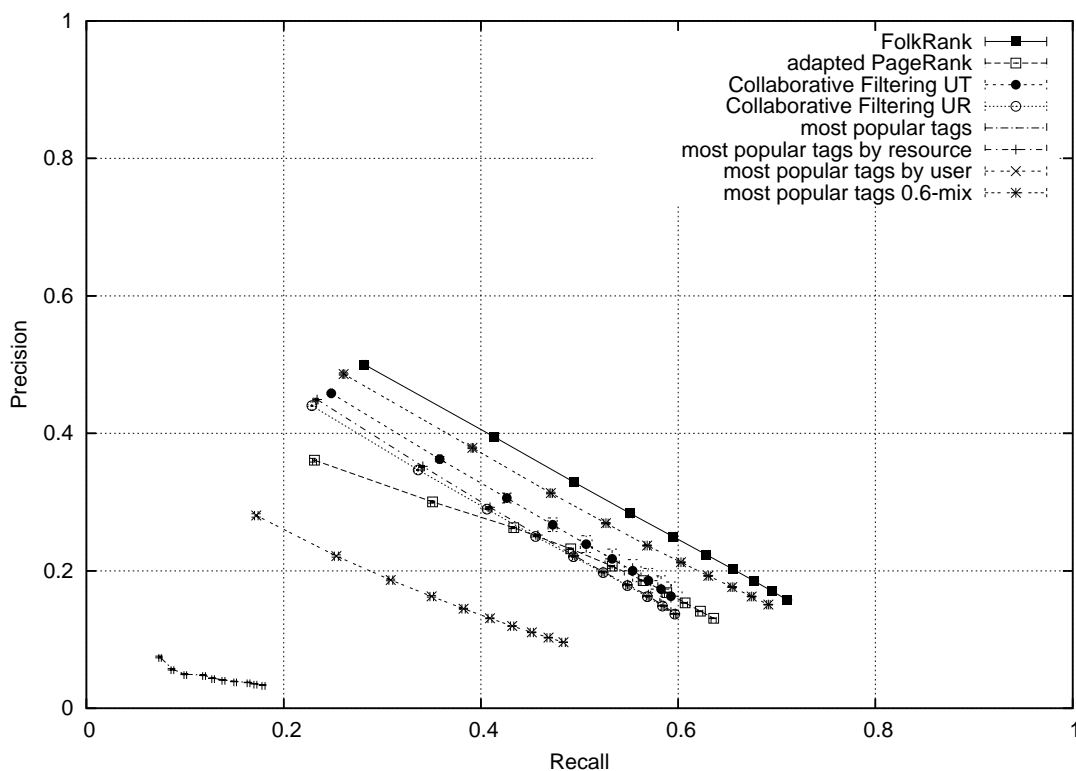


Figure 5.5: Recall and precision for Delicious.

and *most popular tags by resource* show similar results for higher numbers of tags.

5.6.3 Last.fm

On this dataset, FolkRank again outperforms the other approaches. Here, its recall is considerably higher than on the other datasets, see Figure 5.7. Even when just two tags are recommended, the recall is close to 60% and goes up to 92% for 10 tags. The standard deviation for the ten runs of all algorithms on this dataset is for both precision and recall below 7%.

The most surprising observation is, though, that here *most popular tags by user* is considerably better than *most popular tags by resource* and even *Collaborative Filtering*, such that it is the second best algorithm after *FolkRank*. An explanation could be the average number of tags a user has in this dataset (cf. Table 5.3). Compared to the Delicious and BibSonomy datasets, here the average is much lower with around twelve tags. Additionally, the average number of tags per resource in the Last.fm dataset is much higher than in the other two datasets and in particular higher than the average

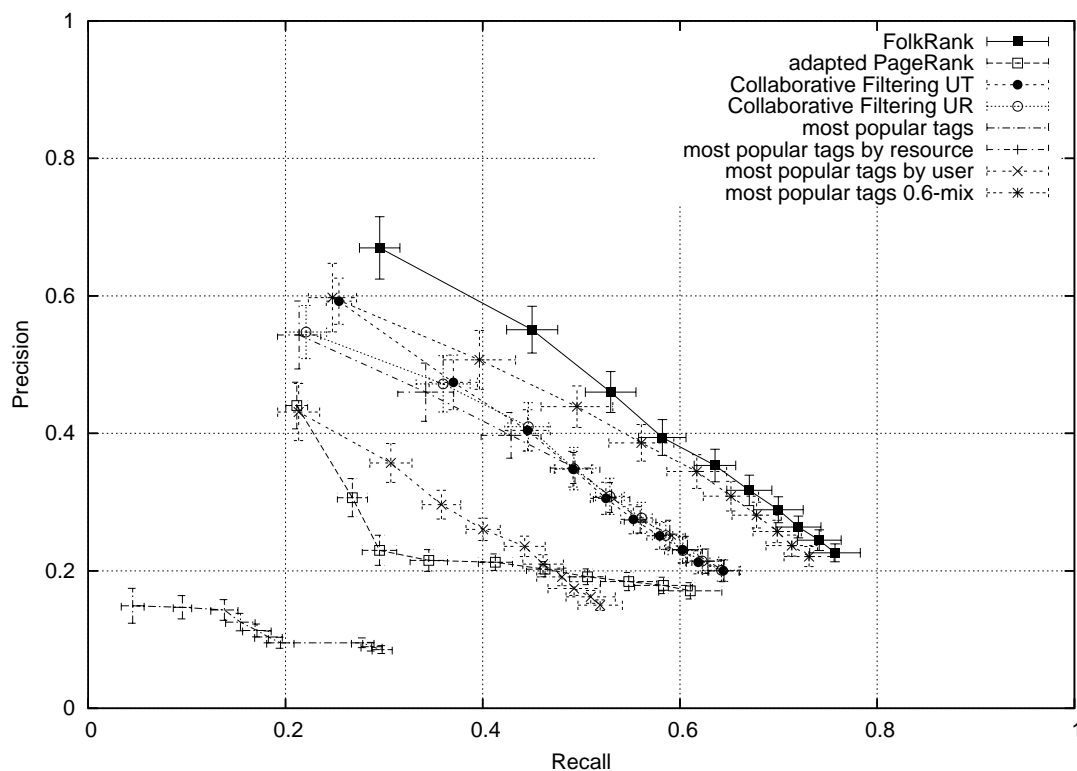
Figure 5.6: Recall and precision for BibSonomy p -core at level 5.

Table 5.3: Average number of tags per user and tags per resource.

dataset	$\frac{1}{ U } \sum_{u \in U} T_u $	$\frac{1}{ R } \sum_{r \in R} T_r $
Delicious p -core at level 10	59.18	25.87
BibSonomy p -core at level 5	31.85	14.14
Last.fm p -core at level 10	11.84	44.19

number of tags per user (in contrast to the other two datasets, where it is the other way around). Hence, if a user has on average only twelve tags, proposing tags he used earlier instead of tags other users attached to the resource provides a better chance to suggest the tags the user finally chose. Needless to say that it would be interesting to know, why the averages on the Last.fm dataset are so different from the other datasets. It could depend on the rather limited domain of the resources which can be tagged in Last.fm, but might also result from the crawling strategy which was deployed to gather this dataset.

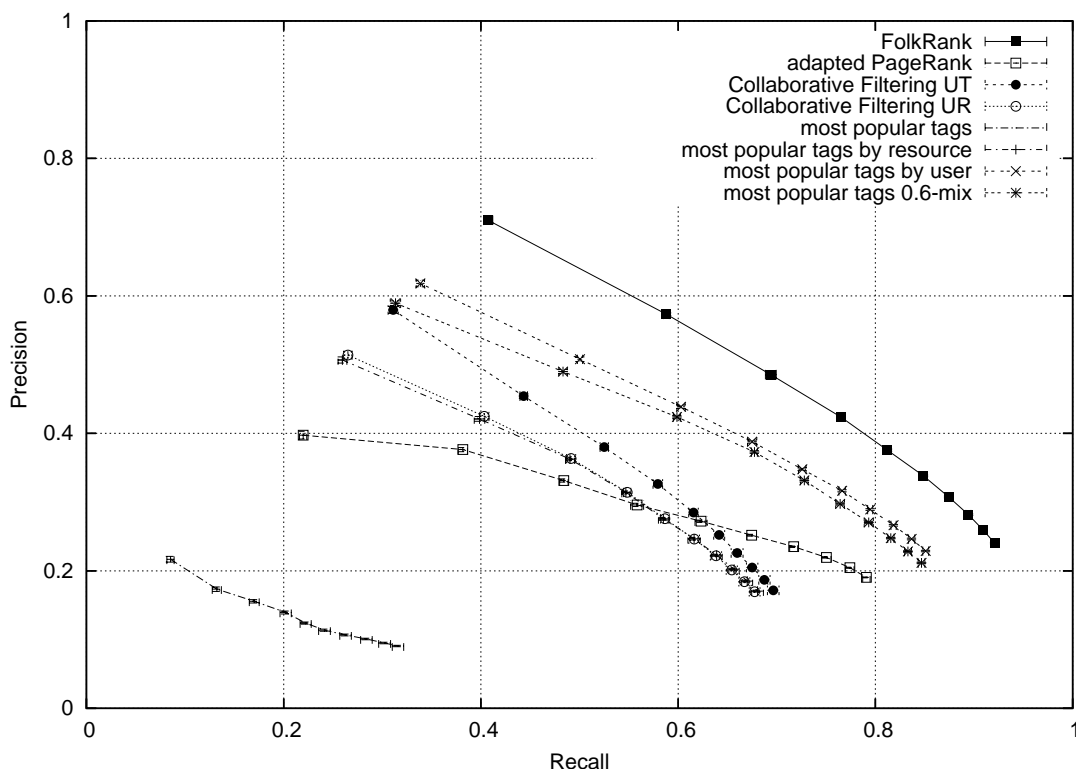


Figure 5.7: Recall and precision for Last.fm p -core at level 10.

Due to the different performance of the *most popular tags by user/resource* recommendations, the performance of the *most popular tags ρ -mix*, of course, differs significantly from the results on the other datasets. A comparison (not shown here) of different values for ρ showed, that the *most popular tags ρ -mix* on this dataset mostly performed worse than *most popular tags by user* (although always better than *most popular tags by resource*).

5.7 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 – as discussed in Section 5.4 – 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 proposed in this work 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. In the next chapter we will see how this methods perform in an online setting.

Finally, despite its simplicity and non-personalized aspect, the *most popular tags* achieved reasonable precision and recall on the small datasets (Last.fm and BibSonomy) which indicates its adequacy for the cold start problem.

Part III

Applications

In this part we present three applications related to folksonomies. First, we deploy tag recommenders in BibSonomy and evaluate different methods in an online setting. Second, we present a community support application which transfers knowledge discovery methods for folksonomies to the social semantic desktop. Finally, we apply the folksonomy paradigm to search engine click logs by introducing logsonomies and we provide first evidence that their structural properties are similar to those of folksonomies.

Chapter 6

A Tag Recommendation Framework for BibSonomy

To continue the research on tag recommendations described in the previous chapter, we implemented a tag recommendation framework for BibSonomy. It allows us to test, evaluate and compare different tag recommendation algorithms in an online setting, where the users of BibSonomy actually see the recommendations during the posting process. The chapter is based on work published in (Jäschke et al., 2009a,b).

6.1 Introduction

In this chapter we describe the tag recommendation framework we developed for BibSonomy and present first insights we gained from evaluating different recommender systems using the framework. The motivation to design such a framework is manifold:

- The arguments to provide the user with tag recommendations discussed in Section 5.1 support the need for good recommendations in BibSonomy. Therefore, we need a foundation to implement and run appropriate methods in the online system.
- We want to transfer the research results attained in Chapter 5 into practice.
- The experience we gained by organizing the ECML PKDD Discovery Challenge 2008 has shown that evaluation and comparison of different recommender systems in an offline setting (as done in Chapter 5) can suffer from artifacts present in the data like masses of imported or automatically annotated posts. Furthermore, a realistic setting should force the recommenders to adhere to timeouts and other constraints which are difficult to control in an offline setting. Therefore, we needed a framework which allowed us the evaluation of online tag recommendations as one task of the Discovery Challenge 2009 we also organized (more on this in Section 6.6).
- We want to offer the tag recommendation research community a realistic testbed for their methods.
- Existing frameworks (cf. Section 6.2) mostly do not fit the tag recommendation scenario we have to handle (e. g., they do not suggest re-occurring items).

Home ▼ myBibSonomy post bookmark post publication tags authors relations ▼ groups

Feel free to edit your bookmark

general information

url*

title*

description, comment

tags*

(space separated)

recommender recognition recht

recommendation **conference 2007 2009 rs systems**

tags of copied item **systems 2008 acm pc recommender conference**

Figure 6.1: BibSonomy’s recommendation interface on the bookmark posting page. The box labeled ‘tags’ contains a text input field where the user can enter the (space separated) tags, tags suggested for autocompletion, the tags from the recommender (bold), and the tags from the post the user just copies.

The framework is responsible for delivering tag recommendations to the user in two situations: when he edits a bookmark or publication post. Since the part of the user interface showing recommendations is very similar for both the bookmark posting and the publication posting page, we show in Figure 6.1 the relevant part of the ‘postBookmark’ page only.¹

Below the fields for entering URL, title, and a description (which are typically automatically filled), the box labeled ‘tags’ keeps together the tagging information. There, the user can manually enter the tags to describe the resource. During typing she is assisted by a JavaScript autocompletion which selects tags among the recommended tags and all of her previously used tags whose prefix matches the already entered letters. The suggested tags are shown directly below the tag input box (in the screenshot *recommender*, *recognition*, and *recht*). Further down there are in bold letters up to five recommended tags ordered by their score from left to right. Thus, the recommender in action regarded *conference* to be the most appropriate tag for this resource and user. To the very right of the recommendation is a small icon depicting the *reload* button. It allows the user to request a new tag recommendation if she is unsatisfied with the

¹Logged in users can access this page at <http://www.bibsonomy.org/postBookmark>.

one shown or wants to request further tags. We investigate the usage of this button in Section 6.5.2.

Besides triggering autocompletion with the tabulator key during typing, users can also click on tags with their mouse. They are then added to the input box. When the user copies a resource from another user’s post, the tags the other user used to annotate the resource are shown below the recommended tags (‘tags of copied item’). They are also regarded for autocompletion.

Aside from describing the framework we also try to answer such questions like “What is the performance of a recommender?”, “Are there users with a tendency to a certain recommender?”, or “Which click behaviour do users show?”. Therefore, we formalize the tag recommendation task as: Given a resource r and a user u who wants to annotate r , the recommender shall return a set of recommended tags $\tilde{T}(u, r) := \{t_1, \dots, t_k\}$ together with a *scoring function* $f: \tilde{T}(u, r) \rightarrow [0, 1]$ which assigns to each tag a score.² The value of k is fixed to 5 throughout our analysis, although a recommender is allowed to return less than k tags.

This chapter is structured as follows: In Section 6.2 we review related work in the field and explain in Section 6.3 the details of our tag recommendation framework. Then we elaborate on the evaluation methods (cf. Section 6.4) we have used to gather the results presented in Sections 6.5 and 6.6. We close with a conclusion and ideas for future work.

6.2 Related Work

Although having a different recommendation target (resources rather than tags), the REFEREE framework described by Cosley et al. (2002) is most closely related to our work. It provided recommendations for the CiteSeer (formerly ResearchIndex) digital library. REFEREE recommends scientific articles to users of ResearchIndex while they search and browse. An open architecture allows researchers to integrate their methods into REFEREE. Besides the different recommendation target, the focus of the paper is more on the evaluation of several different strategies than on the details of the framework.

A powerful, open, and well documented framework for recommendations is the Duine Framework³ developed by Novay. It is based on work by van Setten (2005) and has a focus on explicit user ratings and non re-occurring items, e. g., like in a movie recommendation scenario where one does not recommend movies the user has already seen. This is in contrast to tag recommendations, where re-occurring tags are a crucial requirement of the system. Similar to what we present in Section 6.3.2, the framework implements various hybrid recommenders. They have been studied extensively by the research community – for a survey see (Burke, 2002).

²Although, of course, f also depends on u and r , we will omit those two variables to simplify notation.

Since f always appears together with $T(u, r)$, it should be clear from context, which f is meant.

³<http://duineframework.org/>

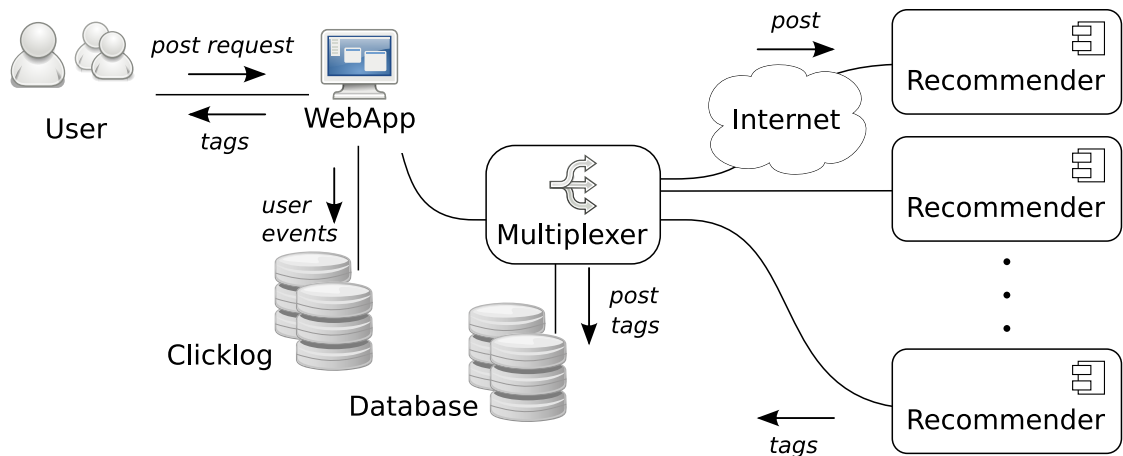


Figure 6.2: The involvement of BibSonomy’s components in a schematic recommendation process.

Another recommendation framework is the AURA project’s ‘TasteKeeper’ (Green and Alexander, 2008) from Sun Microsystems. Despite not having been described in the literature, it has a strong focus on collaborative filtering algorithms.

6.3 The Framework

Implementing a tag recommendation framework requires to tackle several challenges. For example, having enough data available for recommendation algorithms to produce helpful recommendations is an important requirement. The recommender needs access to the systems database and to what the user is currently posting (which could be accomplished, e.g., by (re-)loading recommendations using techniques like AJAX). Further data – like the full text of documents – could be supplied to tackle the cold-start problem (e.g., for content-based recommenders). Further aspects which should be taken into account include implementation of logging of user events (e.g., clicking, key presses, etc.) to allow for efficient evaluation of the used recommendation methods in an online setting. Together with an online evaluation, this also allows us to tune the result selection strategies to dynamically choose the (currently) best recommendation algorithm for the user or resource at hand. The multiplexing of several available algorithms together with the simple inclusion of external recommendation services (by providing an open recommendation interface) is one of the benefits of the proposed framework.

Figure 6.2 gives an overview on the components of BibSonomy involved in a recommendation process. The web application receives the user’s HTTP request and queries the multiplexer (cf. Section 6.3.4) for a recommendation – which provides post infor-

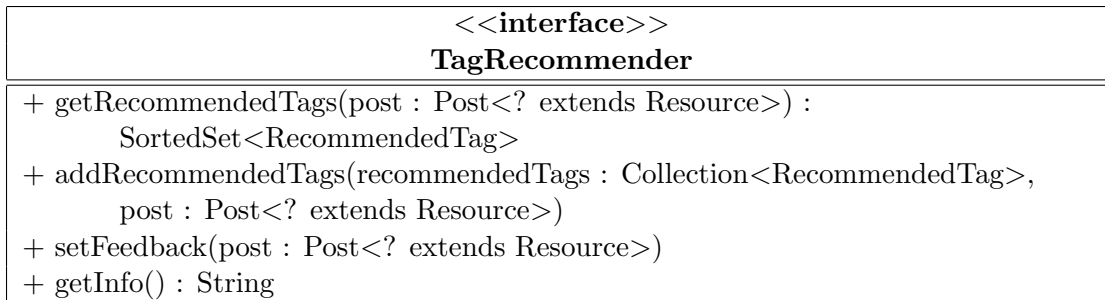


Figure 6.3: The UML class diagram of the tag recommender interface.

mation like URL, title, user name, etc. Besides, click events are logged in a database (see Section 6.4.3). The multiplexer then requests the active recommenders to produce recommendations and selects one of the results. The suggested tags and the post are then logged in a database and the selected recommendation is returned to the user.

6.3.1 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 6.3 shows the UML class diagram of the *TagRecommender* interface one must implement to deliver recommendations to BibSonomy. We decided to keep the interface as simple as possible by requiring only four methods, building on BibSonomy's existing data model (Post, Tag, etc.) 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 which could be handed to *addRecommendedTags*, 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 (including the tags) as the user stored it

in the database is given to the recommender such that it can measure and potentially improve its performance. Additionally, the *postID* introduced in Section 6.4.3 is contained in the post (as well as in the posts given to the first two methods) such that the recommender can connect the post with the recommended tags it provided.

Finally, the *getInfo* method allows the programmer to provide some information describing the recommender. This can be used to better identify recommenders or can be shown to the user.

Two further classes augment the interface: The *RecommendedTag* class basically extends the *Tag* class of the BibSonomy model (cf. Section 3.4.1) 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.

Our implementation is based on Java. All described classes are contained in the module *bibsonomy-model*, which is available online as a Java archive in a Maven2 repository.⁴ However, implementations are not restricted to Java – using the remote recommender (see Section 6.3.3) one can implement a recommender in any language which is then integrated using XML over HTTP requests.

6.3.2 Meta Recommender

Meta or *hybrid recommenders* (Burke, 2002) do not generate recommendations on their own but instead call other recommenders and modify or merge their results. Since they also implement the *TagRecommender* interface, they can be used like any other recommender. More formally, given n recommendations $\tilde{T}_1(u, r), \dots, \tilde{T}_n(u, r)$ and corresponding scoring functions f_1, \dots, f_n , a meta recommender produces a merged recommendation $\tilde{T}(u, r)$ with scoring function f . The underlying design pattern known from software architecture is that of a *Composite* (Gamma et al., 1995).

As we will see in Section 6.3.5, meta recommenders allow the building of complex recommenders from simpler ones and thus simplify implementation and testing of algorithms and even stimulate development of new methods. Furthermore, they allow for flexible configuration, since their underlying recommenders can be exchanged at runtime. This section introduces the meta recommenders that are currently used in the framework.

First Weighted By Second

As an example of a cascade hybrid, the idea behind this recommender is to re-order the tags of one recommendation using scores from another recommendation. More precisely, given recommendations $\tilde{T}_1(u, r)$ and $\tilde{T}_2(u, r)$ and corresponding scoring functions f_1 and

⁴<http://dev.bibsonomy.org/maven2/org/bibsonomy/bibsonomy-model/>

f_2 , this recommender returns a recommendation $\tilde{T}(u, r)$ with scoring function f , which contains all tags from $\tilde{T}_1(u, r)$ which appear in $\tilde{T}_2(u, r)$ (with $f(t) := f_2(t)$) plus all the remaining tags from $\tilde{T}_1(u, r)$ (with lower f but respecting the order induced by f_1). If $\tilde{T}_1(u, r)$ does not contain enough recommendations, $\tilde{T}(u, r)$ is filled by the not yet used tags from $\tilde{T}_2(u, r)$ – again with f being lower than for the already contained tags and respecting the order induced by f_2 .

Weighted Merging

This weighted hybrid recommender enables merging of recommendations from different sources and weighting of their scores. Given n recommendations $\tilde{T}_1(u, r), \dots, \tilde{T}_n(u, r)$, corresponding scoring functions f_1, \dots, f_n , and (typically fixed) weights ρ_1, \dots, ρ_n (with $\sum_{i=1}^n \rho_i = 1$), the weighted merging recommender returns a recommendation $\tilde{T}(u, r) := \bigcup_{i=1}^n \tilde{T}_i(u, r)$ and a scoring function $f(t) := \sum_{i=1}^n \rho_i f_i(t)$ (with $f_i(t) := 0$ for $t \notin \tilde{T}_i(u, r)$).

6.3.3 Remote Recommender

The remote recommender retrieves recommendations from an arbitrary external service using HTTP requests in REST-based (Fielding, 2000) interaction. Therefore, it uses the XML schema of the BibSonomy REST API (cf. Section 3.5.4). This recommender has three advantages: it allows us to distribute the recommendation work over several machines, it opens the framework to include recommenders from auxiliary partners, and it enables programming language independent interaction with the framework.

To simplify implementation and integration of external recommenders, we provide an example web application needing almost no configuration to include a custom Java recommender.⁵ Furthermore, we plan to integrate recommendations into the REST API to allow clients to retrieve recommendations, e. g., such that the Firefox browser add-on can show recommendations during bookmark posting.

6.3.4 Multiplexing Tag Recommender

Our framework's technical core component is the so called *multiplexing tag recommender* (see Figure 6.2). Implementing BibSonomy's tag recommender interface, it provides the web application with tag recommendations by querying one of the configured recommenders. Furthermore, the multiplexer logs all recommendation requests and each recommender's corresponding result in a database (see Section 6.4.3). For this purpose, every tag recommender is registered during startup and assigned to a unique identifier.

Whenever the *getRecommendedTags* method of the multiplexer is invoked, the corresponding recommendation request is delegated to each available recommender, spawning a separate thread for each recommender. After a timeout period of 100 ms, one of the

⁵<http://dev.bibsonomy.org/maven2/org/bibsonomy/bibsonomy-recommender-servlet>

collected recommendations is selected, applying a preconfigured *selection strategy*. For our evaluation procedure we implemented a ‘*sampling without replacement*’ strategy which randomly chooses exactly one recommender and returns all of its recommended tags. If the user requests recommendations more than once during the same posting process (e. g., by using the ‘reload’ button), the strategy selects recommendations from a recommender the user has not yet seen during this process.

6.3.5 Example Recommender Implementations

Using the proposed framework, we implemented several recommendation methods. Two of them were active in BibSonomy during the evaluation period in Section 6.5. Both build upon the meta recommenders described in Section 6.3.2 and simpler recommenders which we describe only briefly because they are fairly self-explanatory. The short names in parentheses are for later reference.

Most Popular ρ -Mix (MP ρ -mix)

Motivated by the good results of mixing tags which often have been attached to the resource with tags the user has often used, we implemented a variant of the *most popular ρ -mix* recommender described in Section 5.3.3. Another factor was its efficient computability which can be supported by appropriate tables and indexes in the database. Again, we set the parameter ρ of this recommender to $\rho = 0.6$ for evaluation. The recommender has been implemented as a combination of three recommenders:

1. the *most popular tags by resource* recommender which returns the k tags $\tilde{T}_1(u, r)$ which have been attached to the resource most often (with $f_1(t) := \frac{|Y \cap U \times \{t\} \times \{r\}|}{|Y \cap U \times T \times \{r\}|}$, i. e., the relative tag frequency),
2. the *most popular tags by user* recommender which returns the k tags $\tilde{T}_2(u, r)$ the user has used most often (with $f_2(t) := \frac{|Y \cap \{u\} \times \{t\} \times R|}{|Y \cap \{u\} \times T \times R|}$, i. e., the relative tag frequency), and
3. the *weighted merging* meta recommender described in Section 6.3.2 which merges the tags of the two former recommenders, with weights $\rho_1 = \rho = 0.6$ and $\rho_2 = 1 - \rho = 0.4$.

Currently, the existing indexes on the tag assignment table are sufficient to quickly retrieve the most popular tags of users and resources. Nevertheless, we intend to create separate tables for counting user-tag and resource-tag co-occurrences, since they could also speed up retrieval of the corresponding tag clouds.

Title Tags Weighted by User Tags (TbyU)

Inspired by the first recommender implemented in BibSonomy (Illig, 2006) and by similar ideas in (Lipczak, 2008), we implemented a recommender which scores tags extracted from the resource’s title using the frequency of the tags used by the user. Technically, this is again a combination of three recommenders:

1. a simple *content based recommender*, which extracts k tags $\tilde{T}_1(u, r)$ from the title of a resource, cleans them and checks against a multilingual stopword list,
2. the *most popular tags by user* recommender as described in the previous section – here returning *all* tags $\tilde{T}_2(u, r)$ the user has used (by setting $k = \infty$), and
3. the *first weighted by second* meta recommender described in Section 6.3.2 which weights the tags from the content based recommender by the frequency of their usage by the user as given by the second recommender.

Other

Besides the simple recommenders introduced along the $MP\rho$ -mix and TbyU recommender, we have implemented recommenders for testing purposes (a *fixed tags recommender* and a *random tags recommender*), a recommender which proposes tags from a web page’s HTML meta information keywords, as well as a recommender using the FolkRank algorithm.

More complex recommenders can be thought of, e. g., a nested *first weighted by second* recommender, whose first recommender is a *weighted merging* meta recommender merging the suggestions from a *content based recommender* and a *most popular tags by resource* recommender and then scoring the tags by the scores from the *most popular tags by user* recommender.

6.4 Evaluation

One important incentive to develop the framework is to provide a testbed for evaluation of different recommender systems. Therefore, we show in the following two sections which kind of evaluation the framework allows and how recommenders perform in practice. In doing so, we use the terminology introduced in Section 5.5.3 and evaluate the performance of a recommender – given a resource and a user – by comparing the tags the recommender suggested with the tags the user used to annotate the resource. Then recall (“Which fraction of the used tags could be suggested?”) and precision (“Which fraction of the suggested tags was used?”) quantify the quality of the recommendation. Furthermore, the logging of click events allows us to evaluate the user behavior in more detail.

6.4.1 Measures

As performance measures we use precision and recall as defined in Section 5.5.3. Therefore, we compare for each post $(u, T(u, r), r)$ the recommended tags $\tilde{T}(u, r)$ with the tags $T(u, r)$ the user has finally assigned. We can then calculate the f1-measure (f1m) as harmonic mean of precision and recall:

$$\text{f1m} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} .$$

6.4.2 Data Cleansing

Before comparing $\tilde{T}(u, r)$ with $T(u, r)$, we clean the tags in both sets according to the Java method *cleanTag* shown in Algorithm 6.1. This means, we ignore the case of tags and remove all characters which are neither numbers nor letters.⁶ Since we assume all characters to be UTF-8 encoded, the method will *not* remove umlauts and other non-latin characters. We also employ Unicode normalization to normal form KC⁷ using `java.text.Normalizer`. Finally, we ignore tags which are ‘empty’ after normalization (i. e., they neither contained a letter nor number) or which are equal to the strings *imported*, *public*, *systemimported*, *nn*, *systemunfiled*. Thus, in the following we always regard cleaned tags.

Algorithm 6.1 The Java method used to clean tags.

```
1 public String cleanTag(String tag) {
2     return Normalizer.normalize(tag, Normalizer.Form.NFKC).
3         replaceAll("[^0-9\\p{L}]+", "").
4         toLowerCase();
5 }
```

6.4.3 Logging

For further evaluating the performance of the available tag recommenders, we store in a database for each recommendation process the corresponding bookmark or publication post as well as each recommender’s recommendation, identified by a unique *recommendationID*. Furthermore, the applied selection strategy together with the selected recommenders and tags are stored.

⁶See also the documentation of `java.util.regex.Pattern` at <http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html>.

⁷<http://www.unicode.org/unicode/reports/tr15/tr15-23.html>

Several recommendation requests may refer to a single posting process (i. e., when the user pressed the ‘reload’ button or forgot to enter a required field). For identifying these correspondences, a random identifier (*postID*) is generated whenever a post or editing process is started and retains valid until the corresponding post is finally stored in the database. This *postID* is mapped to each corresponding *recommendationID*. At storage time, the *postID* together with the corresponding user name, time stamp and the intra hash identifying the resource is stored. This connects each post of each user with all referring recommendations and vice versa.

Additionally, the user interaction is tracked by logging mouse click events using JavaScript. Each click on one of BibSonomy’s web pages is logged using AJAX into a separate logging table. Information like the shown page, the DOM path of the clicked element, the underlying text, etc. is stored.⁸

6.5 Results

In this section we show by means of the two simple recommenders introduced in Section 6.3.5 which kind of evaluation the framework supports and how those two recommenders perform in practice. The analysis is based on data from posting processes between May 15th and June 26th 2009. Only public posts from users not flagged as spammer were taken into account.⁹ Since tag recommendations are provided in the web application only when *one* resource is posted, posts originating from imports (e. g., Firefox bookmarks, or BIB_TE_X files) or BibSonomy’s API are not contained in the analysis.

6.5.1 General

We start with some general numbers: In the analyzed period, 5,840 posting processes (3,474 for publications, 2,366 for bookmarks) have been provided with tag recommendations. The MP ρ -mix recommender served recommendations for 2,935 postings, the TbyU recommender for 3,006. Their precision and recall is depicted in Figure 6.4. On the plotted curve, from left to right the number of evaluated tags increases from one to five. I. e., we first regard only the tag t with the highest value $f(t)$, then the two tags with highest f , and so on. Thus, the more recommended tags are regarded, recall increases while precision decreases. In general, both precision and recall are rather low with the MP ρ -mix recommender performing better than the TbyU recommender.

⁸Note that users can disable logging on the settings page, thus not all posting processes yield clicklog events.

⁹Users can be flagged as spammers manually or by BibSonomy’s spam detection framework (Krause et al., 2008b).

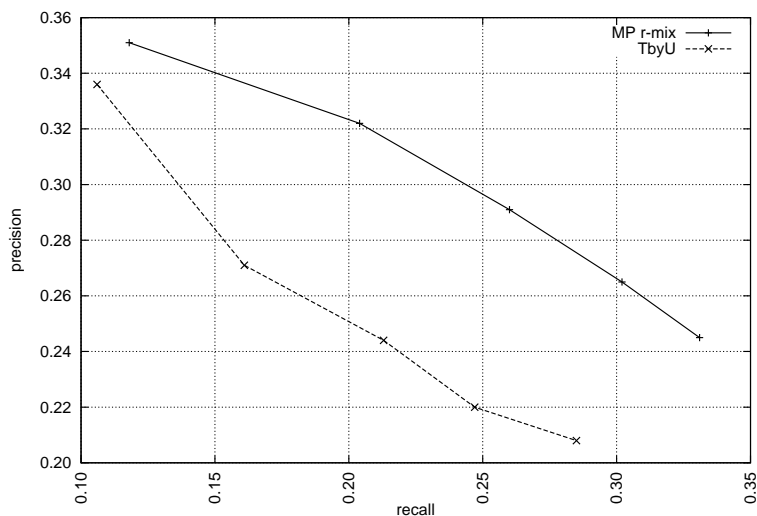


Figure 6.4: Recall and precision of the two deployed recommenders. The number of recommended tags increases from one on the left to five on the right as described in Section 5.6.

6.5.2 Influence of the ‘reload’ Button

Since users can request to reload recommendations when posting a resource, we here investigate the influence of the ‘reload’ button. Is the first recommendation sufficient or do users request another recommendation? Are recommendations which got replaced by the user pressing the ‘reload’ button worse than those shown last? Has one recommender more often been reloaded than the other?

In 767 (274 bookmark, 493 Bib_{TEX}) of the 5,840 posting processes the users requested to reload the recommendation. Thus, in around 13% of all posting processes users requested another recommendation.

Recommendations from several recommenders can be displayed during one posting process. There is the recommendation which appears directly after loading the posting page (*first*), there are recommendations which appear after the user has pressed the ‘reload’ button, and there is the recommendation shown before the user finally saves the post (*last*). Thus, given a recommender τ , we can define the set F_τ to contain those posts, where the recommender τ showed the first tags, and L_τ as the set of posts where the recommender τ showed the last tags (i. e., before the post is finally stored). For each recommender τ , we can then look at the sets $F_\tau \setminus L_\tau$, $L_\tau \setminus F_\tau$, and $F_\tau \cap L_\tau$. Posts where the user did not press the reload button are contained in both F_τ and L_τ and thus in $F_\tau \cap L_\tau$. Table 6.1 shows the result of our analysis.

For both of the two deployed recommenders and for all three sets, the table shows the

Table 6.1: The influence of the ‘reload’ button.

measure	#posts		f1m@5	
	MP ρ -mix	TbyU	MP ρ -mix	TbyU
$F_{\tau} \setminus L_{\tau}$	337	319	0.258	0.270
$L_{\tau} \setminus F_{\tau}$	331	363	0.380	0.364
$F_{\tau} \cap L_{\tau}$	2,271	2,339	0.277	0.224

number of posts in the corresponding set, and the average f1-measure at the fifth tag.¹⁰ As one can see, the number of posts where the reload button has not been pressed ($F_{\tau} \cap L_{\tau}$) is quite large for both recommenders (around 2,300). There is also only little difference in the number of posts for the recommenders over the different sets, besides the higher number of posts for the TbyU recommender in the $L_{\tau} \setminus F_{\tau}$ set. It contains those posts, where the user requested to reload the recommendation and where the recommender at hand delivered the last recommendation. Thus, the TbyU recommender more often provided the last recommendation than the MP ρ -mix recommender.

The most noticeable observation is the good performance of both recommenders for the posts where a reload occurred and the recommender showed the last recommendations ($L_{\tau} \setminus F_{\tau}$). There, both precision and recall are much higher than for the other two sets. This suggests that the first suggestion was rather bad and caused the user to request another recommendation which indeed better fitted his needs. The worse values for the $F_{\tau} \setminus L_{\tau}$ set also support this thesis. A noteworthy difference between the two recommenders is the performance of the TbyU recommender for the $F_{\tau} \setminus L_{\tau}$ set which is better than its overall performance (i. e., on the $F_{\tau} \cap L_{\tau}$ set). This could be an indicator that those users which actively used the recommender (by pressing the ‘reload’ button) took better notice of this recommender’s tag suggestions.

The usage of the ‘reload’ button seems to be a good indicator for the interest of the user in the recommendations. However, the data we gathered during the one month evaluation period is still rather sparse, thus no final conclusions can be drawn.

6.5.3 Logged ‘click’ Events

Next we evaluate the data from the log which records when a user clicked on a recommended tag (cf. Section 6.4.3). Clicks are rather sparse: in only 1,061 (485 bookmarks, 576 publications) of the 5,840 posting processes, users clicked on a recommendation.

First, we want to answer the questions “How is clicking distributed over users?” and “Are there users which always/never click?”. Figure 6.5 shows the users sorted by the fraction of posting processes at which they have clicked on a recommended tag. The size

¹⁰We omit precision and recall, since whenever the f1m for one set was better/worse than for another set, precision and recall were better/worse, too.

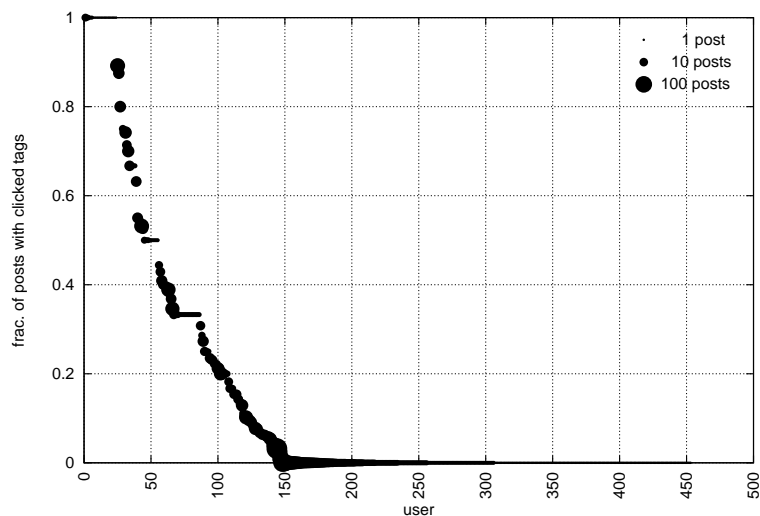


Figure 6.5: Users sorted by their fraction of click/noclick-posts. The y-axis depicts the fraction of posts where recommended tags were clicked. Each circle represents a user. As shown in the scale at the upper right corner, the size of each circle depicts the logarithm of the user's number of posts regarded for the analysis.

of each circle depicts the logarithm of the user's number of posting processes incorporated into the analysis. Closer to the left are users which in almost all posting processes clicked on a recommendation; users closer to the right never clicked a recommended tag during posting. Only around 150 users clicked on a tag and half of the remaining users are represented by only one post. This could mean that only after some time users discover and use the recommendations. However, there are also some active users which almost never clicked on a recommendation.

In Figure 6.6 we see for each number of recommended tags (from one to five), the fraction of matches which stem from a click on the tag (instead of manual typing). For the TbyU recommender, around 35% of the matches come from the user clicking on a tag. Thus, although users infrequently click on tags, a large fraction of the correctly recommended tags of that recommender has been clicked instead of typed. Why there is a difference of around 15% between the two recommenders with a higher click fraction for the TbyU recommender (in contrast to its worse precision and recall) is not clear. One explanation could be the different sources of tags the two recommenders use: while the $MP\rho$ -mix recommender delivers popular tags the user might have used before and thus can easily type, the TbyU recommender also suggests new and probably complicated tags extracted from the title which are easier to click than to type.

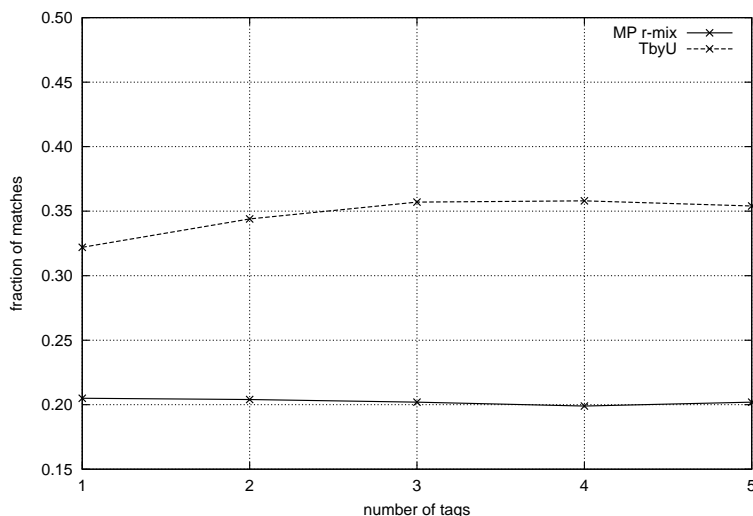


Figure 6.6: The fraction of matching tags which have been clicked.

6.5.4 Average F1-Measure per User

Which properties of a posting process could help a multiplexer strategy to smartly choose a certain recommender instead of randomly selecting one? We here focus on the user only – other characteristics could be likewise interesting (e. g., resource type or the recommended tags). Figure 6.7 shows the average f1m of the MP ρ -mix recommender versus the average f1m of the TbyU recommender for each of the 380 users¹¹ in the data. In the plot, each user is represented by a circle whose size depicts the logarithm of the user’s number of posts.

The most interesting users are reflected by the circles farthest from the diagonal, i. e., those users who have a high f1m for one but a low f1m for the other recommender. As one can see, such users exist even at higher post counts. Once such a user is identified, one could primarily select recommendations from the user’s preferred recommender, e. g., by increasing the probability for randomly selecting the recommender.

6.6 The ECML PKDD Discovery Challenge 2009

Continuing the analysis presented in the previous section, we now focus on a larger setting: The framework was the cornerstone of the ECML PKDD¹² Discovery Challenge

¹¹Only users which got recommendations from *both* recommenders were taken into account.

¹²The *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery and Data Mining* is according to its website (<http://www.ecmlpkdd.org/content/past-conferences>) the “largest European conference in these areas”.

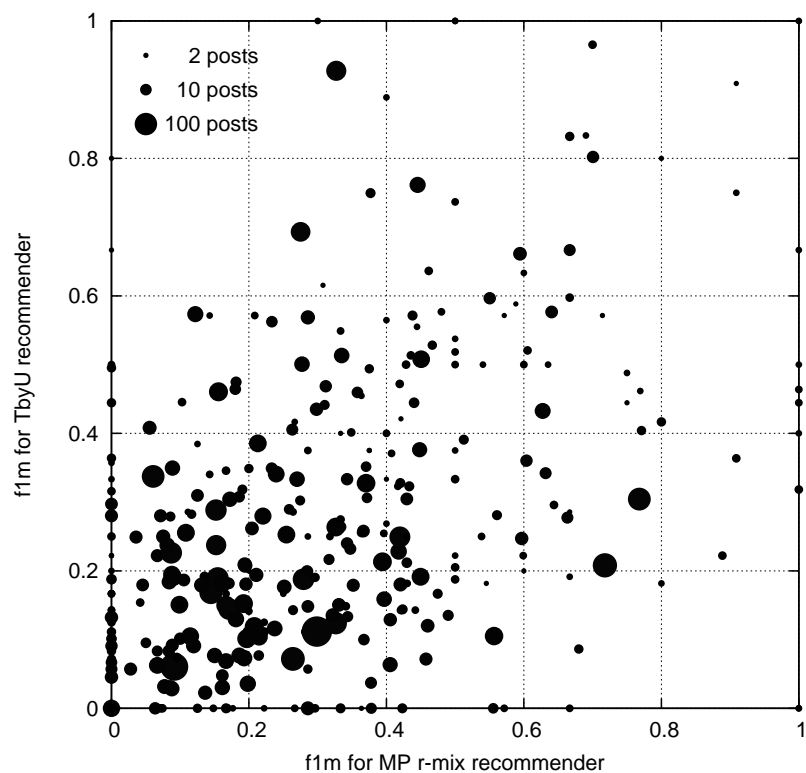


Figure 6.7: Average f1-measure for each user and recommender. Each circle represents a user as described for Figure 6.5.

2009 (Eisterlehner et al., 2009) where one task required the participants to deliver online recommendations to BibSonomy. This was a larger stress test for external recommenders and the framework itself. In this section we give a brief overview on the setting, the methods some recommenders used and the resulting recommendation performances.

6.6.1 Setting

The participants implemented recommenders which were integrated into the framework using instances of the remote recommender. Overall, ten participants from seven countries deployed 13 recommenders – seven of them (from four participants) were running on machines in BibSonomy’s network, the remaining six were distributed all over the world (amongst other countries, in Canada and South Korea). All recommenders had to adhere to a timeout of 1000ms between the sending of the recommendation request and the arrival of the result. If they failed to deliver their recommendation in time, we set precision and recall for the corresponding post to zero and showed the user another

Table 6.2: The number of posts regarded for evaluation.

recommender-id	3	5	6	7	12	13	14	16
#posts	347	391	361	415	385	380	370	398

recommendation.

The recommendations were evaluated over the period from July 29th to September 2nd 2009. During that time, more than 28,000 posting processes had to be handled, where each recommender was randomly selected to deliver recommendations for at least 2,000 processes. For evaluation we regarded only public, non-spam posts and therefore the results in the next section are based on approximately 380 relevant posting processes per recommender (for the exact counts, see Table 6.2).

Although 13 recommenders participated in the online task, only eight of them managed to deliver results in at least 50 % of all requested posting processes. The remaining five recommenders answered only in less than 5 % of all cases and are thus ignored in Table 6.2 and in the figures and discussion following in Section 6.6.3.

6.6.2 Methods

Details on the tag recommendation methods evaluated during the challenge can be found in the proceedings (Eisterlehner et al., 2009). Here we only briefly introduce the three best recommenders of the online task.

The winning recommender 6 (Lipczak et al., 2009) uses a method based on the combination of tags from the resource’s title, tags assigned to the resource by other users and tags in the user’s profile. The system is composed of six recommenders and the basic idea is to augment the tags from the title by related tags extracted from two tag-tag-co-occurrence graphs and from the user’s profile and then rescore and merge them.

Recommender 3 (Si et al., 2009) performs so called “Feature Driven Tagging” by extracting and weighting features like words, ids, hashes, phrases from the resources. Each feature then generates a list of tags. The weight of the features is estimated using $TF \times IDF$ and $TF \times ITF$ (term frequency \times inverted document frequency and term frequency \times inverted tag frequency – see (Baeza-Yates and Ribeiro-Neto, 1999)); the tags of the features are determined using co-occurrence counts, mutual information, and χ^2 statistics.

For recommender 5, Cao et al. (2009) divide the posts in four categories, depending on the case if the user or resource of the post is known or not. Then, for each category they learn a model to rank the tags using a ranking SVM. To augment the available tags for posts (besides the full text and the tags of the resource), the authors use post-content similarity and k -Nearest-Neighbors.

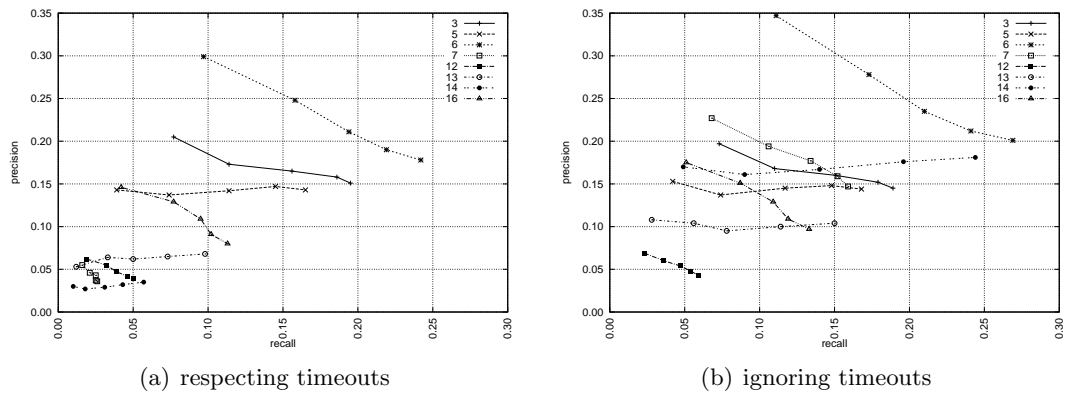


Figure 6.8: Recall and precision of the deployed recommenders.

6.6.3 Results

Overall Performance. First, we have a look at precision and recall of each recommender in the evaluation mode relevant for the challenge (Figure 6.8(a)). For a posting process in which the recommender could not deliver a recommendation in time, precision and recall were set to zero. In this setting, recommender 6 (Lipczak et al., 2009) is the clear winner with an f1m of 0.205 for five tags. The performance of the remaining methods varies between an f1m of 0.030 and 0.171 for five tags – all those recommenders have a recall of less than 0.2.

Influence of the Recommendation Latency. If we disregard the timeout limit of 1000 ms and also evaluate the suggestions which came later (cf. Figure 6.8(b)), we get a different picture. Of course, all recommenders improve – but in particular recommender 14 gains both precision and recall. This can be explained by the latency plot shown in Figure 6.9. It shows for each recommender the latency of the delivered recommendations for the selected posting processes, ordered in ascending order by latency. The curves do not reach 100% because in some cases the recommenders did not deliver a result at all. One can see that recommender 14 returned a suggestion in almost as many posts as the winning recommender 6. However, only 20% of the posts were delivered in time – in contrast to almost 80% of the posts for recommender 6. Consequently, timeouts are a serious issue in this setting – with a timeout of 2000 ms, the competition for the best performance would have been much closer. Nevertheless, a timeout of 2000 ms would be too long for recommendations which shall be shown after loading the page. One should also note that in principle network latency was not an issue since the winning recommender was located in Canada.

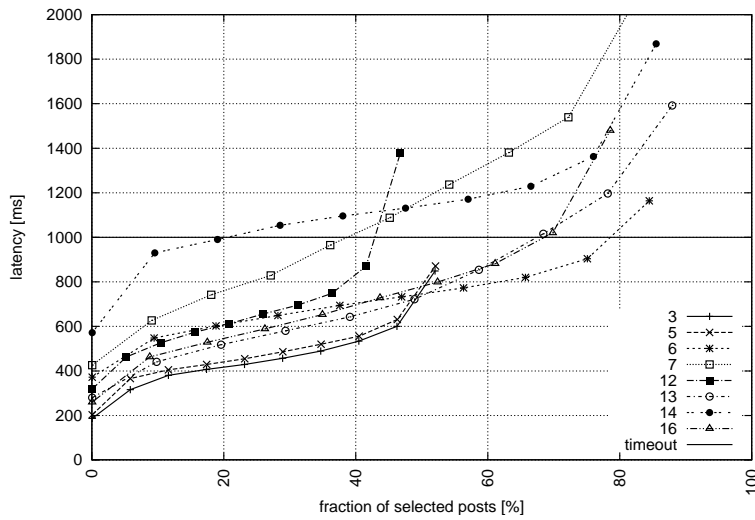


Figure 6.9: Latency of the recommenders. No curve reaches 100% because none of the recommenders delivered results for all posting processes they have been selected for.

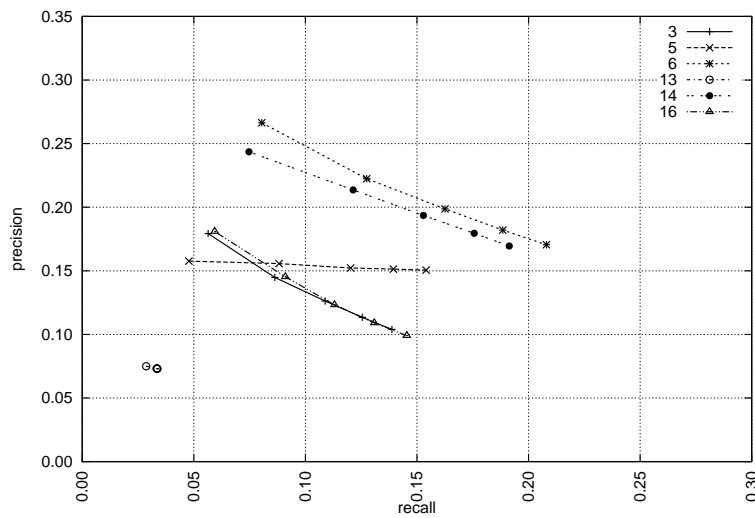


Figure 6.10: Performance in the offline task (recommenders 7 and 12 did not participate).

Comparison with Offline Results. Before the participants tested their recommenders in the online setting, most of them performed an offline evaluation against a dataset from BibSonomy. Interestingly, some recommenders gained better results in the online challenge than in the offline challenge (see Figure 6.10). Without going too much into

detail, one explanation could be the fact that in the online challenge the user actually *saw* the recommender's suggestion and thus had the chance to utilize it. This suggests that users actively used the recommendations and are indeed influenced by them.

6.7 Conclusion

In this chapter we presented the tag recommendation framework that we developed for BibSonomy. It allows us to not only integrate and judge recommendations from various sources but also to develop clever selection strategies. A strength of the framework is its ability to log all steps of the recommendation process and thereby making it traceable. E. g., the diagrams and tables presented in this chapter are automatically generated and will be integrated in a web application for analysing and controlling the framework and its recommenders.

As the results in Section 6.5 show, there is no clear picture which of the two recommendation methods performs better. There is a dependency on the number of regarded tags, the user at hand, and also slightly on the moment of recommendation. This suggests that we can achieve better performance not only by adding improved recommendation methods but also by implementing adaptive selection strategies. In case of the user dependency, one could prefer the better performing recommender by increasing its selection probability or even couple the probability with the current recommendation quality.

The Discovery Challenge allowed us to evaluate the framework in a larger setting. It passed that stress test and gave us important insights into the handling of timeouts and distributed recommendations. An interesting finding is the better performance of most recommenders in the online setting compared to their offline performance. Future tag recommendation challenges and evaluations should take this into account and probably consider performing an online instead of an offline evaluation to get more realistic results.

Chapter 7

Community Support for the Social Semantic Desktop

The realization of a new kind of computer desktop paradigm – the *Social Semantic Desktop* – is the aim of the Nepomuk project (Groza et al., 2007) in which we implemented parts of the community support architecture. Thereby, we transferred the insights on tag recommendation methods we gained in Chapter 5 to a network of peers where users annotate arbitrary resources on their desktop. The work in this chapter is based on the Nepomuk project deliverables D5.1, D5.2, and D5.3 (Demartini et al., 2006, 2007; Stecher et al., 2008), the details on the underlying methods are described in (Hotho et al., 2006b; Jäschke et al., 2008c).

7.1 Introduction

Current computer desktop systems lack support to connect different pieces of information which semantically belong together, e. g., the author of a document is not connected with the author’s entry in the address book, or a web page is not related to the files one has downloaded from it and stored on disk. Typically, the user has only the tree-like folder structure of the file system to organize her documents and proprietary, unconnected files or databases for contacts, e-mails, tasks, bookmarks, etc.

The vision of a new kind of desktop paradigm – the *Social Semantic Desktop* (Decker and Frank, 2004) – describes the idea of a computer desktop where files, emails, contacts, tasks, etc. are connected by technologies developed for the Semantic Web (Berners-Lee et al., 2001) and where these information items are linked across desktops by a (peer-to-peer) network. This aims to allow users to get a topical view of their data, e. g., to access all information pieces belonging to a certain project or task at once and thus spend less time filtering and filing information (Decker and Frank, 2004). The central step to accomplish this goal is the “transfer [of] the Semantic Web to desktop computers” (Sauermann et al., 2005) – not only the technology, but the philosophy behind it – to identify resources on the desktop with Uniform Resource Identifiers (URI) and connect them in an RDF (Resource Description Framework) graph.

The Nepomuk project (Groza et al., 2007)¹ aims at implementing and pushing the

¹<http://nepomuk.semanticdesktop.org/>

vision of the social semantic desktop by integrating technologies from the computer desktop, the Semantic Web (definition and exchange of metadata and ontologies; inferencing), and peer-to-peer (P2P) networking (large networked communities without centralized infrastructure). In the context of Nepomuk, we focused on two aspects of the Social Semantic Desktop's *social network and community services* – which according to Decker and Frank (2004) comprise *community and interaction support* – namely tag recommendations and community detection and labeling.

In this chapter we describe the architecture implemented to facilitate these tasks within the Nepomuk framework. We start by motivating our approach (Section 7.2) and briefly review related work (Section 7.3). Then, in Section 7.4, we introduce the Nepomuk architecture and continue in Section 7.5 to give a brief overview on how the developed components integrate into that architecture. The realization of and the interaction with our two developed components TagRecommender (Section 7.6) and CommunityManager (Section 7.7) is the topic of the following two sections. Further details can be found in the corresponding project deliverables (Demartini et al., 2006, 2007; Stecher et al., 2008).

7.2 Motivation

Collaborative tagging systems have acquired large numbers of users, who have created huge amounts of information within less than two years. The frequent use of these systems shows clearly that web- and folksonomy-based approaches are able to overcome the knowledge acquisition bottleneck, which was a serious handicap for many knowledge-based systems in the past. Tagging is integrated in Nepomuk as one way to annotate resources, since it is a very simple form of annotation (in contrast to building and maintaining an ontology) which can easily be understood and adopted by the users. Applications like a wiki allow users to annotate their resources (e. g., wiki pages) with tags; keywords may be extracted from pictures users saved on Flickr or BIBTEX references they managed with BibSonomy. Hence, a rich amount of tagged resources will be available on the local desktop of each user. To store the tagging information in the local RDF repository and facilitate sharing of it in the peer-to-peer network, the tag annotations are mapped to RDF using NAO, the Nepomuk Annotation Ontology (cf. Section 7.5.2). Thus, since the users can share their (tagging) information within the peer-to-peer network, a folksonomy is inherently present in the network of peers which contains the aggregated information of its users.

In contrast to the automatically extracted metadata of the extractors contained in Nepomuk, the advantage of tags is that they are manually generated by the user. Often, tags represent the users' opinion about a certain resource or how it relates to a topic. Thus, tagging data is a good foundation to detect communities of users with similar taste, opinion, or knowledge. Furthermore, such analysis typically relies upon large

amounts of available data. The popularity of tagging in web based applications suggests that similarly large folksonomies can grow in the network of social semantic desktops and thus be a rich source of data.

The benefits of tag recommendations as described in Chapter 5 also apply to the social semantic desktop scenario. However, since adding tags to a resource on the desktop is a voluntary task, a tag recommender's ability to encourage usage of the tagging facility plays an important role. It also facilitates harmonization of the vocabulary across users and further speeds up the tagging process. Thereby, tag recommendations are another building block of Nepomuk's social services.

7.3 Related Work

Another prominent approach to link various information pieces is Haystack (Quan et al., 2003). As an integrated environment it replaces many applications (word-processor, email client, etc.) to allow better linkage of information. This is in contrast to the approach chosen in Nepomuk, where the goal was to provide a framework for applications to allow better knowledge exchange.

In many ways the predecessor of Nepomuk is Gnowsis (Sauermaun, 2003), a semantic desktop implementation centered around a local semantic web server. Gnowsis pushed the idea of integrating various existing data sources through so called *data wrappers*. They extract metadata from email clients, addressbooks, the file system, etc. and store it in RDF format in a local repository.

Integration of ranking – or in general, data mining – methods on the desktop has become popular with the ascent of desktop search engines like Google Desktop,² or Beagle.³ An extension of Beagle which adds semantic search features is Beagle++ (Chirita et al., 2005). It restores the semantic relations between different entities which have been extracted from crawled documents by using entity identification mechanisms. Furthermore, Beagle++ uses the ObjectRank (Balmin et al., 2004) algorithm to compute a ranking on the extracted metadata.

7.4 Overview of the Nepomuk Architecture

Nepomuk's architecture (Reif et al., 2008) of the Social Semantic Desktop is based on several *services* – like messaging or sharing – that provide the wanted functionality. Figure 7.1 (based on Reif et al., 2008) gives an overview of the services implemented within Nepomuk. The different services are grouped by their functionality into *Social Services*, *Semantic Desktop Services* and *Extension Services*. They are not applications on their own, but accessed via *Enduser Applications*. For each service, there is at

²<http://desktop.google.com/>

³<http://beagle-project.org/>

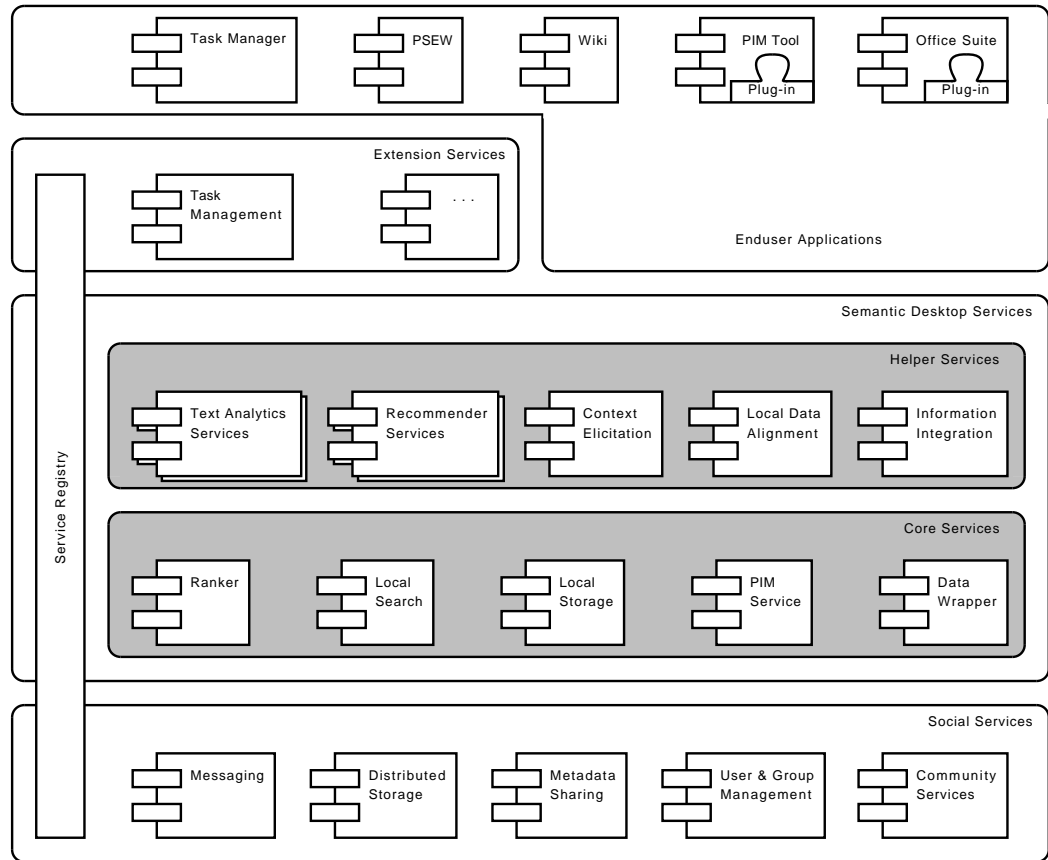


Figure 7.1: Services of Nepomuk's implementation of the Social Semantic Desktop.

least one concrete *component* implementing it. For example, the *CommunityManager* component introduced in Section 7.5 provides a *Community Service*. Since for most services there is just one component implementing it, we often use the terms *component* and *service* interchangeably. In this section we give a brief overview on the different types of services, enduser applications, and the *Service Registry* as central piece connecting the different services.

7.4.1 Semantic Desktop Services

These services facilitate the stand-alone *Semantic Desktop* without an interaction across desktops. They are further divided into *Core Services* and *Helper Services*.

Core Services

The *Core Services* provide the main functionality that is required in every installation of Nepomuk. For example, all metadata is stored in an RDF repository that facilitates a *Local Storage* service – as does the local file system of the computer. Another central facility is the *PIM Service* which allows the user to manage his personal information model ontology (PIMO) – an ontology that is stored in the RDF repository, too.

Helper Services

Services that are not required in every setup are regarded as *Helper Services*. They support the user to extract information from texts (*Text Analytics Services*), or to merge ontologies and metadata (*Local Data Alignment*).

7.4.2 Service Registry

The glue between the different services that allows them to interact is the *Service Registry*. It provides essential functionalities to register, detect and call components that make certain services available. Upon startup, every component registers itself at the Service Registry, such that other components can find and use it; when a request to another component is necessary, the Service Registry provides an instance of it. Finally, a SOAP connector for each component is provided by the Service Registry. The connector allows programmers to easily expose the methods of their components via the SOAP protocol.⁴ Thereby, local applications and components running outside the Nepomuk environment can access components using the SOAP protocol.

7.4.3 Social Services

At the heart of the *Social Semantic Desktop* are the *Social Services*. They facilitate the exchange of metadata between peers (*Metadata Sharing*) and allow services running on different peers to communicate with each other (*Messaging*). The users can also exchange files using the *Distributed Storage* and create groups of users with the *User & Group Management* service. *Community Services* are provided by the tag recommendation and in particular the community detection and labeling components described in this chapter. Both components analyze the shared metadata of the community of peers and strengthen knowledge exchange between users.

7.4.4 Extension Services

Services which augment the functionality of Nepomuk but are not necessarily part of it are called *Extension Services*.

⁴<http://www.w3.org/TR/soap12-part1/>

7.4.5 Enduser Applications

Ideally, all enduser applications support and use the services supplied by the Social Semantic Desktop implementation. However, until standard software (like Microsoft Office) will include such support, adapters in form of *plug-ins* are required. Thus, there are two types of enduser applications: those provided by the Nepomuk consortium, like the *Task Manager*, which natively builds upon Nepomuk's services, and standard software, like a *PIM tool*, which is integrated into the Nepomuk architecture via plug-ins.

Among the applications supplied by the Nepomuk consortium is a workbench which provides graphical user interfaces for some of the services and also allows users to manage their personal information model ontology. This GUI is PSEW, the *Personal Semantic Eclipse Workbench*, which is used in Section 7.6 as an exemplary application for interacting with the TagRecommender component.

7.5 Community Services Architecture

Within the Nepomuk architecture, we established community services that facilitate community driven tag recommendation (Section 7.6) and community detection (Section 7.7). This functionality is provided by the components *TagRecommender*, *CommunityManager*, and *FolkPeer*. As all services in Nepomuk, the components are not applications on their own, but they form an infrastructure which can be adopted and used by other applications. The *TagRecommender* provides tag recommendations for resources, tags, and users; the *CommunityManager* provides detection and labeling of communities of users; the *FolkPeer* as backend of the two former components analyzes the folksonomy found in the network of Nepomuk peers and provides methods for other components or applications to access the results. This section provides an overview on which concrete functionalities the components provide, how they are integrated into the Nepomuk architecture, and what their dependencies to other services are.

7.5.1 Component Descriptions

Here we describe the functionality of the three implemented components *TagRecommender*, *CommunityManager*, and *FolkPeer*.

FolkPeer

The central component of the community services is the *FolkPeer*. It implements the community detection and tag recommendation algorithm and handles the underlying data. Typically, the FolkPeer runs on a distinguished peer in the network capable of handling the expected amount of data and computation. Nevertheless, it is possible to have several FolkPeers, e. g., for different subgroups in the P2P network which share their

(meta)data in separated distributed indexes. More precisely, the FolkPeer is responsible for the following tasks:

- *Selection and collection of data.* As basis for the analysis we use the metadata the users shared within the P2P network the FolkPeer is connected to. The component regularly queries the Metadata Sharing service for appropriate metadata and collects it in its own data warehouse. For details see Section 7.5.2.
- *Computation of the FolkRank algorithm.* The FolkRank algorithm (cf. Section 5.3.2) is used to compute both tag recommendations and communities of users. For each incoming request, the FolkPeer computes a ranking on the collected data and further processes it.
- *Listening for requests.* Applications and components on other peers can communicate with the FolkPeer using the Messaging service. During startup, the FolkPeer registers a listener and waits for requests.

TagRecommender

As provider of social tag recommendations (in contrast to content- or metadata-based tag recommendations) functions the *TagRecommender* component. Therefore, it is both a *Recommender Service* and a *Social Service*. Running locally on each peer, it exposes the methods *getTagsForResources* and *getTagsForTags*. Each method forwards a request to the FolkPeer which computes the corresponding tag recommendation and returns the result – an ordered set of tags – to the TagRecommender.

CommunityManager

The *CommunityManager* works very similar to the TagRecommender: it runs locally on each peer and forwards all requests to the FolkPeer. The two methods *getCommunityForTags* and *getCommunityForUsers* allow components to fetch communities around a given set of tags or a given set of users, resp. In both cases the returned result is a community consisting of a set of users and a set of tags. The users are the members of the community, the sets are the labels which describe it.

7.5.2 Gathering Data for Analysis

The peer-to-peer network of Nepomuk allows the users to exchange metadata between peers (Demartini et al., 2007). This metadata is represented as (subject, predicate, object) triples in an RDF graph that can be queried by the peers of the network using the *Metadata Sharing* service. To respect the privacy of the users, our components rely on the shared metadata only and do not inspect the users' local storage (e. g., , the RDF

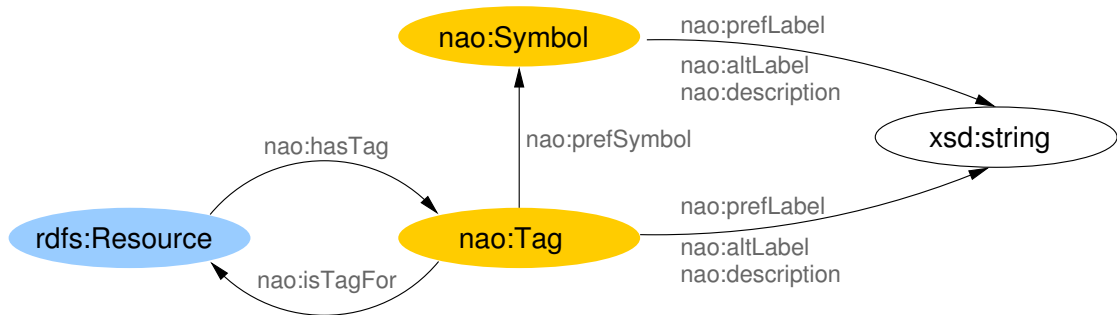


Figure 7.2: Tagging as modeled in NAO (Scerri et al., 2007).

```

PREFIX nao: <http://www.semanticdesktop.org/ontologies/2007/03/nao#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?url ?tag
WHERE {
  ?url nao:hasTag ?y .
  ?y nao:prefLabel ?tag .
  ?y rdf:type nao:Tag
}

```

Figure 7.3: Example SPARQL query used to gather tagging data.

repository). Thus, the FolkPeer collects the data necessary for analysis by extracting relevant information from the shared metadata of the P2P network.

In Nepomuk tagging data is represented in RDF with the vocabulary of the Nepomuk Annotation Ontology (NAO) (Scerri et al., 2007). There the tag assignments are modeled with the ‘hasTag’ relation, as can be seen in Figure 7.2. It relates a resource to another resource of class ‘Tag’ which has attached a user readable string by the properties ‘prefLabel’, ‘altLabel’, or ‘description’. The relevant (resource, tag) tuples are received by the FolkPeer from the Metadata Sharing service with a query similar to the SPARQL⁵ query depicted in Figure 7.3. Additionally, the Metadata Sharing service provides for each retrieved tuple information about the user that shared it.

7.5.3 Relation to Other Services

In a system like Nepomuk, where tasks are distributed among services, naturally, also the three described components depend on other services. Hence, we here provide an overview on the CommunityManager’s, FolkPeer’s and TagRecommender’s relation to

⁵SPARQL Protocol and RDF Query Language

other services. A more detailed description of the interaction between the different services can be found in Sections 7.6 and 7.7.

Messaging

The Messaging service is responsible for the communication of components across peers. It provides a network-independent abstraction of messaging methods which in its current implementation is based on XMPP.⁶ Both the TagRecommender and the Community-Manager use the Messaging service for communicating with the FolkPeer.

Metadata Sharing

The sharing of metadata between peers is a crucial property of the Nepomuk infrastructure. It is accomplished by the Metadata Sharing service – a layer on top of the peer-to-peer network provided by the Distributed Storage service (Darlagiannis et al., 2008). Access to this service is embedded into desktop applications to allow users to share the metadata of their resources with other peers. In our scenario, this service plays a central role as provider of the data which is the basis for the community detection and tag recommendation tasks, since for privacy reasons, only metadata the users explicitly shared with other users is available for such analysis.

Services Delivering Data as Input for the Analysis

Applications and services like the *Wiki*, the *Task Manager* or the *Data Wrapper* are one of the main providers of metadata stored in the RDF repository. Among the metadata, user annotated resources which can be used for the analysis are widely available, e. g., the Wiki allows users to tag pages, the Task Manager to assign keywords to tasks, meetings, etc., and the Data Wrapper might gather the user's bookmarks from an existing collaborative tagging service or extract keywords from BIB_T_E_X files. However, it is necessary that this information is shared by the user in the peer-to-peer network to make it available for the social services of Nepomuk.

7.6 Tag Recommendations

In this section we describe the *TagRecommender* component which provides tag recommendations based on the network of tags used by other users of the Nepomuk P2P network to annotate resources. Since our analysis in Chapter 5 has shown that the FolkRank algorithm yields good tag recommendations, we integrated this method to compute social tag recommendations in Nepomuk. We focus on the integration of the

⁶<http://xmpp.org/>

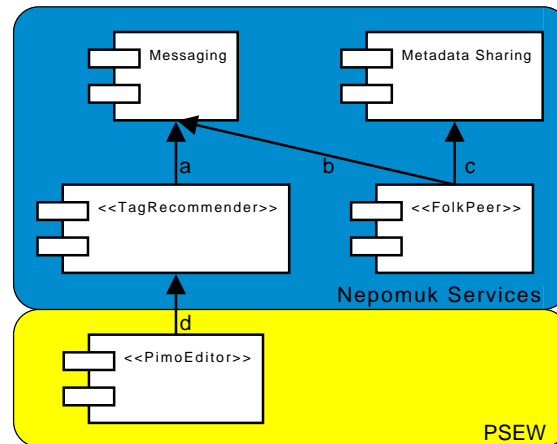


Figure 7.4: The dependency graph of the components *TagRecommender* and *FolkPeer*.

TagRecommender into PSEW, the Personal Semantic Eclipse Workbench, and describe possible interactions of the users with it.

7.6.1 Realization

Tag recommendations are realized by the integration of the FolkRank algorithm (Hotho et al., 2006b) into the *FolkPeer*, a component that can be accessed by other services via the *TagRecommender*. As described in Section 5.3.2, for a given user and resource, we compute a tag recommendation with FolkRank by setting a high value for the user and the resource in the preference vector. Then, the top-ranked tags are used as recommendation. Additionally, we compute recommendations for tags the user has already entered by increasing the preference values of those tags (and the user) and returning the top-ranked tags that are different from the given tags. Further implementation details regarding the TagRecommender, can be found in Deliverable 5.2 (Demartini et al., 2007), regarding the FolkPeer we refer to (Demartini et al., 2006). Here we focus on the integration of the TagRecommender into the Nepomuk framework and its interaction with other services.

Figure 7.4 shows the direct dependencies of the components *TagRecommender* and *FolkPeer* to other services. There is a distinction between services (dark box) and enduser applications that are part of PSEW (light box). The names of concrete implementations of services in the form of components are enclosed in `<<>>`. The figure depicts the dependencies with labeled arrows which point to the services/components that are needed by the originating components to work. The TagRecommender uses the Messaging service for communication with the FolkPeer (a and b). Since the FolkPeer is not necessarily running on the same peer as the TagRecommender, the Messaging service

allows the two components to exchange information between peers. The FolkPeer uses information about tagged resources available in the P2P network using the Metadata Sharing service (c) to analyze the tagging network to compute tag recommendations. Finally, the PimoEditor (which is described in the next section) requests tag recommendations from the TagRecommender (d).

When the user who is annotating a resource in the PimoEditor starts to enter some tags describing it, the PimoEditor requests recommendations from the TagRecommender (giving it the resource or the tags the user already entered). A message is then sent by the TagRecommender to the FolkPeer using the Messaging service. The FolkPeer answers with a set of recommended tags it computed by analyzing the regularly crawled tagging information of the P2P network the Metadata Sharing service offers access to. Those tags are then shown in the frontend to the user as suggestion close to the tag input form (cf. the right half of Figure 7.5).

7.6.2 Interaction

As an example showing the possibilities of Nepomuk and the implemented tag recommendation component, we show how users can interact with the TagRecommender using PSEW. Therefore, we describe the scenario of a user who wants to edit his personal information model using the *PimoEditor*, a component that is a part of PSEW. The PimoEditor allows users to add, edit, and remove concepts and relations of their personal information model ontology (PIMO). In the scenario at hand, the user wants to annotate the resource ‘London’ he has earlier added to his PIMO. Therefore, he starts PSEW and accesses the PimoEditor by following the “Window > Perspectives > PIMO” click path in the menu and thereby switches to the Pimo perspective. The resulting window looks similar to the one shown in Figure 7.5.

Assuming the user has already created a resource ‘London’ as instance of the class ‘City’, he can then annotate that resource by clicking on it in the class browser shown on the left side of Figure 7.5 and then follow the ‘edit!’ link located in the PimoEditor at the right half of the window. At the top of the PimoEditor then an input box named “Annotate! add tags and relations. comma-separated” appears. Clicking into that input box and entering the string ‘england’ triggers the PimoEditor to find some annotations relevant for the resource ‘London’. It thereby also contacts the TagRecommender to provide recommendations for the resource and the already entered tag ‘england’.

The tags recommended by the TagRecommender can be seen in Figure 7.5 below the input box into which the user entered the string ‘england’. They contain highly relevant tags like ‘uk’, ‘london’, ‘travel’, and ‘guide’. The user can now add the relevant tags by clicking on the ‘accept’ link close to them. This adds those tags to the user’s personal information model and connects them with the resource ‘London’.

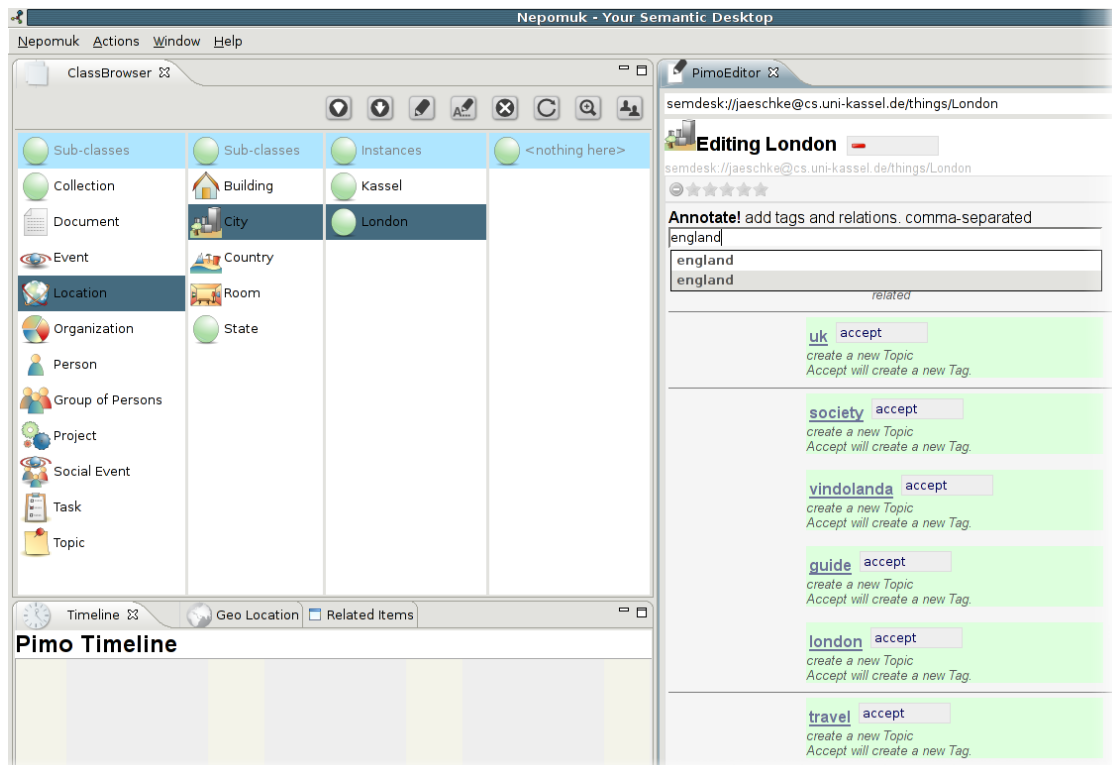


Figure 7.5: Recommendations in the *PimoEditor* returned by the *TagRecommender* when annotating the concept *London*.

7.7 Community Detection and Labeling

This section explains how the community detection and labeling task in Nepomuk is realized by the integration of the *CommunityManager* component into PSEW. Furthermore, a step-by-step walk through PSEW shows the interaction with the *CommunityManager*.

7.7.1 Motivation

The detection and labeling of communities can improve the workflow of the individual Nepomuk user, as can be seen in the following example. Imagine a scenario from Mandriva⁷ which describes how Kim⁸ searches for help on the Mandriva Club web site to connect a harddisk with a relatively new technology. The web page presenting the search results could also show Kim a community of users related to the search terms he

⁷One of the project partners of Nepomuk.

⁸A fictitious person that represents a group of several users with similar usage behaviour.

entered. When Kim clicks on the community, he finds options to contact the community members and ask them for help. The users were found by the *CommunityManager* as belonging to that community because they tagged the same or similar resources with the same or similar tags. Hence, they seem to be interested in the kind of problem Kim has and might even have solved similar problems. Kim finds André in the community list and knows his name from other solutions he has contributed and which were very helpful for him. Thus, Kim decides to contact André and ask him for advice. If André does not know a solution, Kim could send a message to the top community members and ask them for advice.

The *CommunityManager* together with the *FolkPeer* constitutes a prototypical implementation to detect communities of collaborators who share common interests. It makes their structure explicit, thus enabling various additional enhancements, such as easier finding and joining of relevant communities.

A community of users surrounding a user or a given topic can technically be described as a set of users which all have a low distance to the given user or topic. Thus, a method that can compute distances between users or between users and topics allows to form communities by ordering the users according to their distance and then regarding the closest users as members of a community. One such method is ranking: if we rank the relevance of users to a given tag, the rank of a user can be interpreted as the distance between the user and the tag. Similarly, we can compute distances between users by ranking users according to their relevance for a given user. A community can then be extracted by regarding the top users of the ranking only.

Using ranking for community detection has some advantage: we can affect the size of the community as it is necessary for the relevant application. Thus, applications can decide how many users of the community they use. Furthermore, a user can be in several communities at the same time – with different degrees of confidence for each, depending on the particular rank of the user. Hence, applications can present several communities a user is member of and along the way support different aspects of his interest.

7.7.2 Realization

Similar to the tag recommendation task described in Section 7.6, the community detection in Nepomuk is realized by the integration of the *FolkRank* algorithm (Hotho et al., 2006b) into the *FolkPeer*, and exposition of the corresponding community detection methods via the *CommunityManager* component. For further implementation details regarding the *CommunityManager* or the *FolkPeer*, refer to (Demartini et al., 2006). Here we focus on the integration of those components into the Nepomuk architecture and their interaction with other services.

As described in the previous section, we can use the ranking computed by *FolkRank* to detect communities of users. Given a tag for which we want to find a set of relevant users, we set a high preference for that tag and return the top users in the ranking

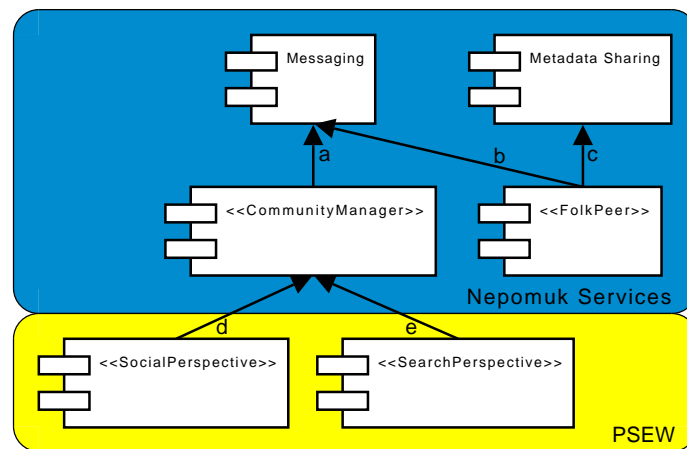


Figure 7.6: The dependency graph of the components *CommunityManager* and *FolkPeer*.

as the community. Additionally, the top tags from that ranking are used as labels for the community. The community for a given user is computed in a similar fashion by increasing the preference weight of that user.

Analogous to Figure 7.4, Figure 7.6 shows the direct dependencies of the components *CommunityManager* and *FolkRank* to other services. The Messaging service allows the *CommunityManager* to communicate with the *FolkPeer* (a and b). Again, the Metadata Sharing service provides the folksonomy data which the *FolkPeer* uses for community detection (c). One of the PSEW perspectives which enables the user to access the social Nepomuk services is the *SocialPerspective*. It allows users to find communities around topics (d). The *SearchPerspective* integrates the *CommunityManager* as one source for search, as described in detail in the following section.

When a user requests to detect a community using the facilities of the *SocialPerspective* or the *SearchPerspective*, both components forward the request to the *CommunityManager*. The *CommunityManager* sends a message to the *FolkPeer* using the Messaging service. The *FolkPeer* then computes a ranking on the data it gathered regularly from the Metadata Sharing service and extracts a community. Finally, the result is send back to the *CommunityManager* using the Messaging service.

7.7.3 Interaction

In the following, we describe from an end-user's point of view, how to interact with the *CommunityManager* by using the PSEW interface of Nepomuk. The user can detect communities of other users using the *search* perspective of PSEW. Having explored the detected community, the user can create a group from it for sharing resources among

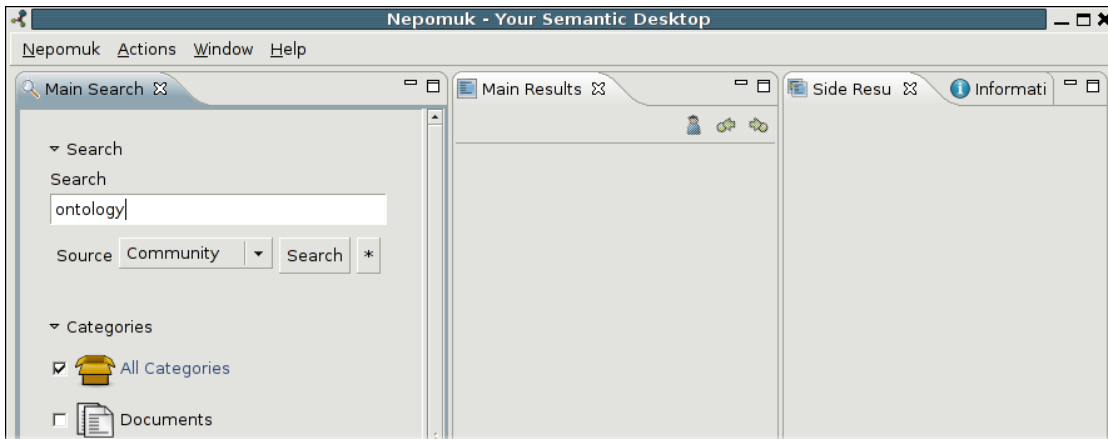


Figure 7.7: The *search* perspective of PSEW.

the members in the P2P network.

We will accompany *Claudia*⁹ during her search for users which are interested in *ontologies*. When preparing a trip to a project meeting in Belfast, Claudia inspects the meeting agenda, and finds an interesting talk on ‘ontology engineering’. She wants to invite other team members which could be interested in this talk. Having found a community of Nepomuk users relevant to that topic, Claudia decides to make the community explicit by creating a group of users. This group allows her to share some resources regarding the talk among the members.

After having started Nepomuk PSEW, Claudia opens the search perspective by clicking in the *Window* menu the *Perspectives* and there the *Search* entry. The search perspective opens and Claudia sees a window like in Figure 7.7. Here she can search for documents, contacts, communities, etc. and thereby access Nepomuk’s corresponding services.

To search for a community, Claudia changes the source for searches by clicking on the dropdown list below the search input box and choosing among the four options ‘Main’, ‘Expert’, ‘Distributed’, and ‘Community’ the last one (‘Community’). Now she enters the term ‘ontology’ into the text input box labeled ‘Search’ as shown in Figure 7.7. By hitting the ‘enter’ key on her keyboard (or clicking with her mouse on the button labeled ‘Search’ below the text input box), she triggers the *CommunityManager* to extract the ‘ontology’ community.

The result of the community detection can be seen in Figure 7.8. A list of users appears in the main result list and allows Claudia to contact the users via e-mail. She can also create a *group* out of the extracted community. Groups allow her to share

⁹Another fictitious person.

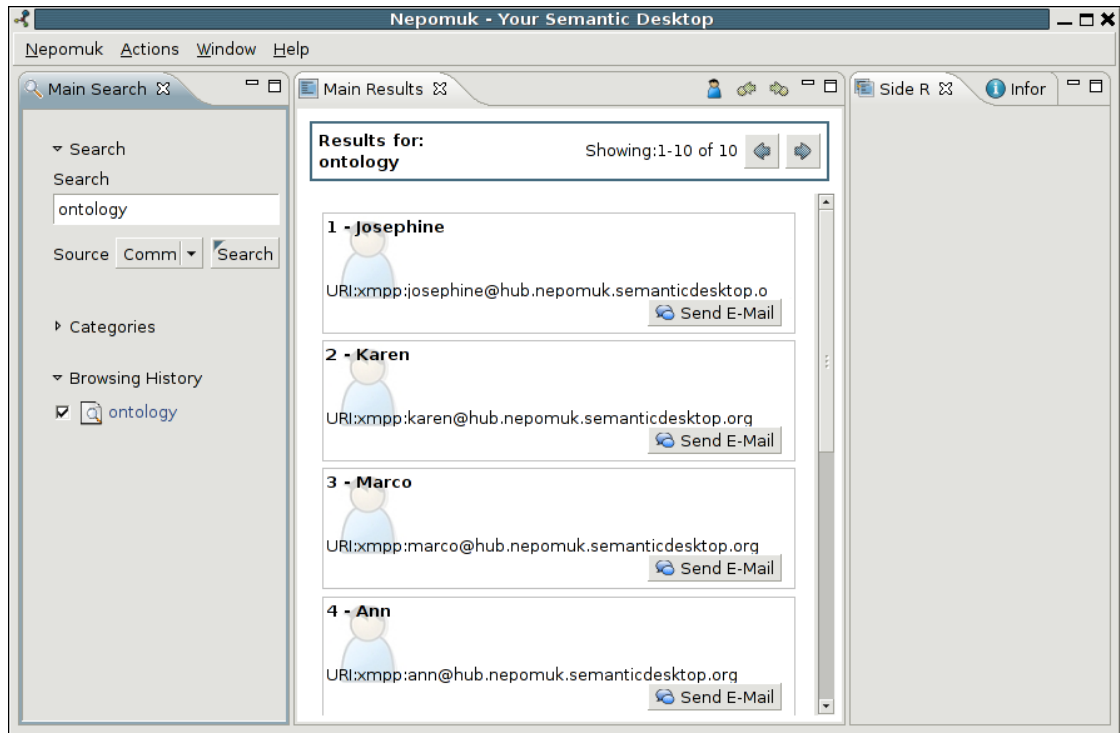


Figure 7.8: The community for the topic ‘ontology’.

resources or documents only among certain people in the network. To create a group, Claudia clicks on the leftmost icon at the top right of the “Main Results” view.¹⁰ Then a window opens entitled “Create group” where Claudia enters “Ontology Engineering Interest Group” as the group name and “ontology engineering” as the topic of the group. After pressing the ‘OK’ button, a small message dialog window pops up acknowledging the successful creation of the new group. Claudia can now use the group to distribute some documents to its members using the Distributed Storage service of Nepomuk.

7.8 Conclusion

The components described in this chapter augment the *social* services of the prototypical social semantic desktop implementation of Nepomuk with tag recommendation and community detection features based on the analysis of the folksonomy structure inherently present in the tagging data of the peer-to-peer network. They allow users to build upon the collected knowledge of the users in the network by using their tags as

¹⁰The icon looks like a small person with a blue shirt.

recommendation and by searching for communities of users.

Changing the perspective from a web based central system as origin of our folksonomy analysis to the peer-to-peer network of interconnected semantic desktops we had to face several challenges. First, the representation of data in RDF format required to construct precise ontologies describing how tag assignments are represented. Naturally, this raised the complexity of querying, exchanging and storing such data. Furthermore, privacy considerations suggested to base our analysis only on tag assignments which the users voluntarily shared across desktops. On the one hand, this released us from collecting the data from several peers, since the metadata sharing component provides access to this shared data. On the other hand, our components now rely on the will and the ability of the users to share their metadata. Finally, the complex interaction of components on one desktop and across desktops was a challenge on its own.

Summing up, we have seen the successful integration of methods to analyze the folksonomy structure into the social semantic desktop. The implemented backend components now paved the way for interesting new end-user applications.

Chapter 8

Logsonomies

As a new way of looking at search engine query logs and the interaction of users with search engines in general, this chapter presents the concept of *logsonomies*. It connects folksonomies with search engines by regarding a user's click on a link as an annotation of that link with the terms of the query. The work in this chapter has been presented in (Jäschke et al., 2008b; Krause et al., 2008a).

8.1 Introduction

Search engines are probably the most important application in the web and one of the key reasons for its success. They index the web and offer a simple user interface to search in this index. Typically, search has been the only way for users to access the index. In contrast, a folksonomy can be explored in different dimensions taking users, tags and resources into account. A further, fundamental difference consists in the way a folksonomy's and a web search engine's index are created: While search engines automatically crawl the web, the content of a folksonomy is determined by its users. As a consequence, the content selection and retrieval in folksonomies is a social process, in which users decide about relevance.

User relevance feedback is integrated into search engine ranking algorithms as well. The feedback is extracted from log files which track a user's click history. However, as the evolvement of social bookmarking systems or recommendation systems on popular websites such as Amazon have shown, web searchers are not only interested in a ranked list of search results, but they like to explore community content as well.

While the previous chapters presented examples of adapting established techniques to folksonomies, we now show parts of our work on an approach in the opposite direction: By analyzing structural similarities between folksonomies and search engine query logs, we bridge the gap between both worlds and make folksonomy analysis techniques applicable to clicklog information of search engines. Thereby we discuss the realization of such 'search communities' within search engines by building an anonymized folksonomy from search engine logdata. As logdata contains queries, clicks and session IDs, the classical dimensions of a folksonomy can be reflected: Queries or query terms represent tags, session IDs correspond to users, and the URLs clicked by users can be considered

as the resources they tagged with the query terms. Search engine users can then browse this data along the well known folksonomy dimensions of tags, users, and resources.

A search engine folksonomy, which we call *logsonomy* in the sequel, brings a variety of features to search engines. As discussed in blogs (e.g., Smith, 2005), one can picture users adding additional tags to their pages to have them higher ranked. Temporal aspects can be introduced by incorporating a fourth dimension and showing popular tags, users or resources at a certain time. Finally, search engine users may interact with each other, commenting and copying search results of each other.

Logsonomies open a wide field of exploration. What kind of semantics can we extract from logsonomies? Is the serendipitous discovery of information also possible in logsonomies? How does the structure of logsonomies differ from folksonomies? In this chapter, we address these questions by analyzing the topological properties of two logsonomy datasets and comparing our findings to a social bookmarking system. In previous work (Cattuto et al., 2007b), it was shown that folksonomies exhibit specific network characteristics (e.g., small world properties, power laws, and long tail degree distributions). These characteristics help to explain why people are fascinated from this structure: A small world leads to short ways between users, resources and tags, which allows for finding interesting resources by browsing the system randomly. High clustering coefficients show dense neighbourhoods which are tracked by the formation of communities around different topics. Finally, co-occurrence graphs show the building of user-enabled shared semantics.

We have introduced and analyzed logsonomies in (Jäschke et al., 2008b; Krause et al., 2008a). The analysis is based on work by Cattuto et al. (2007b), which is also the foundation of the analysis of the tag-tag-co-occurrence graph presented in this chapter. We here focus on the semantics behind the querying and clicking behavior in a logsonomy. For an exploration of the topological structure (i.e., degree distribution, connected components, small world properties), we refer to (Krause et al., 2008a).

8.2 Related Work

In this section we review related work on extracting social information from log data, practical approaches of integrating social features into search engines as well as research on the analysis of social networks our analysis is based on.

A first consideration of the tripartite structure of query logs was presented by Zhang and Dong (2002), where an algorithm to rank resources based on the relationships among users, queries and resources of a search engine's log is proposed. Baeza-Yates and Tiberi (2007) proposed to present query-logs as an implicit folksonomy where queries can be seen as tags associated to documents clicked by people making those queries. The authors extracted semantic relations between queries from a query-click bipartite graph where nodes are queries and an edge between nodes exists when at least one equal

URL has been clicked after submitting the queries. Our work differs in the underlying formal folksonomy model. Furthermore, we study various topological characteristics of the tripartite graph of user, resource and query nodes, while Baeza-Yates and Tiberi only focus on a bipartite graph. Our tag-tag-co-occurrence analysis is related to their graph analysis, but we consider a strength analysis, while they extract semantic relations between queries. Overall, to the best of our knowledge, a comparison between a real folksonomy dataset and logsonomy datasets as presented in this chapter was not carried out before.

Several popular search engines have integrated social services. This includes social bookmarking services where users can explicitly assign bookmarks to share them with other search engine users.¹ Furthermore, individual search history information is provided: Users can browse through clicked pages of the past, view their top searches and most frequently visited pages.² Comprehensive statistics about the overall search activities are provided by tools such as Google Trends,³ Yahoo! Buzz,⁴ or Ask IQ.⁵ Most of the statistical information is derived from query logs. To the best of our knowledge, these query logs have not been transformed to a folksonomy alike search and navigation experience before. Search engine providers do not detail to which extent click data is used to improve search, but none is currently providing a folksonomy-style navigation of query logs. A major reason can be seen in privacy considerations which would need to be addressed carefully (Adar, 2007).

Social network topology features. The graph-theoretic notions our analysis is based on are defined in (Watts and Strogatz, 1998; Dorogovtsev and Mendes, 2003). Cattuto et al. (2007b) extend the small world characteristics to folksonomies. The analysis of structural properties of social networks has been addressed by a number of studies. For example, in (Ahn et al., 2007), topological characteristics of social networking services are described taking degree distribution, clustering properties, degree correlation and evolution over time into consideration. The structure of internal cooperate blogs is analyzed by Kolari et al. (2007) to improve information retrieval and expert finding in companies.

Key studies along the structure and dynamics of social tagging systems are (Cattuto et al., 2007a; Halpin et al., 2006; Marlow et al., 2006; Mika, 2005). Major findings include the power law distribution of tags, the evolution of a vocabulary growth over time and the small world properties of the underlying graph. While the representation of clickdata in form of a folksonomy has not been realized before, clickdata was represented as a bipartite graph using queries and URLs as nodes by Beeferman and Berger (2000) and Xue et al. (2004). An analysis of the bipartite clickdata graph was conducted by

¹<http://www.google.com/s2/sharing/stuff>

²<http://www.google.com/history>

³<http://www.google.com/trends>

⁴<http://buzz.yahoo.com/>

⁵<http://sp.ask.com/en/docs/iq/iq.shtml>

Shi (2007) on the AOL data set which we consider also in our study. An analysis of clickdata as tripartite hypergraph as well as a comparison to folksonomy properties has not been carried out so far.

8.3 Search Engine Query Logs as Logsonomies

Let us consider the query log of a search engine. To map it to the three dimensions of a folksonomy $\mathbb{F} := (U, T, R, Y)$, we set

- U to be the set of *users* of the search engine. Depending on how users in logs are tracked, a user is represented either by an anonymized user ID, or by a session ID.
- T to be the set of *queries* the users gave to the search engine (where one query either results in one tag, or will be split at whitespaces into several tags).
- R to be the set of *URLs* which have been clicked by the search engine users.

In a logsonomy, we assume an association between $t \in T$, $u \in U$ and $r \in R$ when a user u clicked on a resource r of a result set after having submitted a query t (eventually with other terms). The resulting relation $Y \subseteq U \times T \times R$ corresponds to the tag assignments in a folksonomy.

We call the resulting structure a *logsonomy*, since it resembles the formal model of a folksonomy described in Section 2.3. This is motivated by the observation that the process of creating a logsonomy shows similarities to that of a folksonomy. The user describes an information need in terms of a query. He or she then restricts the result set of the search engine by clicking on those URLs whose snippets indicate that the website has some relation to the query. These querying and clicking combinations result in the logsonomy.

However, logsonomies differ from folksonomies in some important points which may affect the resulting structure of the graph:

- Users experience a bias towards clicking top results in a result list. In query log analysis these clicks are usually discounted. To construct a logsonomy, this bias may be integrated by introducing weights for the hyperedges.
- While tagging a specific resource can be seen as an indicator for relevance, users may click on a resource to check if the result is important and then decide that it is not important. However, the act of clicking already indicates an association between query and resource in our case.
- A user might click on a link of a query result list because it is interesting to him for other reasons than the query.

- A user may click on a resource several times in response to the same query when repeating search after some time. This information is lost when constructing the logsonomy, since tag assignments are not weighted.
- When a resource never comes up for search, it cannot be tagged as such.
- Session IDs (in the case of our MSN dataset) differ from a typical user. They are probably more coherent. We have analyzed the differences between users and sessions in (Krause et al., 2008a).

The described differences may lead to a different underlying topological structure regardless of the similar nature of the overall process. We here focus on a comparison of the major properties of the underlying graph and do not specifically investigate the influence of the discussed differences on this results. Considering these differences to get a better understanding of querying and tagging dynamics is an interesting task for future work.

8.4 Datasets

We use three datasets in our study: two click datasets obtained from commercial search engines (MSN and AOL), and one dataset from the social bookmarking system Delicious. The sizes of the different datasets are presented in Table 8.1.

MSN. This dataset consists of about 15 million queries submitted in 7,470,915 different sessions which were tracked from the MSN search engine users in the United States in May 2006. The dataset was provided as part of the “Microsoft Live Labs: Accelerating Search in Academic Research” award in 2006.⁶

We transformed the data to obtain two logsonomy datasets. In the first, the set of tags is the set of complete queries, the set of users is the set of sessions and the set of resources is the set of clicked URLs. Thus, a click on a URL r after submitting the query q within a session s results in the triple (s, q, r) of Y . To make the dataset comparable to the AOL dataset, we reduced the URLs to host only URLs, i. e., we removed the path of each URL leaving only the host name. In the following, we refer to this dataset as *MSN complete queries*. For the second dataset, we also considered host only URLs but additionally, we decomposed each query q at whitespace characters into single terms (q_1, \dots, q_k) and collected the triples (s, q_i, r) (for $i \in \{1, \dots, k\}$) in Y instead of (s, q, r) . This splitting shall better resemble the tags added to resources in folksonomies which typically are single words. As we removed stopwords,⁷ a minor fraction of users (1,375) and URLs

⁶http://research.microsoft.com/ur/us/fundingopps/RFPs/Search_2006_RFP.aspx

⁷We were using an English stopword list from Cornell University (Ithaca, USA) available at <ftp://ftp.cs.cornell.edu/pub/smart/english.stop>.

Table 8.1: Folksonomy and logsonomy datasets. The cause for the lower number of users and resources in the ‘split queries’ logsonomy dataset are queries which consisted of stopwords only.

dataset	$ T $	$ U $	$ R $	$ Y $
Delicious host only URLs	430,526	81,992	934,575	14,730,683
Delicious complete URLs	430,526	81,992	2,913,354	16,217,222
AOL complete queries	4,811,436	519,250	1,620,034	14,427,759
AOL split queries	1,074,640	519,203	1,619,871	34,500,590
MSN complete queries	3,545,310	5,680,615	1,861,010	10,880,140
MSN split queries	902,210	5,679,240	1,860,728	24,204,125

(282) disappeared because of their relation to a query consisting only of stopwords. The second dataset is called *MSN split queries* in the sequel.

AOL. This is a snapshot of queries to the AOL search engine from March, 1st to May, 31st 2006. The dataset consists of 657,426 unique user IDs, 10,154,742 unique queries, and 19,442,629 click-through events (Pass et al., 2006). Analogously to the MSN dataset, we transformed the data into two different datasets (called *AOL complete queries* and *AOL split queries* resp.).⁸

Delicious. To compare the logsonomy structure to a folksonomy, we also used a social bookmarking dataset from Delicious containing posts from 81,992 users up to July, 31st 2005. Again, we have two datasets: one consisting of full URLs to be comparable to prior work on folksonomies (Cattuto et al., 2007b), and one reduced to the host part of the URL only to be comparable to the logsonomy datasets.

8.5 Strength in the Tag-Tag-Co-Occurrence Graph

In this section we analyze the semantics behind the querying and clicking behavior in a logsonomy. Therefore we study the properties of the *tag-tag-co-occurrence graph*, as it mainly reflects the semantics describing the clicked URLs with respect to the queries. This graph consists of tags which are linked by an edge if they occur in the same post.⁹ More formally, $G := (T, E)$ with $E := \{(t_1, t_2) \mid \exists u \in U, \exists r \in R: (u, t_1, r) \in Y \wedge (u, t_2, r) \in Y\}$ defines the tag-tag-co-occurrence graph on the set T of tags. Naturally, we can add weights to the edges by counting in how many posts two tags appear together. We define the *weight* $w(t_1, t_2)$ of an edge (t_1, t_2) to be $w(t_1, t_2) := |\{(u, r) \in U \times R \mid$

⁸We used unique user IDs, because session IDs were not included in the AOL dataset.

⁹In our logsonomy setting, a post comprises all queries in which a resource has been clicked.

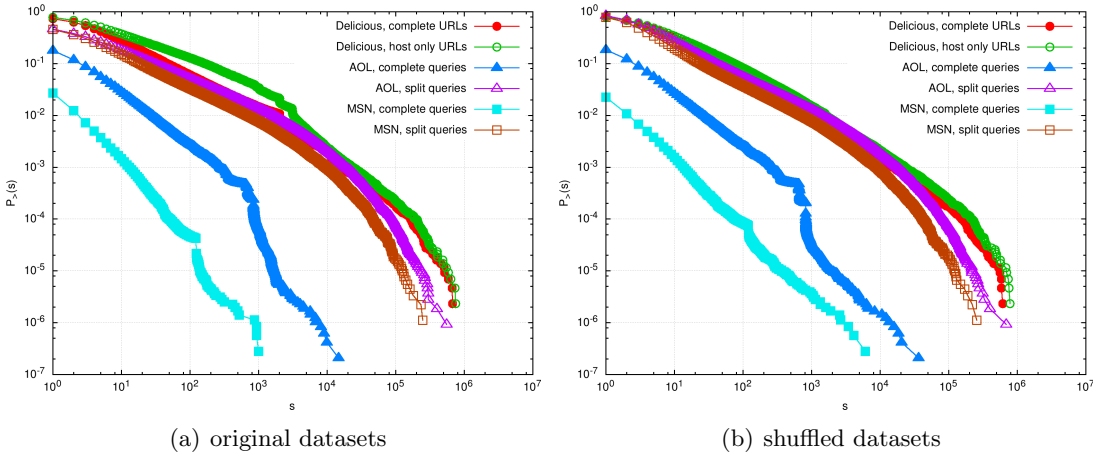


Figure 8.1: The cumulative strength distribution for the network of co-occurrence of tags and queries for all datasets. Split query logsonomies show a very similar distribution to the Delicious folksonomy.

$(u, t_1, r) \in Y \wedge (u, t_2, r) \in Y\}$. The *strength* s_t of a tag t in the graph is then defined as

$$s_t := \sum_{t' \neq t} w(t, t'). \quad (8.1)$$

Each of the following figures contains data from two types of datasets: the *raw* data of the datasets as described in Section 8.4, and a *shuffled* version of the datasets where we shuffled the tags in the triples of the relation Y . To do so, for each triple $(u, t, r) \in Y$ we randomly picked a tag $t' \in T$ and exchanged (u, t, r) in Y with (u, t', r) . We picked each tag with a probability according to its degree, such that the tag degree distribution of the resulting folksonomy is identical to the original one.

8.5.1 Cumulative Strength Distribution

One of the standard measures of complex network theory is the *cumulative strength distribution* $P_>(s)$ (Cattuto et al., 2007b). It specifies for a given node strength the probability that a node will exceed this strength. For the Delicious dataset we observe the same fat tailed distribution as in (Cattuto et al., 2007b) (cf. Figure 8.1). The logsonomy with split queries for AOL as well as for MSN shows a very similar distribution to the Delicious folksonomy. This distribution is also not disturbed by the shuffling process on the tags – which confirms that the strength distribution for both the logsonomy and folksonomy data only depends on the tag frequencies and not on their semantics – which is destroyed by the shuffling process.

We observe a different behavior for the datasets with complete queries. Queries with high strengths (above 10^2 for MSN and above 10^3 for AOL) show up less frequently than expected: these frequencies are significantly below those obtained for the shuffled versions. At least for the MSN data, this can be explained by the construction process of the dataset: The probability that a user clicks on the same URL within one session but based on another query is very unlikely. The number of queries that are connected to many other queries by some (user, resource) pairs is therefore below expectation.

8.5.2 Average Nearest-Neighbor Strength

Next, we want to take a closer look into the co-occurrence network of the logsonomy to see whether another property holds or not. Therefore, we measure the *average nearest-neighbor strength* of tags in this graph. For that purpose we define the neighborhood N_t of a tag t to be $N_t := \{t' \mid (t, t') \in E\}$. The average nearest-neighbor strength is then defined as:

$$S_{nn}(t) = \frac{1}{|N_t|} \sum_{t' \in N_t} s_{t'}. \quad (8.2)$$

For each tag $t \in T$, we will set its average nearest-neighbor strength $S_{nn}(t)$ in relation to its own strength s_t . This relation can reveal the difference between human-produced social networks and technological artefacts (Newman, 2002): a positive correlation – called *assortative mixing* – hints at social networks while a negative correlation frequently shows up in technological and biological networks.

Each of the following six figures (8.2(a) to 8.4(b)) shows the strength of each tag versus its average strength for the raw dataset and its corresponding shuffled version. Since the points in the plot can overlap, each point represents at least one tag. Therefore, the figures additionally contain linear least squares fits for each dataset. Those are splitted into two regions: tags with low strength ($s_t < 10^3$) and tags with high strength ($s_t > 10^3$).

Figure 8.2 shows the relation between s_t and $S_{nn}(t)$ for the two Delicious datasets in consideration. The plot for the dataset with complete URLs (8.2(a)) shows that the average strength of the neighbors of tags with low strength varies strongly while for tags with higher strength the variation is much smaller, as already observed by Cattuto et al. (2007b). For tags with high values of s_t , their average nearest-neighbor strength and s_t itself are slightly anti-correlated. Clusters in the diagram (regions of points separated from the main point cloud, like the one for $10^3 < s_t < 10^4, 10^3 < S_{nn}(t) < 10^4$) are mainly caused by artifacts in the data (such as spam).

The shuffled data shows a more regular distribution of the average nearest-neighbor strength over s_t and a larger anti-correlation for higher values of s_t . Since shuffling destroys semantics inherent in the original network, the obvious difference to the raw data, especially in the low strength regions, is a strong indicator that the infrequent

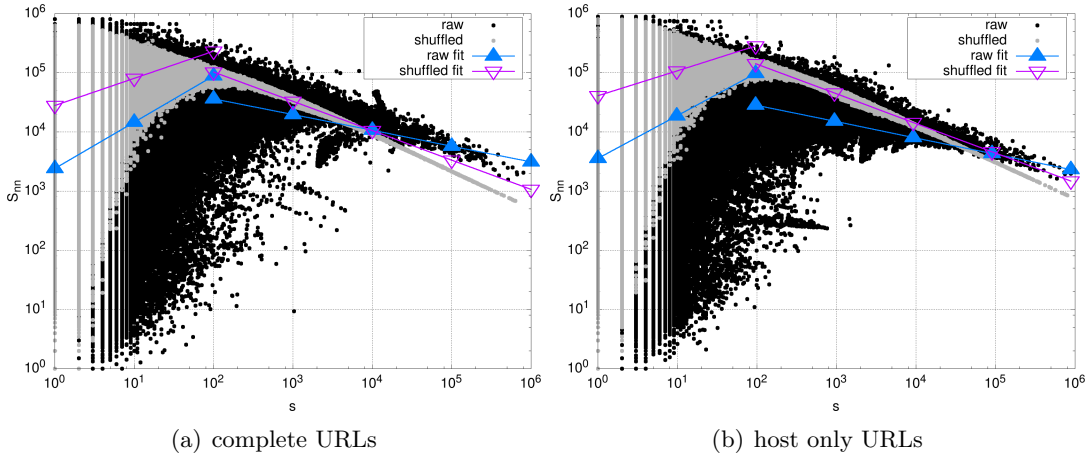


Figure 8.2: Average nearest-neighbor strength S_{nn} of tags in relation to the tag strengths in Delicious. The distribution of both datasets is very similar: the average nearest-neighbor strength for tags with low strength varies strongly, while for tags with higher strength the variation is much smaller.

tags are frequently grouped together by their inherent semantics – an effect which is destroyed by shuffling (Cattuto et al., 2007b). Removing the paths from the URLs of the Delicious dataset does not change the picture much: only some clusters (dis)appear, as Figure 8.2(b) shows.

The strength distributions of the split versions of the AOL and MSN datasets (Figures 8.3(a) and 8.4(a)) show noticeable similarity to the behaviour in Delicious for both the original and the shuffled data. This supports the hypothesis that the semantics of the single words within web search engine queries provide topically organized local structures on the tag-tag-co-occurrence graph similar to the behavior in a folksonomy.

The strength distributions for the complete queries of AOL and MSN (Figures 8.3(b) and 8.4(b)), on the other hand, differ substantially from the distributions of the Delicious data. Not only are the strengths and average nearest-neighbor strengths smaller than in Delicious (which is in line with the results for the cumulative strength distribution in Figure 8.1), but also the shape is different: it is more strongly bulged on its lower part, which results from a large number of queries with medium to high strength (around 10^2 in AOL and $10^{1.5}$ in MSN) that are connected in average to less strong queries. We assume that this structure stems from frequency effects rather than from semantically induced structures, as now the shuffled data differs only slightly from the raw data.

In the distribution for the complete AOL queries, we additionally observe – both for the raw and the shuffled data – a separated cluster on top of the distribution. We currently lack an explanation for this phenomenon.

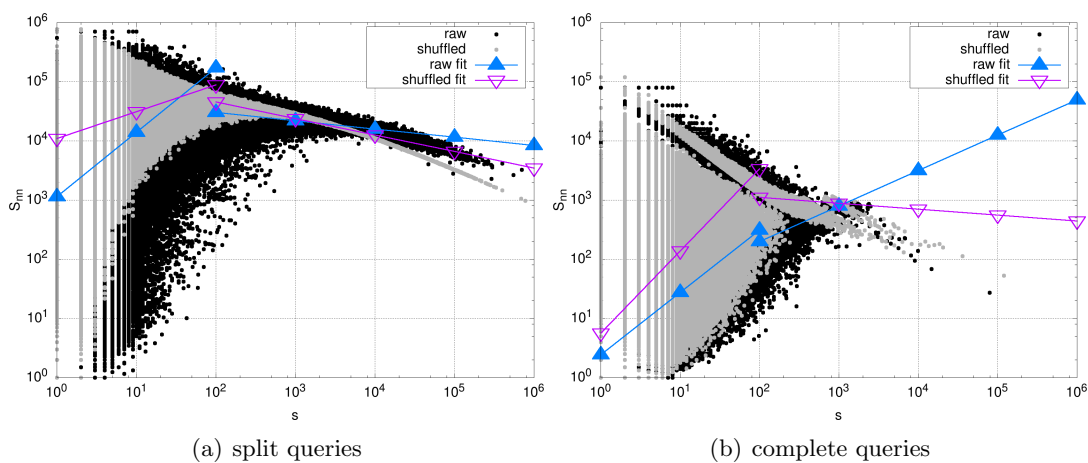


Figure 8.3: Average nearest-neighbor strength S_{nn} of tags in relation to the tag strengths in AOL. The datasets with split queries show a similar assortative and dissortative behaviour for the original and shuffled datasets. The full query dataset differs in size and shape.

We summarize the results of the analysis of the tag-tag-co-occurrence graph with the conclusion that the logsonomies based on split queries are closer in terms of semantical behavior to folksonomies than the logsonomies based on complete queries.

8.6 Outlook

In this chapter we presented the idea of transforming a search engine query log into a ‘logsonomy’. We analyzed the resulting graph structure to find similarities and dissimilarities to the existing folksonomy Delicious. The analysis of the strength in the tag-tag-co-occurrence network revealed very similar properties between folksonomies and logsonomies with split queries.

Overall, the results support our vision to merge the search engine and folksonomy worlds into one system. While some search engines already allow to store and browse search results, they do not provide folksonomy-alike navigation or the possibility to add or change tags. From a practical point of view, the following considerations are further arguments for a logsonomy implementation and its combination with a folksonomy system:

- Users could enrich visited URLs with their own tags (besides the automatically added words from the query) and the search engine could use these tags to consider such URLs for later queries – also from other users. Thus, those tags could improve the quality of the search engine.

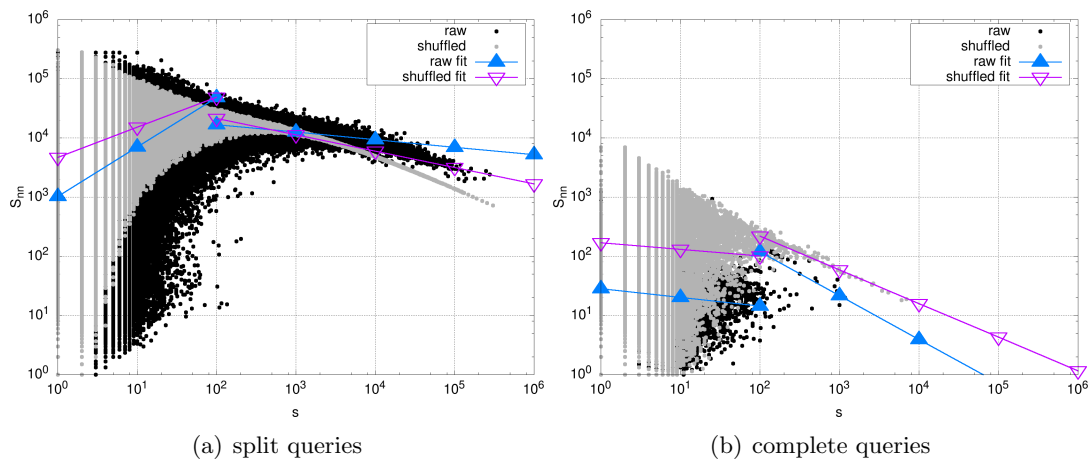


Figure 8.4: Average nearest-neighbor strength S_{nn} of tags in relation to the tag strengths in MSN. The datasets with split queries show a similar assortative and disassortative behaviour for the original and shuffled datasets. The full query dataset differs in size and shape.

- The popularity of folksonomy systems could increase the customer loyalty for a search engine. The community-feeling known from folksonomies could pass over to search engines.
- Search engines typically have the problem of finding new, unlinked web pages. Assumed, users store new pages in the folksonomy, the search engine could direct its crawlers better to new pages. Additionally, those URLs would have been already annotated by the user's tags – even without crawling the pages it would be possible to present them in result sets.
- As described in (Röttgers, 2007) and (Hotho et al., 2006c), folksonomies can assist in finding trends in society. Many social bookmarking users can be viewed as trend setters or early adopters of innovative ideas – their data is valuable for improving a search engine's topicality.
- Bookmarked URLs of the user may include pages, the search engine can not reach (intranet, password-protected pages, etc.). These pages can then be integrated into personalized search results.

However, privacy issues are very important when talking about search engine logs. They provide details of a user's life and often allow to identify the user himself (Adar, 2007). Certainly, this issue needs attention when implementing a logsonomy system.

As a next step we planned to analyze the conceptual structure of logsonomies by applying the TRIAS algorithm presented in Chapter 4. However, the diversity of the datasets caused several artifacts: (i) On the datasets with split queries, TRIAS mainly re-builds the queries, i. e., the tags of the larger tri-concepts were the terms of a query; (ii) for a proper application of TRIAS, one would need not sessions but real user histories; (iii) host-only URLs in principle provide larger concepts, however, they also merge queries which do not belong together, e. g., on hosts with a large variety of topics like flickr.com.

Chapter 9

Outlook

In this thesis we have focused on BibSonomy and two topics centered around collaborative tagging systems: tag recommendations and Formal Concept Analysis. With an outlook on current trends and emerging developments in those two areas as well as in collaborative tagging in general, this chapter presents compelling impulses for possible future work.

9.1 Collaborative Tagging Systems

Although collaborative tagging systems now exist since about five years, they still develop in an amazing pace. With still new systems appearing, motivating more and more users to annotate their resources, it is difficult to oversee the future of collaborative tagging. Therefore, we briefly discuss new paradigms and challenges, and dare an outlook into the future of BibSonomy.

9.1.1 New Paradigms and Challenges

The trend towards *ubiquitous computing*, where interconnected mobile devices provide web access at any place and time, sensor networks produce vast amounts of data, and billions of RFID tags uniquely identify products, items, and living beings, certainly will influence the next generation of collaborative tagging systems. They must cope with an ever increasing amount and variety of data that needs to be accessible in an easy way for more and more users. Which communication paradigm – client-server as it is the state of the art in collaborative tagging, P2P, or something else – will allow the systems to handle the omnipresence of such devices with the mentioned challenges is not clear, yet.

The growing dissemination of *Semantic Web* technologies (like RDF) and ideas (like identification of resources and linking to ontologies) has already influenced collaborative tagging systems. Users of BibSonomy, for instance, can access their publication posts in RDF format (cf. Section 3.5.2) as well as their user profile (using the FOAF ontology). With ontologies like SCOT¹ the systems can represent their tagging information in a machine-readable way and thereby become a part of the Semantic Web (see (Kim et al.,

¹<http://scot-project.org/>

2008) for a comparison of ontologies for folksonomies). On the other hand, collaborative tagging systems might have further paved the way for the Semantic Web by showing millions of users how important annotation of resources is and how simply it can be achieved. To benefit from this development, the Semantic Web community must now provide tools which allow users to manage their knowledge in a similar easy way using Semantic Web technology.

Another challenge is the growing availability of *geo-annotated resources* (Goodchild, 2007) that allows us to add – besides time – another dimension to folksonomies. With state-of-the-art smartphones, for example, users can already take high-resolution pictures which are then automatically geo-annotated and uploaded to a social photo sharing platform like Flickr where users can then add tags. Thereby, people all over the world can immediately find and see the photos on a map.² As Rattenbury et al. (2007) have shown, such geo-annotations can also provide insights into the semantics of tags. It enables them to relate tags to places and thereby extract ‘place tags’, i. e., tags that describe places like *San Francisco*, or *Logan Airport*. This can help to develop search and ranking methods which take the location of the user into account and also be employed for ontology learning (Buitelaar et al., 2005).

9.1.2 BibSonomy

As already mentioned in Section 3.6, the PUMA project will enhance BibSonomy’s support in the area of academic publication management. Amongst other things, this facilitates the interaction with library catalogues such that users can tag books they have lent, comment and rate them. Furthermore, researchers then can manage their scientific publications, do reporting, generate publication lists for their homepage, and provide input for evaluation.

Besides libraries, other institutions like companies, research projects, and research centers have also expressed interest in BibSonomy. One of their requirements typically is having the server under their control – for reasons like security and privacy, stability and accessibility, as well as long-term availability. This means that the number of servers running BibSonomy will grow. Though those institutions often require a non-public instance, synchronization of several instances, e. g., between libraries, is an important future challenge. It is also a requirement for a distributed load balancing setup where not only read access is distributed over several web application instances and database systems (as it is done already now), but also write access.

Another challenge is the improved interaction of BibSonomy with different programs and systems. As this is one of the goals of the Semantic Web, improved support for its formats and protocols could open BibSonomy into that direction. For example, the generation of proper URIs for users, tags, resources, authors, and documents together

²<http://www.flickr.com/map/>

with an RDF support for the REST API would allow other Semantic Web based services to more easily interact with BibSonomy.

9.2 Tag Recommendations

Since the emergence of collaborative tagging systems, research in the field of tag recommendations has established evaluation protocols, defined baselines, and came up with quite some valuable methods. In this section we present a selection of challenges which could be promising entry points for the next generation of tag recommendation algorithms.

9.2.1 Different Types of Tags

Most of the approaches that tackle the problem of tag recommendations in folksonomies are rather generic (e. g., based on Collaborative Filtering, co-occurrence counts, or the content of resources) and don't distinguish different types of tags. However, Golder and Huberman (2005) identified seven kinds of tags, or better functions, tags can perform for a resource: identifying what (or who) it is about, identifying what it is, identifying who owns it, refining categories, identifying qualities or characteristics (*scary*, *funny*, *stupid*, *inspirational*), self reference (*myown*), and task organizing (*toread*, *thesis*). Naturally, a first step to further improve tag recommendations would require methods to classify tags into one of those categories. Steps into this direction have been done by Strohmaier (2008) who tries to identify *purpose tags*, i. e., tags that describe the intent rather than the content. Once different categories can be distinguished, one could then focus the recommender on tags from categories where it is easier to 'guess' the tags for a resource. E. g., it might be rather difficult to recommend tags which describe the organization of tasks like *toread*, since it requires to know why the user is tagging the resource at hand.

9.2.2 Personalization

An important challenge is the personalization of recommendations by respecting the different tagging habits of users. A tag, for example, can be in different categories, depending on the user's understanding. Since tagging allows the user to capture his view on a certain resource (besides merely describing its content), methods to predict the opinion of a user regarding a resource could lead to more personalized recommendations. Therefore, methods are necessary which identify tags expressing opinions about items. Most existing approaches try to identify the opinion of users based on written texts.³ Breck et al. (2007), for example, suggest a method to identify words and phrases in texts which express opinions. Recommending such 'opinion tags' without influencing the user

³For a survey on opinion mining and sentiment analysis see (Pang and Lee, 2008).

by pushing his opinion into a certain direction is a challenging task – it has been shown by Cosley et al. (2003) that recommender systems affect the user’s opinion.

Another option could be the integration of semantic approaches to capture the user’s personal information model (see (Sauer mann, 2003)). This could improve personalization, since tags could then explicitly be assigned to a category or distinguished from other tags by the user.

In general, however, too personalized tag recommendations hamper the goal of consolidating the tag vocabulary across users. A recommender which perfectly adapts to the language and thinking of the user does not give him the chance to see other users’ vocabulary and thereby adopt popular tags or follow accepted tagging habits. Finding the balance between recommending what the user wants and what is good for him, the system, or other users is a difficult task and a widely discussed topic in the recommender community.

9.2.3 Trust

A relatively new topic in the recommender community is the incorporation of *trust between users* into the systems. This can be seen as a special way of personalization. Although it is very difficult to define trust (personal background, history of interaction, etc. play a role), in the context of recommender systems trust usually describes the similarity of users in their opinion about a topic. It is therefore important to take into account the context based on which trust has been computed – someone a user would trust in recommending a movie might not be trustworthy in providing a recommendation for a certain product. Recommendation methods like Collaborative Filtering, which incorporate similarity between users, can use trust instead of or in combination with similarities. Trust can also be used to filter or sort information.

Up to now, there is no attempt known to leverage trust for tag recommendations – some typical applications are in the movie domain (see the survey by Golbeck (2006)). One reason for that might be the lack of appropriate data to compute trust in folksonomies (like ratings). Although most systems provide some kind of social networking features (groups, friends), this data often is not easily available to the research community and it is also questionable if it is a useful basis to compute trust in tagging. Both friends and group members might have an interest in the same topics the user has, however, their tagging behaviour or used vocabulary might be quite different.

Another category of trust is the *user’s trust in the recommendation system* or *im-personal trust* (O’Donovan and Smyth, 2005). It describes to what respect the user trusts the system to give recommendations which are helpful to him. One approach to strengthen the user’s trust is the *explanation* of recommendations (Herlocker et al., 2000) which makes recommendations more traceable – a well-known example are Amazon’s “Customers Who Bought This Item Also Bought” recommendations. Vig et al. (2009) find that tags can be used to explain movie recommendations and thereby help the

user to understand why an item was recommended and to decide if they like it. Although up to now there is no research known on explaining tag recommendations, depending on the used algorithm existing approaches can be adopted, e.g., (Herlocker et al., 2000) for Collaborative Filtering. One hindrance in building up trust is spam, since it can influence recommendation systems and degrade the quality of recommendations (Lam and Riedl, 2004).

9.2.4 Further Aspects

As the results of the 2009 ECML PKDD Discovery Challenge (Eisterlehner et al., 2009) have shown, clever combinations of simple methods yield tag recommendation results which can outperform state-of-the-art machine learning approaches. If more sophisticated methods will gain performance is thus at least questionable, in particular since they are often not able to solve the cold-start problem, i. e., to deliver recommendations for unknown users or resources.

Systems like BibSonomy, which allow users to maintain a relation between tags (cf. Section 3.5.5) suggest that more complex forms of recommendations might be necessary. I. e., instead of recommending just tags, one could also provide more structure by recommending elements for the user's tag relation. Beyond tagging, one could incorporate ontology learning techniques (Buitelaar et al., 2005) and discuss the recommendation of concepts, general relations or even (parts of) ontologies. This has been partly addressed by (Haase et al., 2005), who use a Collaborative Filtering approach to suggest personalized ontology changes.

9.3 Formal Concept Analysis

We here pick up three topics we have already touched in Chapter 4: the visualization of (tri-)lattices, the handling of large datasets, and triadic association rules.

9.3.1 Visualization

Still a challenging task for both dyadic and triadic FCA, in particular for larger contexts, is the visualization of (tri-)lattices. Although several algorithms exist for drawing lattices (Battista et al., 1994), typically the graphs become unreadable for concept lattices with several hundred or more concepts. Furthermore, it is questionable anyway, if standard Hasse diagrams are a good visualization for such large lattices.

For the triadic case, the situation is even worse, with no existing methods for automatically drawing tri-lattices. The simplification into *neighborhoods* as presented in Section 4.5 paves the way for using standard graph-drawing algorithms. However, methods which allow us to visualize the complete tri-lattice (as Hasse diagrams do for lattices)

are more desirable. One interesting approach could be a variant of nested line diagrams for tri-lattices, like in dyadic FCA (Ganter and Wille, 1999).

9.3.2 Large Datasets

As discussed in Section 4.3.3, the number of (frequent) concepts may grow exponentially with the size of the context, and thus, for large datasets, computation is very time-consuming.

Parallelization of algorithms solves this problem up to a certain scale and is nowadays a common technique – with the availability of multi-core processors and multi-threading programming languages. In Section 4.6 we have suggested how TRIAS, when using NEXT CLOSURE as underlying algorithm to compute binary concepts, can be parallelized in a natural way.

Another option is the aggregation of data before applying FCA, e. g., by clustering and then using the clusters as objects and their features as attributes. Besides lowering the complexity this can also help to reduce noise. This idea has been applied by Hotho (2004, Sec. 8.5.3) to cluster text documents using the k -means algorithm. For the triadic case, multi-way clustering methods which respect all three dimensions – like the approach presented by Bekkerman et al. (2005) – need to be adopted.

9.3.3 Triadic Association Rules

As discussed in Section 4.2.4, frequent itemsets are typically employed to compute association rules on binary contexts. Such rules have the form $A \Rightarrow B$ with A and B being sets of attributes and expressing that objects that have all attributes from A (with a certain confidence) also have all attributes from B . For example, by analyzing supermarket transaction data, association rules allow the marketing staff to find products which are often bought together by customers. The situation is different for triadic formal contexts, where the notion of association rules has not been defined, yet.

A first step towards truly ‘triadic association rules’ has been done by Ganter and Obiedkov (2004), who discuss *implications* in triadic contexts. Implications are a special form of association rules where the right-hand-side holds for all objects (i. e., with confidence 1). They refer to (Biedermann, 1998a), whose rules allow to express facts like *If a resource has been tagged with all tags from $T_1 \subseteq T$ by all users from $V \subseteq U$, then it also has been tagged with all tags from $T_2 \subseteq T$ by all users from V .*⁴ Ganter and Obiedkov further consider two classes of implications they call ‘conditional attribute implication’⁵

⁴More formally, for a triadic formal context (G, M, B, Y) , Biedermann’s rules of the form $(R \rightarrow S)_C$ are interpreted as “If an object has all attributes from R under all conditions from C , then it also has all attributes from S under all conditions from C ” (with $R, S \subseteq M$ and $C \subseteq B$).

⁵ $R \xrightarrow{C} S$ which is interpreted as “ R implies S under all conditions from C ”

and ‘attribute×condition implication’.⁶ As a next step, one must think of equivalent notions of *support* and *confidence* for the triadic case to provide appropriate relevance measures for triadic association rules.

9.4 Conclusion

At the time of writing, it seems that the popularity of collaborative tagging systems is still increasing and there is almost no new Web 2.0 system coming out without tagging facilities included. Thus, there is a growing playground that can be potentially used to test and integrate the methods presented in this thesis. The topics that have been addressed in this last chapter provide in the author’s opinion a good starting point to continue the work of this thesis. Yet, there are many more facets of collaborative tagging systems which could be explored and improved, like the visualization of tag clouds, the interaction and integration of the various systems, or the effects of time on many aspects of such systems.

⁶ $R \rightarrow S$ (with R and S being subsets of $M \times B$)

Bibliography

- E. Adar. User 4xxxxx9: Anonymizing query logs. In *Workshop on Query Log Analysis at the 16th World Wide Web Conference*, May 2007.
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94)*, pages 478–499. Morgan Kaufmann, Sept. 1994.
- R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE Computer Society Press, Mar. 1995.
- R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, New York, NY, USA, May 1993. ACM. ISBN 0-89791-592-5. doi: 10.1145/170035.170072.
- Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong. Analysis of topological characteristics of huge online social networking services. In *Proceedings of the 16th International Conference on World Wide Web*, pages 835–844, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242685.
- A. Arnauld and P. Nicole. *La logique ou l'art de penser — contenant, outre les règles communes, plusieurs observations nouvelles, propres à former le jugement*. Ch. Saveux, 1668.
- R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *KDD '07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 76–85, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-609-7. doi: 10.1145/1281192.1281204.
- R. A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. ISBN 020139829X.
- L. Balby Marinho and L. Schmidt-Thieme. Collaborative tag recommendations. In *Data Analysis, Machine Learning and Applications*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 533–540, Berlin/Heidelberg, 2007. Springer. ISBN 978-3-540-78239-1. doi: 10.1007/978-3-540-78246-9_63.

- A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: authority-based keyword search in databases. In *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases*, pages 564–575. VLDB Endowment, 2004. ISBN 0-12-088469-0.
- P. Basile, D. Gendarmi, F. Lanubile, and G. Semeraro. Recommending smart tags in a social bookmarking system. In *Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*, pages 22–29, 2007. URL <http://www.kde.cs.uni-kassel.de/ws/eswc2007/proc/RecommendingSmartTags.pdf>.
- Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Explor. Newsl.*, 2(2):66–75, 2000. ISSN 1931-0145. doi: 10.1145/380995.381017.
- V. Batagelj and M. Zaversnik. Generalized cores. *CoRR*, cs.DS/0202039, Feb. 2002. URL <http://arxiv.org/abs/cs/0202039>.
- S. Bateman, C. Brooks, G. McCalla, and P. Brusilovsky. Applying collaborative tagging to e-learning. In *Proceedings of the Workshop on Tagging and Metadata for Social Information Organization (WWW'07)*, 2007.
- G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*, 4(5): 235–282, 1994. ISSN 0925-7721. doi: 10.1016/0925-7721(94)00014-X.
- R. J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 85–93, New York, NY, USA, June 1998. ACM. ISBN 0-89791-995-5. doi: 10.1145/276304.276313.
- D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 407–416, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347176.
- R. Bekkerman, R. El-Yaniv, and A. McCallum. Multi-way distributional clustering via pairwise interactions. In *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*, pages 41–48, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102357.
- D. Benz and A. Hotho. Position paper: Ontology learning from folksonomies. In A. Hinneburg, editor, *Workshop Proceedings of Lernen - Wissensentdeckung - Adaptivität (LWA 2007)*, pages 109–112, Halle/Saale, Sept. 2007. Martin-Luther-Universität Halle-Wittenberg. ISBN 978-3-86010-907-6.

- D. Benz, K. Tso, and L. Schmidt-Thieme. Automatic bookmark classification: A collaborative approach. In *Proceedings of the Second Workshop on Innovations in Web Infrastructure (IWI 2006)*, Edinburgh, Scotland, 2006.
- T. Berners-Lee, J. A. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001. ISSN 0036-8733.
- K. Biedermann. Triadic Galois connections. In K. Denecke and O. Lüders, editors, *General algebra and applications in discrete mathematics*, pages 23–33, Aachen, 1997a. Shaker Verlag.
- K. Biedermann. How triadic diagrams represent conceptual structures. In D. Lukose, H. S. Delugach, M. Keeler, L. Searle, and J. F. Sowa, editors, *Conceptual Structures: Fulfilling Peirce's Dream*, volume 1257 of *Lecture Notes in Computer Science*, pages 304–317, Berlin/Heidelberg, 1997b. Springer. ISBN 978-3-540-63308-2. doi: 10.1007/BFb0027865.
- K. Biedermann. *A foundation of the theory of trilattices*. Dissertation, TU Darmstadt, Aachen, 1998a.
- K. Biedermann. Powerset trilattices. In M.-L. Mugnier and M. Chein, editors, *Conceptual Structures: Theory, Tools and Applications*, volume 1453 of *Lecture Notes in Computer Science*, pages 209–224, Berlin/Heidelberg, 1998b. Springer. ISBN 978-3-540-64791-1. doi: 10.1007/BFb0054900.
- S. Bloehdorn, O. Görlitz, S. Schenk, and M. Völkel. TagFS - tag semantics for hierarchical file systems. In *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06)*, Graz, Austria, Sept. 2006.
- M. Bork. Webservice API für Bibsonomy. Project report, Fachgebiet Wissensverarbeitung, Universität Kassel, 2006. URL <http://www.kde.cs.uni-kassel.de/lehre/arbeiten/documents/bork2006webservice.pdf>.
- J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 75–85, Berlin/Heidelberg, 2000. Springer. ISBN 978-3-540-41066-9. doi: 10.1007/3-540-45372-5.
- E. Breck, Y. Choi, and C. Cardie. Identifying expressions of opinion in context. In *IJCAI'07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2683–2688, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

- J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufman, 1998.
- C. H. Brooks and N. Montanez. An analysis of the effectiveness of tagging in blogs. In *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs*. AAAI, Mar. 2005.
- P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*, volume 123 of *Frontiers in Artificial Intelligence*. IOS Press, July 2005.
- R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002. ISSN 0924-1868. doi: 10.1023/A:1021240730564.
- A. Byde, H. Wan, and S. Cayzer. Personalized tag recommendations via tagging and content-based similarity metrics. In *Proceedings of the International Conference on Weblogs and Social Media*, Mar. 2007.
- A. Bykowski and C. Rigotti. A condensed representation to find frequent patterns. In *PODS '01: Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 267–273, New York, NY, USA, 2001. ACM. ISBN 1-58113-361-8. doi: 10.1145/375551.375604.
- T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Principles of Data Mining and Knowledge Discovery*, volume 2431 of *Lecture Notes in Computer Science*, pages 1–42, Berlin/Heidelberg, 2002. Springer. ISBN 978-3-540-44037-6. doi: 10.1007/3-540-45681-3_7.
- H. Cao, M. Xie, L. Xue, C. Liu, F. Teng, and Y. Huang. Social tag prediction base on supervised ranking model. In Eisterlehner et al. (2009), pages 35–48.
- C. Carpineto and G. Romano. GALOIS: An order-theoretic approach to conceptual clustering. In *Machine Learning Proc. ICML 1993*, pages 33–40. Morgan Kaufmann Publications, 1993.
- C. Carpineto and G. Romano. *Concept Data Analysis*. Wiley, 2004.
- C. Cattuto, V. Loreto, and L. Pietronero. Collaborative tagging and semiotic dynamics. *CoRR*, abs/cs/0605015, May 2006. doi: 10.1073/pnas.0610487104. URL <http://arxiv.org/abs/cs/0605015>.

-
- C. Cattuto, A. Baldassarri, V. D. P. Servedio, and V. Loreto. Vocabulary growth in collaborative tagging systems. *CoRR*, abs/0704.3316, Apr. 2007a. URL <http://arxiv.org/abs/0704.3316>.
- C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. Network properties of folksonomies. *AI Communications*, 20(4):245–262, Dec. 2007b. ISSN 0921-7126.
- C. Cattuto, D. Benz, A. Hotho, and G. Stumme. Semantic analysis of tag similarity measures in collaborative tagging systems. In *Proceedings of the 3rd Workshop on Ontology Learning and Population (OLP3)*, pages 39–43, Patras, Greece, July 2008. ISBN 978-960-89282-6-8.
- C. Cattuto, A. Barrat, A. Baldassarri, G. Schehr, and V. Loreto. Collective dynamics of social annotation. *CoRR*, abs/0902.2866, Apr. 2009. URL <http://arxiv.org/abs/0902.2866>.
- P.-A. Chirita, S. Ghita, W. Nejdl, and R. Paiu. Semantically enhanced searching and ranking on the desktop. In S. Decker, J. Park, D. Quan, and L. Sauermann, editors, *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, pages 92 – 106, Galway, Ireland, Nov. 2005.
- P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, Secaucus, NJ, USA, 2006. ISBN 0387306323.
- D. Cosley, S. Lawrence, and D. M. Pennock. REFEREE: an open framework for practical testing of recommender systems using ResearchIndex. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 35–46. VLDB Endowment, 2002.
- D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions. In *CHI '03: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 585–592, New York, NY, USA, 2003. ACM. ISBN 1-58113-630-7. doi: 10.1145/642611.642713.
- V. Darlagiannis, N. Bonvin, S. Y. Reddy, and R. Narendula. Nepomuk Deliverable D4.3 – Distributed Search and Storage System. Technical report, NEPOMUK consortium, Dec. 2008. URL <http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D4-3>.
- F. Dau and R. Wille. On the modal understanding of triadic contexts. In R. Decker and W. Gaul, editors, *Classification and Information Processing at the Turn of the Millenium*, Proc. Gesellschaft für Klassifikation, 2001.

- W. de Nooy, A. Mrvar, and V. Batagelj. *Exploratory Social Network Analysis with Pajek*. Number 27 in Structural Analysis in the Social Sciences. Cambridge University Press, New York, NY, USA, 2005. ISBN 0521602629.
- S. Decker and M. Frank. The social semantic desktop. Technical Report DERI-TR-2004-05-02, Digital Enterprise Research Institute (DERI), Galway, Ireland, May 2004. URL <http://www.deri.ie/fileadmin/documents/DERI-TR-2004-05-02.pdf>.
- G. Demartini, P. Haghani, R. Jäschke, A. Johnston, M. Kiesel, and R. Paiu. Nepomuk Deliverable D5.1 – Community Support Software First Version. Technical report, NEPOMUK consortium, June 2006. URL <http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D5-1>.
- G. Demartini, R. Jäschke, R. Stecher, and P. Haghani. Nepomuk Deliverable D5.2 – Report on Metadata Sharing and Recommendation Application. Technical report, NEPOMUK consortium, Dec. 2007. URL <http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D5-2>.
- M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004. ISSN 1046-8188. doi: 10.1145/963770.963776.
- H. Dicky, C. Dony, M. Huchard, and T. Libourel. On automatic class insertion with overloading. In *OOPSLA '96: Proceedings of the 11th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, pages 251–267, New York, NY, USA, 1996. ACM. ISBN 0-89791-788-X. doi: 10.1145/236337.236364.
- S. Dorogovtsev and J. Mendes. *Evolution of Networks: From Biological Nets to the Internet and WWW*. Oxford University Press, Oxford, Jan. 2003.
- M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 193–202, New York, NY, USA, May 2006. ACM. ISBN 1-59593-323-9. doi: 10.1145/1135777.1135810.
- F. Eisterlehner, A. Hotho, and R. Jäschke, editors. *Proceedings of the ECML PKDD Discovery Challenge 2009 (DC09)*, volume 497 of *CEUR-WS.org*, Sept. 2009.
- J. Ellson, E. Gansner, E. Koutsofios, S. North, and G. Woodhull. Graphviz and dynagraph – static and dynamic graph drawing tools. In M. Junger and P. Mutzel, editors, *Graph Drawing Software*, Mathematics and Visualization, pages 127–148. Springer, Berlin/Heidelberg, 2004. ISBN 3-540-00881-0.
- U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: an overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and

-
- R. Uthurusamy, editors, *Advances in knowledge discovery and data mining*, pages 1–34. American Association for Artificial Intelligence, Menlo Park, CA, USA, 1996. ISBN 0-262-56097-6.
- C. Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA, May 1998. ISBN 978-0-262-06197-1.
- R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. URL <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>.
- C.-S. Firan, W. Nejdl, and R. Paiu. The benefit of using tag-based profiles. In *5th Latin American Web Congress, October 31 - November 2 2007, Santiago de Chile*, 2007.
- E. Gamma, R. Helm, and R. E. Johnson. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam, 1st edition, 1995. ISBN 0201633612.
- B. Ganter. Algorithmen zur formalen Begriffsanalyse. In B. Ganter, R. Wille, and K. E. Wolff, editors, *Beiträge zur Begriffsanalyse*, pages 241–254. B.I.-Wissenschaftsverlag, Mannheim, 1987.
- B. Ganter and S. A. Obiedkov. Implications in triadic contexts. In *Conceptual Structures at Work: 12th International Conference on Conceptual Structures*, volume 3127 of *Lecture Notes in Computer Science*, pages 186–195, Berlin/Heidelberg, 2004. Springer. ISBN 3-540-22392-4.
- B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin/Heidelberg, 1999.
- B. Ganter, G. Stumme, and R. Wille, editors. *Formal Concept Analysis – Foundations and Applications*, volume 3626 of *Lecture Notes in Artificial Intelligence*, Berlin/Heidelberg, 2005. Springer.
- IT Baseline Protection Manual*. German Federal Office for Information Security, Oct. 2003. URL <http://www.bsi.de/gshb/>.
- R. Godin, H. Mili, G. W. Mineau, R. Missaoui, A. Arfi, and T.-T. Chau. Design of class hierarchies based on concept (galois) lattices. *Theor. Pract. Object Syst.*, 4(2):117–133, 1998. ISSN 1074-3227. doi: 10.1002/(SICI)1096-9942(1998)4:2(117::AID-TAPO6)3.3.CO;2-I.
- J. Golbeck. Trust on the world wide web: A survey. *Foundations and Trends in Web Science*, 1(2):131–197, Jan. 2006. ISSN 1555-077X. doi: 10.1561/18000000006.

- S. Golder and B. A. Huberman. The structure of collaborative tagging systems. *CoRR*, abs/cs/0508082, Aug. 2005. URL <http://arxiv.org/abs/cs/0508082>.
- M. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, Aug. 2007. ISSN 0343-2521. doi: 10.1007/s10708-007-9111-y.
- S. Green and J. Alexander. The advanced universal recommendation architecture (AURA) project. <http://www.tastekeeper.com/>, 2008.
- T. Groza, S. Handschuh, K. Moeller, G. Grimnes, L. Sauermann, E. Minack, C. Mesnage, M. Jazayeri, G. Reif, and R. Gudjónsdóttir. The NEPOMUK Project – On the way to the Social Semantic Desktop. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages 201–211. JUCS, Sept. 2007.
- T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
- P. Haase, A. Hotho, L. Schmidt-Thieme, and Y. Sure. Collaborative and usage-driven evolution of personal ontologies. In A. Gómez-Pérez and J. Euzenat, editors, *The Semantic Web: Research and Applications*, volume 3532 of *Lecture Notes in Computer Science*, pages 486–499, Berlin/Heidelberg, 2005. Springer. ISBN 3-540-26124-9. doi: 10.1007/11431053_33.
- H. Halpin, V. Robu, and H. Shepard. The dynamics and semantics of collaborative tagging. In K. Möller, A. de Waard, S. Cayzer, M.-R. Koivunen, M. Sintek, and S. Handschuh, editors, *Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW'06)*, volume 209 of *CEUR-WS.org*, Nov. 2006.
- T. Hammond, T. Hannay, B. Lund, and J. Scott. Social Bookmarking Tools (I): A General Review. *D-Lib Magazine*, 11(4), Apr. 2005.
- F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1983. ISBN 0-201-10686-8.
- J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *CSCW '00: Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, pages 241–250, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. doi: 10.1145/358916.358995.
- J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004. ISSN 1046-8188. doi: 10.1145/963770.963772.

-
- P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford InfoLab, Apr. 2006. URL <http://ilpubs.stanford.edu:8090/775/>.
- P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM '08: Proceedings of the International Conference on Web Search and Web Data Mining*, pages 195–206, New York, NY, USA, 2008a. ACM.
- P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 531–538, New York, NY, USA, 2008b. ACM. ISBN 978-1-60558-164-4. doi: 10.1145/1390334.1390425.
- A. Hotho. *Clustern mit Hintergrundwissen*, volume 286 of *Diski*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2004. ISBN 3-89838-286-9.
- A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. BibSonomy: A social bookmark and publication sharing system. In A. de Moor, S. Polovina, and H. Delugach, editors, *Proceedings of the Conceptual Structures Tool Interoperability Workshop at the 14th International Conference on Conceptual Structures*, pages 87–102, Aalborg, Denmark, July 2006a. Aalborg University Press. ISBN 87-7307-769-0.
- A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Berlin/Heidelberg, June 2006b. Springer.
- A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Trend detection in folksonomies. In Y. S. Avrithis, Y. Kompatsiaris, S. Staab, and N. E. O'Connor, editors, *Proc. First International Conference on Semantics And Digital Media Technology (SAMT)*, volume 4306 of *Lecture Notes in Computer Science*, pages 56–70, Berlin/Heidelberg, Dec. 2006c. Springer. ISBN 3-540-49335-2. doi: 10.1007/11930334_5.
- A. Hotho, B. Krause, D. Benz, and R. Jäschke, editors. *Proceedings of the ECML PKDD Discovery Challenge 2008 (RSDC'08)*, 2008. URL <http://www.kde.cs.uni-kassel.de/ws/rsdc08/>.
- G. T. I. Katakis and I. Vlahavas. Multilabel text classification for automated tag suggestion. In Hotho et al. (2008), pages 75–83. URL <http://www.kde.cs.uni-kassel.de/ws/rsdc08/pdf/9.pdf>.
- J. Illig. Entwurf und Integration eines Item-Based Collaborative Filtering Tag Recommender Systems in das BibSonomy-Projekt. Project report, Fachgebiet Wissensverarbeitung, Universität Kassel, 2006. URL <http://www.kde.cs.uni-kassel.de/lehre/arbeiten/documents/illig2006entwurf.pdf>.

- J. Illig, A. Hotho, R. Jäschke, and G. Stumme. A comparison of content-based tag recommendations in folksonomy systems. In *Postproceedings of the International Conference on Knowledge Processing in Practice (KPP 2007)*, 2009. (to appear).
- R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Trias - an algorithm for mining iceberg tri-lattices. In *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06)*, pages 907–911, Hong Kong, Dec. 2006. IEEE Computer Society. ISBN 0-7695-2701-9. doi: 10.1109/ICDM.2006.162.
- R. Jäschke, A. Hotho, C. Schmitz, and G. Stumme. Analysis of the publication sharing behaviour in BibSonomy. In U. Priss, S. Polovina, and R. Hill, editors, *Conceptual Structures: Knowledge Architectures for Smart Applications*, volume 4604 of *Lecture Notes in Computer Science*, pages 283–295, Berlin/Heidelberg, July 2007a. Springer. ISBN 3-540-73680-8. doi: 10.1007/978-3-540-73681-3.
- R. Jäschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron, editors, *Knowledge Discovery in Databases: PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, pages 506–514, Berlin/Heidelberg, 2007b. Springer. ISBN 978-3-540-74975-2. doi: 10.1007/978-3-540-74976-9_52.
- R. Jäschke, A. Hotho, C. Schmitz, B. Ganter, and G. Stumme. Discovering shared conceptualizations in folksonomies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):38–53, Feb. 2008a. ISSN 1570-8268. doi: 10.1016/j.websem.2007.11.004.
- R. Jäschke, B. Krause, A. Hotho, and G. Stumme. Logsonomy – a search engine folksonomy. In *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM 2008)*, pages 192–193, Menlo Park, CA, USA, 2008b. AAAI Press. ISBN 978-1-57735-355-3.
- R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in social bookmarking systems. *AI Communications*, 21(4):231–247, 2008c. ISSN 0921-7126. doi: 10.3233/AIC-2008-0438.
- R. Jäschke, F. Eisterlehner, A. Hotho, and G. Stumme. Testing and evaluating tag recommenders in a live system. In *RecSys '09: Proceedings of the 2009 ACM Conference on Recommender Systems*, pages 369–372, New York, NY, USA, Oct. 2009a. ACM. ISBN 978-1-60558-435-5. doi: 10.1145/1639714.1639790.
- R. Jäschke, F. Eisterlehner, A. Hotho, and G. Stumme. Testing and evaluating tag recommenders in a live system. In D. Benz and F. Janssen, editors, *Workshop on Knowledge Discovery, Data Mining, and Machine Learning*, pages 44–51, Sept. 2009b.

- M. Kamber, J. Han, and Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 207–210, Menlo Park, CA, USA, Aug. 1997. AAAI Press. ISBN 978-1-57735-027-9.
- H. L. Kim, S. Scerri, J. G. Breslin, S. Decker, and H. G. Kim. The state of the art in tag ontologies: a semantic model for tagging and folksonomies. In *DCMI '08: Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications*, pages 128–137. Dublin Core Metadata Initiative, 2008.
- J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. ISSN 0004-5411. doi: 10.1145/324133.324140.
- D. E. Knuth. *The art of computer programming*, volume 3. Addison-Wesley Longman Publishing Co., Boston, MA, USA, 2nd edition, 1998. ISBN 0-201-89685-0.
- P. Kolari, T. Finin, Y. Yesha, Y. Yesha, K. Lyons, S. Perelgut, and J. Hawkins. On the Structure, Properties and Utility of Internal Corporate Blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, Mar. 2007.
- G. E. Krasner and S. T. Pope. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object Oriented Programming*, 1(3): 26–49, 1988. ISSN 0896-8438.
- B. Krause, R. Jäschke, A. Hotho, and G. Stumme. Logsonomy – social information retrieval with logdata. In *HT '08: Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia*, pages 157–166, New York, NY, USA, 2008a. ACM. ISBN 978-1-59593-985-2. doi: 10.1145/1379092.1379123.
- B. Krause, C. Schmitz, A. Hotho, and G. Stumme. 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*, pages 61–68, New York, NY, USA, Apr. 2008b. ACM. ISBN 978-1-60558-159-0. doi: 10.1145/1451983.1451998.
- S. Krolak-Schwerdt, P. Orlik, and B. Ganter. TRIPAT: a model for analyzing three-mode binary data. In H. H. Bock, W. Lenski, and M. M. Richter, editors, *Studies in Classification, Data Analysis, and Knowledge Organization*, volume 4 of *Information systems and data analysis*, pages 298–307. Springer, Berlin/Heidelberg, 1994.
- S. K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 393–402, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988726.

- R. Lambiotte and M. Ausloos. Collaborative tagging as a tripartite network. *CoRR*, abs/cs/0512090, Dec. 2005. doi: 10.1007/11758532_152. URL <http://arxiv.org/abs/cs.DS/0512090>.
- L. Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.
- F. Lehmann and R. Wille. A triadic approach to formal concept analysis. In G. Ellis, R. Levinson, W. Rich, and J. F. Sowa, editors, *Conceptual Structures: Applications, Implementation and Theory*, volume 954 of *Lecture Notes in Artificial Intelligence*, pages 32–43, Berlin/Heidelberg, 1995. Springer. ISBN 3-540-60161-9.
- B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 227–230, Menlo Park, CA, USA, Aug. 1997. AAAI Press. ISBN 978-1-57735-027-9.
- D. Lin and Z. M. Kedem. Pincer-Search: A new algorithm for discovering the maximum frequent set. In *Proceedings of the 6th Biennial International Conference on Extending Database Technology (EDBT'98)*, volume 1377 of *Lecture Notes in Computer Science*, pages 105–119, Berlin/Heidelberg, Mar. 1998. Springer. ISBN 3-540-64264-1.
- M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In Hotho et al. (2008), pages 84–95. URL <http://www.kde.cs.uni-kassel.de/ws/rsdc08/pdf/10.pdf>.
- M. Lipczak, Y. Hu, Y. Kollet, and E. Milios. Tag sources for recommendation in collaborative tagging systems. In Eisterlehner et al. (2009), pages 157–172.
- B. Lund, T. Hammond, M. Flack, and T. Hannay. Social Bookmarking Tools (II): A Case Study - Connotea. *D-Lib Magazine*, 11(4), Apr. 2005.
- M. S. M. Tatu and T. D'Silva. RSDC'08: Tag recommendations using bookmark content. In Hotho et al. (2008), pages 96–107. URL <http://www.kde.cs.uni-kassel.de/ws/rsdc08/pdf/11.pdf>.
- A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.
- H. Mannila. Methods and problems in data mining. In *Proceedings of the 6th biennial International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 41–55, Berlin/Heidelberg, Jan. 1997. Springer.
- C. Marlow, M. Naaman, D. Boyd, and M. Davis. Position Paper, Tagging, Taxonomy, Flickr, Article, ToRead. In *Collaborative Web Tagging Workshop at WWW2006*, Edinburgh, Scotland, May 2006.

- A. Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata, Dec. 2004. URL <http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html>.
- A. K. McCallum. *MALLET: A Machine Learning for Language Toolkit*, 2002. URL <http://mallet.cs.umass.edu/>.
- P. Mika. Ontologies are us: A unified model of social networks and semantics. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 522–536, Berlin/Heidelberg, Nov. 2005. Springer. doi: 10.1007/11574620_38.
- D. Millen, J. Feinberg, and B. Kerr. Social bookmarking in the enterprise. *Queue*, 3(9): 28–35, 2005. ISSN 1542-7730. doi: 10.1145/1105664.1105676.
- G. Mineau, G., and R. Godin. Automatic structuring of knowledge bases by conceptual clustering. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):824–829, 1985.
- G. Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM Press. ISBN 1595933239. doi: 10.1145/1135777.1135961.
- M. Missikoff and M. Scholl. An algorithm for insertion into a lattice: application to type classification. In *Proc. 3rd Intl. Conf. FODO 1989*, volume 367 of *Lecture Notes in Computer Science*, pages 64–82, Berlin/Heidelberg, 1989. Springer.
- M. E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20): 208701, Oct. 2002. doi: 10.1103/PhysRevLett.89.208701.
- J. O'Donovan and B. Smyth. Trust in recommender systems. In *IUI '05: Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 167–174, New York, NY, USA, 2005. ACM. ISBN 1-58113-894-6. doi: 10.1145/1040830.1040870.
- S. Oldenburg, M. Garbe, and C. Cap. Similarity cross-analysis of tag / co-tag spaces in social classification systems. In *SSM '08: Proceeding of the 2008 ACM Workshop on Search in Social Media*, pages 11–18, New York, NY, USA, Oct. 2008. ACM. ISBN 978-1-60558-258-0. doi: 10.1145/1458583.1458587.
- L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, Nov. 1999. URL <http://ilpubs.stanford.edu:8090/422/>.

- B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, Jan. 2008. ISSN 1554-0669. doi: 10.1561/1500000011.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Closed set based discovery of small covers for association rules. In *Actes des 15èmes journées Bases de Données Avancées (BDA'99)*, pages 361–381, Oct. 1999a.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th biennial International Conference on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 398–416, Berlin/Heidelberg, Jan. 1999b. Springer.
- N. Pasquier, R. Taouil, Y. Bastide, G. Stumme, and L. Lakhal. Generating a condensed representation for association rules. *J. Intelligent Information Systems (JIIS)*, 24(1): 29–60, 2005.
- G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06: Proceedings of the 1st International Conference on Scalable Information Systems*, page 1, New York, NY, USA, 2006. ACM. ISBN 1-59593-428-6. doi: 10.1145/1146847.1146848.
- O. Patashnik. *BibTeXing*, 1988. (Included in the BIB_TE_X distribution).
- J. Pei, J. Han, and R. Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.
- C. S. Peirce. *Collected Papers*. Harvard University Press, Cambridge, 1931–1935.
- F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, pages 329–336, 2004.
- D. Quan, D. Huynh, and D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The SemanticWeb - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 738–753, Berlin/Heidelberg, Sept. 2003. Springer. ISBN 978-3-540-20362-9. doi: 10.1007/b14287.
- T. Rattenbury, N. Good, and M. Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 103–110, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-597-7. doi: 10.1145/1277741.1277762.

-
- K. Regulski. Aufwand und Nutzen beim Einsatz von Social-Bookmarking-Services als Nachweisinstrument für wissenschaftliche Forschungsartikel am Beispiel von BibSonomy. *Bibliothek. Forschung und Praxis*, 31(2):177–184, 2007. ISSN 0341-4183.
- G. Reif, T. Groza, S. Scerri, and S. Handschuh. Nepomuk Deliverable D6.2.B – Final NEPOMUK Architecture. Technical report, NEPOMUK consortium, Dec. 2008. URL <http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D6-2-B>.
- P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. doi: 10.1145/192844.192905.
- F. Rioult. *Extraction de connaissances dans les bases de donnees comportant des valeurs manquantes ou un grand nombre d'attributs*. PhD thesis, Université de Caen Basse-Normandie, 2005.
- R. L. Rivest. *The MD5 Message Digest Algorithm*, Apr. 1992. URL <ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt>.
- J. Röttgers. Am Ende der Flegeljahre — Das Web 2.0 wird erwachsen. *c't 25/2007*, page 148, 2007.
- E. Santos-Neto, M. Ripeanu, and A. Iamnitchi. Tracking user attention in collaborative tagging communities. *CoRR*, abs/0705.1013, June 2007. URL <http://arxiv.org/abs/0705.1013>.
- B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM. ISBN 1-58113-348-0. doi: 10.1145/371920.372071.
- L. Sauermann. The gnowsis – using semantic web technologies to build a semantic desktop. Diploma thesis, Technical University of Vienna, 2003. URL <http://www.dfki.uni-kl.de/~sauermann/papers/sauermann2003.pdf>.
- L. Sauermann, A. Bernardi, and A. Dengel. Overview and outlook on the semantic desktop. In S. Decker, J. Park, D. Quan, and L. Sauermann, editors, *Proceedings of the 1st Workshop on The Semantic Desktop at the ISWC 2005 Conference*, volume 175 of *CEUR-WS.org*, pages 1 – 18. CEUR-WS, Nov. 2005.
- S. Scerri, M. Sintek, L. van Elst, and S. Handschuh. *NEPOMUK Annotation Ontology Specification*. NEPOMUK consortium, 2007. URL <http://www.semanticdesktop.org/ontologies/nao/>.

- J. Schachter. now serving: 1,000,000. Blog post, Sept. 2006. URL <http://blog.delicious.com/blog/2006/09/million.html>.
- I. Schmitt and G. Saake. Merging inheritance hierarchies for database integration. In *COOPIS '98: Proceedings of the 3rd IFICIS International Conference on Cooperative Information Systems*, pages 322–331, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8380-5.
- C. Schmitz, A. Hotho, R. Jäschke, and G. Stumme. Mining association rules in folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. Žiberna, editors, *Data Science and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Berlin/Heidelberg, 2006. Springer. ISBN 978-3-540-34415-5. doi:10.1007/3-540-34416-0_28.
- P. Schmitz. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop at WWW2006*, Edinburgh, Scotland, May 2006.
- X. Shi. Social network analysis of web search engine query logs. Technical report, School of Information, University of Michigan, 2007.
- X. Si, Z. Liu, P. Li, Q. Jiang, and M. Sun. Content-based and graph-based tag suggestion. In Eisterlehner et al. (2009), pages 243–260.
- C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets : Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, Jan. 1998.
- G. Smith. Search tagging. Blog post, May 2005. URL http://atomiq.org/archives/2005/05/search_tagging.html.
- S. Sood, S. Owsley, K. Hammond, and L. Birnbaum. TagAssist: Automatic tag suggestion for blog posts. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, Boulder, Colorado, USA, 2007.
- E. Speller. Collaborative tagging, folksonomies, distributed classification or ethnoclassification: a literature review. *Library Student Journal*, Feb. 2007. URL http://informatics.buffalo.edu/org/ljsj/articles/speller_2007_2_collaborative.php.
- S. Staab, S. Santini, F. Nack, L. Steels, and A. Maedche. Emergent semantics. *IEEE Intelligent Systems*, 17(1):78–86, 2002.
- R. Stecher, P. Haghani, R. Jäschke, and C. Kohlschütter. Nepomuk Deliverable D5.2 – Social Network Software. Technical report, NEPOMUK consortium, Nov. 2008. URL <http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D5-2>.

- L. Steels. Collaborative tagging as distributed cognition. *Pragmatics & Cognition*, 14(2):287–292, 2006. ISSN 0929-0907.
- L. Steels. The origins of ontologies and communication conventions in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 1(2):169–194, Oct. 1998.
- S. Strahringer and R. Wille. Conceptual clustering via convex-ordinal structures. In O. Opitz, B. Lausen, and R. Klar, editors, *Information and Classification*, pages 85–98, Berlin/Heidelberg, 1993. Springer.
- M. Strohmaier. Purpose tagging: capturing user intent to assist goal-oriented social search. In *SSM '08: Proceedings of the 2008 ACM Workshop on Search in Social Media*, pages 35–42, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-258-0. doi: 10.1145/1458583.1458603.
- G. Stumme. Off to new shores – conceptual knowledge discovery and processing. *International Journal on Human-Computer Studies (IJHCS)*, 59(3):287–325, Sept. 2003.
- G. Stumme. A finite state model for on-line analytical processing in triadic contexts. In B. Ganter and R. Godin, editors, *Proceedings of the 3rd International Conference on Formal Concept Analysis*, volume 3403 of *Lecture Notes in Computer Science*, pages 315–328, Berlin/Heidelberg, 2005. Springer. ISBN 3-540-24525-1.
- G. Stumme. Conceptual knowledge discovery with frequent concept lattices. FB4-Preprint 2043, TU Darmstadt, 1999. URL <http://www.kde.cs.uni-kassel.de/stumme/papers/1999/P2043.pdf>.
- G. Stumme and R. Wille, editors. *Begriffliche Wissensverarbeitung–Methoden und Anwendungen*. Springer, Berlin/Heidelberg, 2000.
- G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Intelligent structuring and reducing of association rules with formal concept analysis. In F. Baader, G. Brewker, and T. Eiter, editors, *KI 2001: Advances in Artificial Intelligence*, volume 2174 of *Lecture Notes in Computer Science*, pages 335–350, Berlin/Heidelberg, 2001. Springer. ISBN 978-3-540-42612-7. doi: 10.1007/3-540-45422-5_24.
- G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with titanic. *Journal on Knowledge and Data Engineering*, 42(2):189–222, 2002.
- H. Söll. Begriffliche Analyse triadischer Daten: Das IT-Grundschutzhandbuch des Bundesamts für Sicherheit in der Informationstechnik. Diploma thesis, FB Mathematik, TU Darmstadt, Darmstadt, Apr. 1998.

- R. Taouil. *Algorithmique du treillis des fermés : application à l'analyse formelle de concepts et aux bases de données*. PhD thesis, Université de Clermont-Ferrand II, 2000.
- K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 1995–1999, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7. doi: 10.1145/1363686.1364171.
- M. van der Graaf. Academic social referencing tools: a user trial with BibSonomy and Cite-U-Like organized by the Library of the University of Amsterdam, Mar. 2007. URL http://cf.uba.uva.nl/nl/projecten/academic_social_referencing.pdf. User study.
- M. van Setten. *Supporting people in finding information : hybrid recommender systems and goal-based structuring*. PhD thesis, University of Twente, Enschede, The Netherlands, Dec. 2005.
- T. Vander Wal. Explaining and showing broad and narrow folksonomies. Blog post, Feb. 2005. URL http://www.personalinfocloud.com/2005/02/explaining_and_.html.
- T. Vander Wal. Folksonomy. Blog post, Feb. 2007. URL <http://vanderwal.net/folksonomy.html>.
- J. Vig, S. Sen, and J. Riedl. Tagsplanations: explaining recommendations using tags. In *IUI '09: Proceedings of the 13th International Conference on Intelligent User Interfaces*, pages 47–56, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-168-2. doi: 10.1145/1502650.1502661.
- M. Vojnovic, J. Cruise, D. Gunawardena, and P. Marbach. Ranking and suggesting tags in collaborative tagging applications. Technical Report MSR-TR-2007-06, Microsoft Research, Feb. 2007.
- J. Voss. Collaborative thesaurus tagging the wikipedia way. *CoRR*, abs/cs/0604036, Apr. 2006. URL <http://arxiv.org/abs/cs/0604036v2>.
- J. Voss, H. Andreas, and J. Robert. Mapping bibliographic records with bibliographic hash keys. In R. Kuhlen, editor, *Information: Droge, Ware oder Commons?*, Proceedings of the ISI. Hochschulverband Informationswissenschaft, Verlag Werner Hülsbusch, 2009.
- K. Waiyamai, R. Taouil, and L. Lakhal. Towards an object database approach for managing concept lattices. In *Conceptual Modeling – ER '97*, volume 1331 of *Lecture Notes in Computer Science*, pages 299–312, Berlin/Heidelberg, 1997. Springer. ISBN 978-3-540-63699-1. doi: 10.1007/3-540-63699-4_24.

-
- D. J. Watts and S. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, June 1998.
- R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston, 1982.
- R. Wille. The basic theorem of triadic concept analysis. *Order*, 12:149–158, 1995.
- R. Wille and M. Zickwolff. Grundlagen einer triadischen Begriffsanalyse. In G. Stumme and R. Wille, editors, *Begriffliche Wissensverarbeitung. Methoden und Anwendungen*, pages 125–150, Berlin/Heidelberg, 2000. Springer.
- W. Xi, B. Zhang, Z. Chen, Y. Lu, S. Yan, W.-Y. Ma, and E. A. Fox. Link fusion: a unified link analysis framework for multi-type interrelated data objects. In *WWW ’04: Proceedings of the 13th International Conference on World Wide Web*, pages 319–327, New York, NY, USA, 2004. ACM. ISBN 1-58113-844-X. doi: 10.1145/988672.988715.
- Y. Xu, L. Zhang, and W. Liu. Cubic analysis of social bookmarking for personalized recommendation. *Frontiers of WWW Research and Development - APWeb 2006*, pages 733–738, 2006a. doi: 10.1007/11610113_66.
- Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In *Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006*, Edinburgh, Scotland, 2006b.
- G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *CIKM ’04: Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management*, pages 118–126, New York, NY, USA, 2004. ACM. ISBN 1-58113-874-1. doi: 10.1145/1031171.1031192.
- A. Yahia, L. Lakhal, J. P. Bordat, and R. Cicchetti. IO2: An algorithmic method for building inheritance graphs in object database design. In *Conceptual Modeling – ER ’96*, volume 1157 of *Lecture Notes in Computer Science*, pages 422–437, Berlin/Heidelberg, 1996. Springer. ISBN 978-3-540-61784-6. doi: 10.1007/BFb0019938.
- S. A. Yahia, M. Benedikt, L. V. S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *Proceedings of the VLDB Endowment*, 1(1):710–721, Aug. 2008. doi: 10.1145/1453856.1453934.
- M. J. Zaki and C.-J. Hsiao. Charm: An efficient algorithm for closed association rule mining. technical report 99–10. Technical report, Computer Science Dept., Rensselaer Polytechnic, Oct. 1999.
- M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proceedings of the 3rd international conference on Knowledge Discovery and Data Mining (KDD’97)*, pages 283–286. AAAI Press, August 1997.

- D. Zhang and Y. Dong. A novel web usage mining approach for search engines. *Computer Networks*, 39(3):303–310, June 2002. ISSN 1389-1286. doi: 10.1016/S1389-1286(02)00211-6.
- E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *WWW '09: Proceedings of the 18th International Conference on World Wide Web*, pages 531–540, New York, NY, USA, Apr. 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526781.