Integrated Project

Priority 2.4.7

Semantic based knowledge systems

Information Society Technologies

The Social Semantic Desktop

NEPOMUK

Community Support Software

First Version

Deliverable D5.1

Version 1.1

29.06.2007

Dissemination level: PU

Nature	Prototype
Due date	30.06.2007
Lead contractor	LEIBNIZ UNIVERSITAET HANNOVER
Start date of project	01.01.2006
Duration	36 months

Authors

Gianluca Demartini
Parisa Haghani
Robert Jäschke
Ann Johnston
Malte Kiesel
Raluca Paiu

Mentors

Stéphane Laurière, EDGE-IT S.A.R.L
Bosse Westerlund, KUNGLIGA TEKNISKA HOEGSKOLAN

Contributors

Vasilios Darlagiannis
Pär Lannerö

Project Co-ordinator

Dr. Ansgar Bernardi
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Str. 122
67663 Kaiserslautern
Germany
E-Mail: bernardi@dfki.uni-kl.de, phone: +49 631 205 75 105

Partners

DEUTSCHES FORSCHUNGSZENTRUM F. KUENSTLICHE INTELLIGENZ GMBH
IBM IRELAND PRODUCT DISTRIBUTION LIMITED
SAP AG
HEWLETT PACKARD GALWAY LTD
THALES S.A.
PRC GROUP - THE MANAGEMENT HOUSE S.A.
EDGE-IT S.A.R.L
COGNIUM SYSTEMS S.A.
NATIONAL UNIVERSITY OF IRELAND, GALWAY
ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
FORSCHUNGSZENTRUM INFORMATIK AN DER UNIVERSITAET KARLSRUHE
LEIBNIZ UNIVERSITAET HANNOVER
INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS
KUNGLIGA TEKNISKA HOEGSKOLAN
UNIVERSITA DELLA SVIZZERA ITALIANA
IRION MANAGEMENT CONSULTING GMBH

Copyright: NEPOMUK Consortium 2007
Copyright on template: Irion Management Consulting GmbH 2007

Versions

Version	Date	Reason
1.0	06.06.2007	Final version for the mentors as subject to be reviewed
1.1	29.06.2007	Integration of mentor's comments and released as final

Explanations of abbreviations on front page

Nature
R: Report
P: Prototype
R/P: Report and Prototype
O: Other

Dissemination level
PU: Public
PP: Restricted to other FP6 participants
RE: Restricted to specified group
CO: Confidential, only for NEPOMUK partners

Executive summary

WP5000 investigates and exploits the knowledge exchange in social networks and provides tools and services for community identification and analysis and for supporting knowledge exchange in these semantic social networks. The goal of the first deliverable is to provide a first version of the community support software which is composed by a documented and tested prototype of a community detection algorithm and (semi-) automatic metadata alignment. The deliverable provides community detection and metadata alignment functionalities, packaged as two components, Community Manager and Metadata Aligner, and integrated into the NEPOMUK component architecture. Although they are not applications on their own, they form an infrastructure which can be adopted and used by other applications.

Our work resulted in the following:

- The Community Manager[1] component, which identifies social structures in the network of NEPOMUK peers, analyzes these structures and provides methods for other components or applications to access the results.
- The Metadata Alignment[2] component, being responsible for generating mappings between ontologies used by different peers in the NEPOMUK network.

To accomplish the community detection and labeling task provided by the Community Manager,

- we describe how the folksonomy structure of the NEPOMUK network can be analyzed to detect communities of users within the network, introducing social resource sharing systems and formalizing the underlying data structure called folksonomy;
- we present the FolkRank algorithm and its application to community detection;
- we describ the Trias algorithm for mining frequent closed tri-sets of a folksonomy.

The implementation of the Community Manager component, done with this deliverable, allows the extraction of communities of users within the NEPOMUK P2P[3] network. Communities can be defined as groups of users belonging together or groups of users with affinity to a certain topic.

The main task of the Metadata Aligner component is to identify relationships between personal ontologies as created and used by end users. Knowledge of relationships is necessary to implement features such as recommendation of similar documents within the workspace of other users as well as enhancing semantic search. Within the scope of NEPOMUK, we will not formally infer class relationships, but gather evidence for such relationships. This is due to the fact that the PIMO ontologies we want to align are ontologies created by end-users, therefore mostly having a kind of ad-hoc character instead of being completely formally sound.

For the realization of the Metadata Aligner component,

[1] Located on the NEPOMUK SVN Server `https://dev.nepomuk.semanticdesktop.org` in the directory /repos/trunk/java/org.semanticdesktop.nepomuk.comp.communitymanager [2] Located on the NEPOMUK SVN Server `https://dev.nepomuk.semanticdesktop.org` in the directory /repos/trunk/java/org.semanticdesktop.nepomuk.comp.phasealignment [3] Abbreviations are listed in the Appendix A.

- we present a framework that collects and integrates heuristic evidence for ontology mappings. The framework can employ three basic sources of evidence for ontology mappings, namely term-based, topology-based and instance-based evidence.

We have implemented the basic functionalities for the semantic social networking and knowledge exchange contributions of WP5000 to the Social Semantic Desktop NEPOMUK. We developed, implemented, and analyzed methods for community detection and labeling, as well for (semi-)automatic metadata alignment[4].

[4] Note that some of the these functionalities, especially those focused on metadata recommendations, will be described in detail only in deliverable D5.2

Table of contents

1 Introduction

In the context of NEPOMUK - **The Social Semantic Desktop**- the **social** aspect is brought in by WP5000: the social networking components helping to build and maintain topic- and content-specific interconnections between distributed individual workspaces.

The aim of this deliverable is to provide a first version of a documented and tested prototype of community detection algorithm and (semi-) automatic metadata alignment. Even though social network analysis algorithms do already exist, they are only based on static information (e.g. FOAF, in which people manually specify their friends). Our goal was therefore to identify solutions to automatically infer these communities, as well as to find and join them easily. Moreover, since we will face very large networks of independent users, it will be difficult to enforce global metadata schemes. Therefore, another goal was to investigate possible metadata alignment techniques, as well as means to (semi-)automate this process.

To detect communities by analyzing the conceptual commonalities of users reasonable amounts of data are needed. Social information sharing tools, such as Flickr[5] or del.icio.us,[6] have acquired large numbers of users, who have created huge amounts of information within less than two years. One reason for their immediate success is the fact that no specific skills are needed for participating, and that these tools yield immediate benefit for each individual user (e. g. organizing ones bookmarks in a browser-independent, persistent fashion) without too much overhead. 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.

Social resource sharing systems all use the same kind of lightweight knowledge representation, called folksonomy, that is conceptual structures created by the people. In NEPOMUK a folksonomy is inherently present in the network of peers which contains the aggregated information of its users. Applications like the 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 files they managed with BibSonomy[7]. Hence, in plenty of the NEPOMUK scenarios a rich amount of tagged resources will be available on the local desktop of each user.

To this end we propose a formal model for folksonomies, and present a new algorithm, called FolkRank, that takes into account the folksonomy structure to detect communities of users in folksonomy based systems. This algorithm is implemented in the Community Manager component of deliverable D5.1 and will be described in detail in Section 5.3. It has been published at the European Semantic Web Conference 2006 [30].

Another approach for community detection presented in this deliverable is based on Formal Concept Analysis (FCA), a mathematical theory for deriving concept hierarchies from data. We give a formal definition of the problem of mining all frequent tri-concepts (the three-dimensional version of mining all frequent closed itemsets), and present in Section 5.5 our algorithm Trias for mining all frequent tri-concepts of a given dataset [31].

The subsequent aspect to be taken into account is the one of sharing the information created through the resource annotation process. The problem of sharing semantically rich data, is well known in today's information-driven world and much effort has been put on it. This task is inherently difficult, since data represented by different ontologies or schemas need to be mapped or translated into known formats so that queries in those formats can be

[5] http://www.flickr.com/ [6] http://del.icio.us [7] `http://www.bibsonomy.org`

run over them. The motivating reasons of this task are twofold: First, the benefits of semantic sharing are invaluable, and Second, to gain semantic interoperability, we need to overcome the semantic heterogeneity obstacle by producing mappings between schemas. The approach of generating a global ontology or schema, even for a specific domain, to overcome the semantic heterogeneity is now obsolete. In this approach the idea is to maintain a global schema and mappings from other schemas to this one. This is not feasible due to the decentralized nature of the semantic web, as all users would need to consistently follow the global schema.

The problem of producing metadata alignments can be defined as finding the correspondences between elements of two different ontologies or schemas. Much work has been done in this regard, based on several different techniques. Our approach, however, is based on gathering evidences from different sources to infer mappings. This approach can be either totally automatic or it can take into account user's feedback (as new evidence) and be considered as semi-automatic.

The deliverable is structured as follows. In Section 2 we describe the requirements and objectives of the community manager component, of the semi automatic metadata alignment component, and of the integration of these two. Then, in Section 3 we illustrate the state of the art in the field of community detection, social network analysis in folksonomies, and (semi-) automatic metadata alignment. After this, in Section 4, we describe the architecture of the community support component. In Section 5 we describe how the folksonomy structure of the NEPOMUK network is analyzed to detect communities of users. In Section 6 we show our approach to identify relationships between elements of given ontologies by aggregating different available evidences. Section 7 concludes the document summarizing and discussing the obtained results.

2 Requirements and Objectives

In the following we describe the requirements and objectives of the community support software to motivate the components presented in this deliverable.

WP5000 investigates and exploits the knowledge exchange in social networks and, building upon the basic infrastructure in WP2000 and WP4000, provides tools and services for community identification and analysis and for supporting knowledge exchange in these semantic social networks.

The goal of the first deliverable within WP5000 is to provide a first version of the community support software which is composed by a documented and tested prototype of community detection algorithm and (semi-) automatic metadata alignment.

2.1 Community Manager

For this component, the focal point is on Semantic Social Network Analysis to detect communities of users which share similiar interests. To gain insight how the detection of communities can improve the workflow of the individual NEPOMUK user, imagine the first Mandriva scenario[8] which describes how Kim searches for help in the Mandriva Club to connect a harddisk with a relatively new technology. The web page presenting the search results could also contain a community of users related to the search terms he 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 Community Manager as belonging to that community because they tagged the same or similiar resources with the same or similiar tags. Hence, they seem to be interested in the kind of problem Kim has and might even have solved similiar 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.

Consequently, for the Community Manager component we see the following requirements:

- analyze and construct a community infrastructure model upon which further applications will be deployed;
- apply and mutually augment methods of social network analysis and the semantic web technologies over the Social Semantic Desktop;
- provide a prototypical implementation to detect communities of collaborators who share common interests, and make their structure explicit, thus enabling various additional enhancements, such as easier finding and joining of relevant communities.

2.2 (Semi-)Automatic Metadata Alignment

For this component, the focal point is on Metadata extraction and automatic metadata alignment.

[8] `http://nepomuk.semanticdesktop.org/xwiki/bin/view/KTH/MandrivaScenario1`

In a very large network of independent users, it will be hard to enforce global metadata schemas. The goal is to investigate possible metadata alignment methods, as well as means to automate these proposed solutions.

As a good example of how metadata alignment can be used in the context of the social semantic desktop, we can refer to a similar scenario to the one presented in D1.1 as "Ontology enhanced search in semantic helpdesk". Kim is looking for an answer to his current problem (how to install his new graphic card) and issues a query, which tries to find the right driver for his type of graphic card, produced by a certain manufacturer. Formalized, the query looks as follows:

(SELECT ?d WHERE ?x hasManufacturer A AND ?x rdf:type GraphicCard AND ?x hasDriver ?d).

Alistair holds this information, but unfortunately Kim's ontology for computer gadgets is different from Alistair's. If a mapping between the two ontologies would exist in the system, Alistair's data could also be searched and the search function could provide Kim with the answer he needs.

For the (semi-) Automatic Metadata Alignment component we see the following requirements:

- Scalability: define metadata alignment algorithms which allow large networks of independent users, each utilizing a different ontology.
- Quality: the need for these users to interact seamlessly with each other and share structured data.

Since much research has already been done in the area of ontology alignment, we define the following as our objectives for the (semi-)Automatic Metadata Alignment component:

- investigating the existing alignment methods which utilize different types of information to produce mappings between ontologies.
- implementation of some of these approaches, and devising methods for aggregating the results of them to utilize all available information.

Metadata Alignment is one of the mentioned requirements in WP9000 - Professional Business Services Case Study.

3 State of the Art

This section presents the state of the art in research regarding topics which are affected by the algorithms developed in WP5000 and delivered in D5.1. First, we present a selection of techniques which have been used in the past for community detection in complex networks, then we proceed with analysis which has been done in the fields of folksonomies and (Triadic) Formal Concept Analysis. We conclude with an overview on publications about metadata alignment.

3.1 Community Detection

People have used the social network metaphor for almost a century to connote complex sets of relationships between members of social systems at all scales, from interpersonal to international. Starting with the mid 50's, social scientists start using the term systematically to denote patterns of ties that cut across the concepts traditionally used by the public and social scientists: bounded groups (e.g., tribes, families) and social categories (e.g., gender, ethnicity). Also around that time social network analysis has emerged as a key technique in sociology, anthropology, sociolinguistics, geography, social psychology, information science and organizational studies, as well as a popular topic of speculation and study.

Nowadays, social network analysis has moved from being a suggestive metaphor to an analytic approach to a paradigm, with its own theoretical statements, methods and research tribes. Analysts reason from whole to part; from structure to relation to individual; from behavior to attitude. They either study whole networks, all of the ties containing specified relations in a defined population, or personal networks, the ties that specified people have, such as their "personal communities".

Over the past decade, complex networks have attracted an increasing interest, many research areas tackling aspects like information flow within the network, extracting information about the birth of the network or its growth mechanisms. Some examples of complex networks include:

- **co-authorship networks**—where nodes are scientists and an edge is drawn between two nodes if they co-authored one paper
- **the Internet**—where nodes are routers and edges represent the connectivity between the routers
- **protein networks**—where nodes are proteins and edges represent the protein interactions between the nodes
- **friendship networks**—where nodes are people and edges represent the friendship relationships
- **folksonomies**—with users, tags and resources as nodes and a hyperedge connecting each one of them if a user has tagged a resource with a tag

Due to its applicability to a wide scale of disciplines, community detection has also become an important topic. One of the most relevant community detection algorithms was presented in [49], and is based on removal of the edges according to edge betweenness. The edges are repeatedly removed from the network, starting from the node with highest betweenness centrality. After each removal the edge betweenness is recomputed and the algorithms stops where no more edges can be found between vertices. For small networks this divisive algorithm performs very good and finds very accurate results.

Similar to the algorithm in [49] is the approach described in [56]. However, the measure they use is different: instead of edge betweenness this approach is based on counting the short loops in the network, loops of length three, or triangles, in the simplest case. As in the previous algorithm, this measure is recalculated after each removal, but since it is a local measure that can be calculated quickly, the overall algorithm is much faster.

A totally different approach is presented in [16]—the first proposed algorithm for community detection in very large networks. The algorithm is a hierarchical agglomerative approach based on network modularity (a measure which determines the quality of splitting a network into communities). It starts with n communities of n vertices and at each step it merges communities into bigger communities, such that those communities that are picked to be merged yield the highest increase in network modularity. Each potential merge of two communities has a contribution of ΔQ to network modularity Q, and the idea is to merge two communities that yield the highest increase in Q, that is the selection of the tuple with highest ΔQ value at each step.

Another divisive algorithm is presented in [22], which is based on network modularity Q. The network is divided into two random communities, and each node in the communities has a contribution value λ to the network modularity Q. At each step the node with the lowest λ is transferred to the other community. The transfer of nodes between communities continues until there is no increase in the Q value of the network. Then the network is divided into the two communities and the algorithm continues the same process recursively in the smaller communities.

The community detection algorithms mainly try to find communities by dividing the network into "reasonable" partitions or reversely merge nodes iteratively into the same community. Other approaches try to optimize a global property of the network, which is in most cases the network modularity, Q. Nevertheless, none of the described algorithms has been applied to the ternary hypergraph structure of a folksonomy.

3.2 Social Network Analysis in Folksonomies

There are currently only few scientific publications about folksonomy-based web collaboration systems available. The main discussion on folksonomies and related topics is currently taking place on mailing lists, e.g. [17, 67], or blogs.

Two of the first publications in this new field of research are [28] and [41] which provide good overviews of social bookmarking tools with special emphasis on folksonomies, and [43] which discusses strengths and limitations of folksonomies. In [47], Mika defines a model of semantic-social networks for extracting lightweight ontologies from del.icio.us. Besides calculating measures like the clustering coefficient, (local) betweenness centrality or the network constraint on the extracted one-mode network, Mika uses co-occurence techniques for clustering the folksonomy. The focus of those works is on getting an overview of social resource sharing systems and an overall insight in the folksonomy structure.

More recently, work on more specialized topics such as structure mining on folksonomies—e.g. to visualize trends [21] and patterns [59] in users' tagging behavior has been presented. Along those lines of research are articles on analyzing the semiotic dynamics of the tagging vocabulary [15], or examining the dynamics and semantics of folksonomy systems [27]. Those approaches focus on using standard analysis methods to visualize properties of folksonomies. They are a good starting point for developing specialized algorithms for folksonomy analysis but don't provide robust techniques, e.g., to extract clusters automatically. In [30] we presented how an adaptation of the PageRank [11] algorithm can be used for ranking folksonomy contents. The algorithm presented there is the basis for the community detection work done by the Community Manager and is described in detail in Section 5. It allows unsupervised extraction of communities of users from folksonomies.

Another upcoming field of research is the learning of more formal, usually hierarchical conceptual structures (i.e. taxonomies, ontologies) from folksonomies, which has been approached using different mining techniques [48, 60, 29, 36].

One other approach for social network analysis is Formal Concept Analysis (FCA), a mathematical theory for deriving concept hierarchies from data. The amount of publications on Formal Concept Analysis is large. A good starting point for the lecture are the textbooks [26, 14, 25], as well as the proceedings of the Intl. Conference on Formal Concept Analysis[9] and the Intl. Conference on Conceptual Structures[10] series.

The problem of mining frequent itemsets arose first as a sub-problem of mining association rules [1], but it then turned out to be present in a variety of problems: mining sequential patterns [3], episodes [42], association rules [2], correlations [62], multi-dimensional patterns [32, 39], maximal itemsets [6, 75, 40], closed itemsets [68, 51, 52, 54].

The first algorithm based on the combination of association rule mining with FCA was Close [51], followed by A-Close [52], ChARM [74], Pascal [5], Closet [54], and Titanic [65], each having its own way to exploit the equivalence relation which is hidden in the data. Many algorithms can be found at the Frequent Itemset Mining Implementations Repository.[11]

Beside closed itemsets, other condensed representations have been studied: key sets [5]/free sets [10], non-derivable itemsets [13], δ-free sets, disjunction free sets [12], and k-free sets [58]. Closed itemsets and other condensed representations can be used for defining bases of association rules [66, 53].

Following the initial paper [37] by Lehmann and Wille, several researchers started to analyse the mathematical properties of trilattices, e. g., [7, 8, 9, 18, 24, 71, 72].

[37] and [18] present several ways to project a triadic context to a dyadic one. [64] presents a model for navigating a triadic context by visualising concept lattices of such projections. The idea of deriving dyadic contexts from the triadic one is not new and has been presented by Lehmann and Wille in [38], for instance. However, none of the aforementioned works tackled the problem of developing an algorithm for mining tri-concepts in triadic contexts.

3.3 (Semi-) Automatic Metadata Alignment

The need for aligning different ontologies or schemas have been long noticed by the knowledge engineers and database communities. In ontology alignment the aim is to find correspondences between elements of two different ontologies. There is a lot of previous work on developing such alignments in different contexts, such as knowledge representation, machine learning and schema integration. We first briefly describe some of the existing methods and then describe some specific systems or algorithms using them. For more thorough surveys the interested reader is referred to [57],[61],[76].

[9] `http://www.informatik.uni-trier.de/~ley/db/conf/icfca/`

[10] `http://www.informatik.uni-trier.de/~ley/db/conf/iccs/`

[11] `http://fimi.cs.helsinki.fi/`

Alignment approaches can be classified based on the information types they use. Two types of information can be used to infer mappings between ontologies:

- Ontology-based information
- Instance-based information

Approaches which use the ontologies themselves and not instance data, rely on two types of evidences: the first are the similarity measures between individual concepts of the two ontologies. These similarity measures can be either textual and linguistic based or constraint based. Names or descriptions of concepts are compared in order to assess their similarities. Various similarity measures exist for evaluating the similarity between strings which can be used in this context. Linguistic approaches also rely on auxiliary information such as dictionaries. Constraint based approaches use the constraints on elements to determine the similarity of ontology elements. The second type of evidences that ontology-based approaches utilize are the structure or topology information of the ontologies. The graph representations of the two ontologies are used in these approaches.

Constraints such as foreign keys can be used in instance based approaches. A good heuristic in these approaches is that if the instances of certain classes of two ontologies are similar, these two classes have a higher probability of being mapped to one another. However the mapping problem arises here again, in order to determine the similarity of two objects formulated in terms of different ontologies, first the ontologies need to be mapped. However, since in many applications ontologies are just used to manage annotations and not the real instances, other classifier functions can be used. For example if the instances are documents, information retrieval methods could be applied to measure their similarities. In the following we explain some algorithms and systems which have been devised for metadata alignment.

Anchor-Prompt algorithm [50] is a hybrid method. It takes as input two ontologies and a set of anchor-pairs of related items. These anchor-pairs could either be identified by textual techniques or explicitly by the user. They are then refined based on the structure of the two input ontologies. For each of the ontologies a directed labeled graph is constructed. The nodes in this graph are the concepts, and edges between them are the relationships between these concepts. The paths in the sub-graphs limited by the anchors are analyzed. The concepts which frequently appear in similar position of similar paths are also considered as semantically similar.

Similarity flooding [46] is actually a graph matching algorithm. The main idea in this algorithm is that similarity values can be transferred from a concept to its neighboring concepts. To implement this idea, the two input ontologies are first translated into directed labeled graphs. Then from these two graphs a third labeled graph is constructed. The set of nodes in this graph is the cartesian product of the sets of nodes of the ontology graphs. There is a label l between nodes (x_1, y_1) and (x_2, y_2) if there was both an edge labeled l between x_1 and x_2 , and between y_1 and y_2 in the ontology graphs. Some initial similarity values are assigned to each node, based on the similarity of it corresponding concepts in the two ontology graphs. In each iteration of the algorithm, new similarity values are calculated for each node as a function of similarities of it neighboring nodes. The algorithm terminates after a certain number of steps or when the similarity values converge and do not change more than a predefined threshold.

LSD [20] employs machine learning for schema matching. Different learners are used to exploit different type of information available. A meta-learner is then used to aggregate the results of the previous learners. LSD requires examples of mappings between different ontologies as the training sets for its learners.

We use both types of evidence in our work to induce mappings. Our work can be regarded as exploiting all available approaches and aggregating their innovations.

4 Community Support Architecture

The realization of the Social Semantic Desktop epitomized by NEPOMUK is based on an architecture where components encapsulate well-defined pieces of functionality (cf. Figure 1). In the final state of the project there will exist for each component at least one implementation which provides the functionalities that describe the component. One such component is the RDF store which provides the functionality to store, query and retrieve information in RDF format. In the figure it is depicted in the lower left section. Furthermore, this diagram also gives an overview of how the components' are assigned to different workpackages

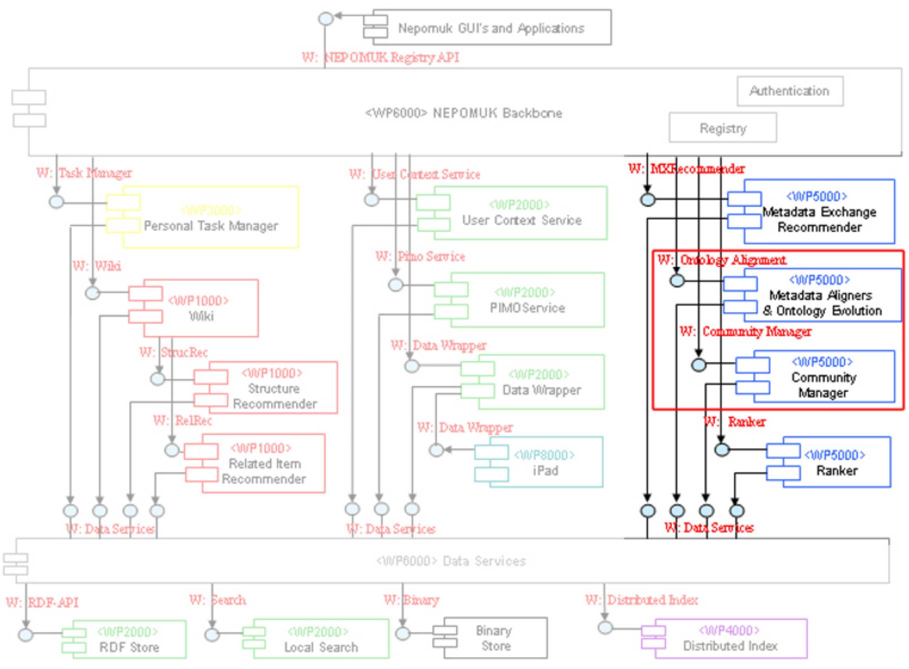


Figure 1: NEPOMUK Components

The diagram also shows how the components Community Manager and Metadata Aligner[12] of this deliverable are integrated into the NEPOMUK component architecture. It can be seen that both components access the RDF Store via the Data Services component which also allows access of the Distributed Index or the Binary Store. This ensures that all data is accessed in a generic way and thus it eases the exchange of components like the RDF Store. As said, the idea behind this principle is that implementations of all components should be easily exchangeable. The functionality which is provided by Community Manager and Metadata Aligner is exposed to the other components and applications of the Social Semantic Desktop via the Web Server shown above the components. Through the use of SOAP requests it facilitates the implementation independent communication across components and native desktop applications.

A more architecture driven diagram shows Figure 2. There the communication across components is ensured by the Local Service Interface which by its Service Registry and Matchmaker allows components to find implementations of other components.

The community support architecture of WP5000 is conceptually integrated into

[12] The Metadata Aligner is depicted there as *Metadata Aligners & Ontology Evolution*.

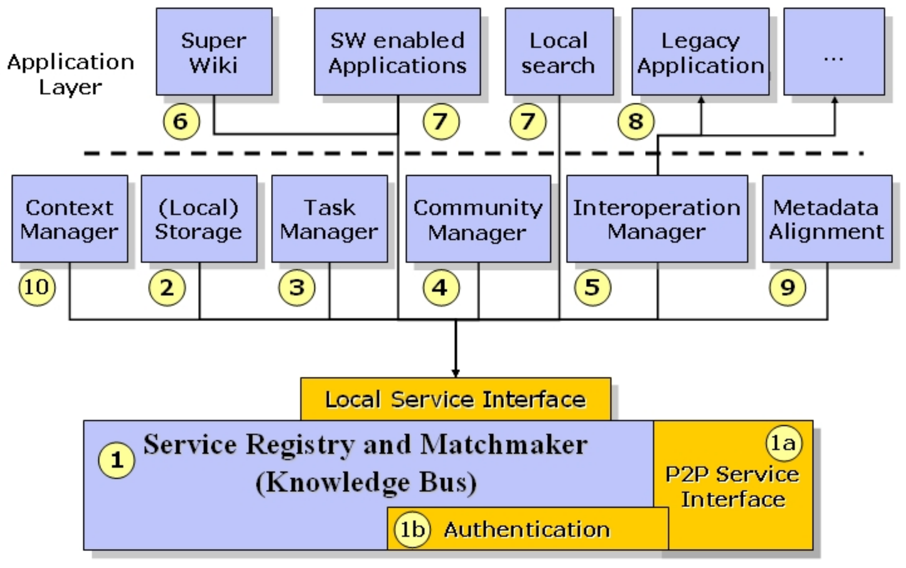


Figure 2: Community Manager (4) and Metadata Alignment (9) in the NEPOMUK architecture.

the NEPOMUK component architecture as components Community Manager, Metadata Aligner, Metadata Recommender and Social Ranker. Their interaction and affiliation to the deliverables D5.1, D5.2 and D5.3 of WP5000 is depicted in Figure 3.

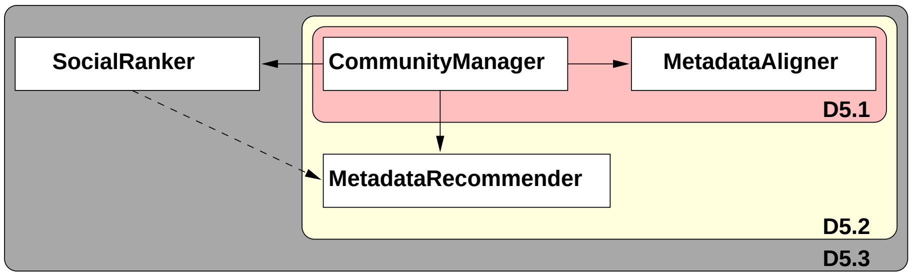


Figure 3: Interaction of components from WP5000 and their affiliation to the deliverables.

In the picture a straight line shows that a component gives input to other components of WP5000, a necessary condition for them to work, while the dotted line shows optional input the Metadata Recommender might receive from the Social Ranker. The inner box depicts that the components belonging to deliverable D5.1 are the Community Manager and the Metadata Aligner. As it can be seen, a central role for all components in WP5000 is played by the Community Manager whose results are used as input for the three other components.

Summarized, this deliverable provides community detection and metadata alignment functionality which can be integrated into other components. Although they are not applications on their own, they form an infrastructure which can be adopted and used by other applications.

4.1 Community Manager

In deliverable D5.1 the functionality provided by the component Community Manager consists of detecting and labeling communities of users. In general, the component analyzes the folksonomy structure 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 what concrete methods the component provides, how it is integrated into the NEPOMUK architecture, and what its dependencies to other components are.

4.1.1 Service Description

In Table 1 the WSDL document describing the functionality of the Community Manager—how to call it, which data structures to send and to expect—is shown. The WSDL document is the "contract" which components have to agree upon when they want to interact with the Community Manager. It defines the structure of the input and output messages, the offered methods and a concrete endpoint of a Community Manager implementation which is willing to fulfill this contract.

The service description starts with the obligatory namespace declarations and defines in lines 9 to 39 the XML Schema for the types underlying the messages used to communicate with Community Manager. These XML Schema definitions are utilized to automatically generate Java classes and to serialize those classes in an XML representation complying to that schema. The serialization is used to compose the SOAP messages that are sent between the components.

All of the types Tag, User and Community are described by an URI and a human readable and identifiable string. In the case of User this is the user's name and for Tag, the tag itself. Furthermore, the Community type contains two lists: a list of users, which represent the community and a list of tags which label the community with its main topics.

Proceeding with the WSDL description, it contains in lines 50 to 55 the specification on how to call the Community Manager. A call to "GetCommunity" requires as input (lines 41–44) a user name or a tag. Hence, it is possible to find a community of users related to a specific topic or otherwise ask for users around a given user. The resulting output is a community as described in the type definitions seen before.

Finally, lines 57 to 70 define a binding which allows components to access a Community Manager implementation via SOAP requests. The concrete implementation is reachable on the local desktop via HTTP at `http://localhost:8181/soap/CommunityManager`.

4.1.2 Architecture

To accomplish the community detection and labeling tasks we implemented the FolkRank algorithm described in Section 5.3. Since analysis is hardly possible in a distributed manner and might influence the performance of the individual peers, we decided to distribute the work necessary for community detection. We divided the data gathering part from the analysis part such that each peer in the distributed network collects relevant data from its RDF repository and sends it to a special peer which we call "FolkPeer". It is the task of this latter then to analyze the data, detect communities and label them,

Table 1: WSDL describing the functionality of the Community Manager.

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions name="CommunityManager"
  targetNamespace="http://www.semanticdesktop.org/wsdl/2007/02/09/CommunityManager.wsdl"
  xmlns:tns="http://www.semanticdesktop.org/wsdl/2007/02/09/CommunityManager.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xmlschema="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

  <types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.semanticdesktop.org/wsdl/2007/02/09/CommunityManager.wsdl">

      <xs:complexType name="User"><xs:all>
        <xs:element name="uri" type="xs:anyURI" />
        <xs:element name="name" type="xs:string" />
      </xs:all></xs:complexType>

      <xs:complexType name="Tag"><xs:all>
        <xs:element name="uri" type="xs:anyURI" />
        <xs:element name="name" type="xs:string" />
      </xs:all></xs:complexType>

      <xs:complexType name="UserList"><xs:sequence>
        <xs:element name="user" minOccurs="0" maxOccurs="unbounded" type="tns:User" />
      </xs:sequence></xs:complexType>

      <xs:complexType name="TagList"><xs:sequence>
        <xs:element name="tag" minOccurs="0" maxOccurs="unbounded" type="tns:Tag" />
      </xs:sequence></xs:complexType>

      <xs:complexType name="Community"><xs:all>
        <xs:element name="uri" type="xs:anyURI" />
        <xs:element name="name" type="xs:string" />
        <xs:element name="users" type="tns:UserList" />
        <xs:element name="tags" type="tns:TagList" />
      </xs:all></xs:complexType>

    </xs:schema>
  </types>

  <message name="GetCommunityRequest">
    <part name="user" type="tns:User" />
    <part name="tag"  type="tns:Tag" />
  </message>

  <message name="GetCommunityResponse">
    <part name="community" type="tns:Community" />
  </message>

  <portType name="CommunityManagerApi">
    <operation name="GetCommunity">
      <input message="tns:GetCommunityRequest" />
      <output message="tns:GetCommunityResponse" />
    </operation>
  </portType>

  <binding name="CommunityManagerSoapBinding" type="tns:CommunityManagerApi">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetCommunity">
      <soap:operation soapAction=""/>
      <input><soap:body use="literal"/></input>
      <output><soap:body use="literal"/></output>
    </operation>
  </binding>

  <service name="CommunityManager">
    <port name="CommunityManagerPort" binding="tns:CommunityManagerSoapBinding">
      <soap:address location="http://localhost:8181/soap/CommunityManager"/>
    </port>
  </service>

</definitions>
```

and make available the results to all the peers in the network. We decided to call this peer "FolkPeer", since the underlying structure we are using for the analysis is a folksonomy and this peer finds communities of users inside a folksonomy. It is this division of tasks that makes efficient mining of communities in the NEPOMUK P2P network possible.

The distribution of tasks between Community Manager and FolkPeer is thus:

- CommunityManager
  - runs on every peer in the network,
  - receives requests of components or desktop applications for community information and forwards them to the FolkPeer,
  - selects and collects data from the RDF repository and sends it to the FolkPeer,
  - reacts upon changes in the RDF repository.
- FolkPeer
  - runs on a distinguished peer in the network,
  - upon request, sends community information to peers,
  - receives data from the peers and collects it in a data warehouse,
  - analyzes the data and stores the results.

4.1.3 Gathering Data

In order to collect the necessary data for the analysis on the FolkPeer, the Community Manager of a peer needs to extract relevant information from the users RDF repository and send it to the FolkPeer. The tagging data is represented in the RDF repository by the vocabulary of the NEPOMUK Annotation Ontology (NAO).[13]

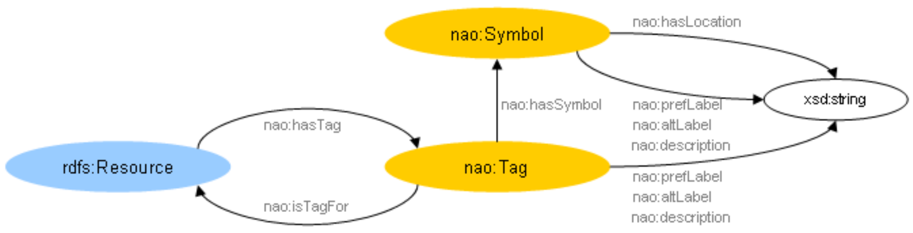


Figure 4: Tagging as modeled in NAO.

Tagging is modeled in NAO with the "hasTag" relation, as can be seen in Figure 4. It relates a resource to another resource of class "Tag" which has attached a user readable string by the properties "prefLabel" or "altLabel". The relevant (resource, tag) tuples are received from the RDF repository by a SPARQL query (cf. Figure 5) and together with the user name sent serialized as XML to the FolkPeer. The FolkPeer then updates its data warehouse and schedules a recomputation of the communities.

[13] The current working draft of NAO is available at `http://svn.nepomuk.semanticdesktop.org/repos/trunk/taskforce/TF-Ont/draft/NAO/NAO.html`.

```
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 5: Example SPARQL query used to gather tagging data.

4.1.4 Relation to other components

In a system like NEPOMUK where tasks are distributed among components, naturally, also the Community Manager depends on services provided by other workpackages' components. Hence, we will here provide an overview on the components necessary for the Community Manager to run.

Middleware. The middleware provides essential functionalities to register, detect and call services, allows to easily expose features via SOAP, and provides human interaction with components via HTML interfaces. All those features are intensively used by the Community Manager. Upon startup it registers itself to the middleware, such that other components can find and use it; when a request to the RDF repository is necessary, the middleware delivers an implementation of it; the functionality of the small application Community Client is exposed to the user via the middleware's webserver; and finally, the SOAP connector of the Community Manager is provided by the middleware. This allows applications and components running outside the middleware to access the Community Manager via HTTP.

RDF Repository. The RDF Repository plays the central role in gathering data for the community analysis. It stores all the information relevant for the algorithms to detect communities. Hence, the Community Manager uses it to collect those pieces of data. This is done by querying the repository with the SPARQL interface it provides. Furthermore, all configuration parameters for the Community Manager are stored in the RDF Repository's "config" area.

PIMOService. The PIMOService (amongst other things) provides methods to identify the user (i. e., get the user name) and to generate unique identifiers for resources. Thus it helps Community Manager to distinguish users, resources and tags.

Components delivering data as input for analysis. Components like the Wiki, the Personal Task Manager or the Data Wrapper are one of the main providers of metadata stored in the RDF Repository. Among them, user annotated resources necessary for the analysis are widely available. The Wiki allows users to tag pages, the Task Manager to assign keywords to tasks, meetings, etc., and the Data Wrapper might gather the users bookmarks from an existing folksonomy service or extract keywords from BibTeX files.

4.1.5 Invocation Example

The Java code snippet in Table 2 shows how to invoke the Community Manager with Java on the OSGi platform. The method getCommunityManager requests in line 30 a reference to an implementation of the Community Manager interface from the bundle context. If an implementation is found, it is casted to the interface CommunityManager and returned (line 35). Such a reference is in line 18 used to request the community for the given user and subsequently print all users belonging to the community (lines 23-25).

Table 2: Example Java Code invoking Community Manager.

```
 1  public class Example {
 2
 3      private BundleContext bc;
 4
 5      public Example(BundleContext context) {
 6          this.bc = context;
 7      }
 8
 9      public void printCommunityOfUser(String user) throws Exception {
10          CommunityManager service = getCommunityManager();
11
12          User user = new User();
13          user.setName(requUser);
14
15          /*
16           * get Community Manager instance
17           */
18          Community community = service.getCommunity(user);
19
20                      /*
21           * print names of users in community of user
22           */
23          for (User comUser: community.getUsers().getUser()) {
24              System.out.println(comUser.getName());
25          }
26
27      }
28
29      private CommunityManager getCommunityManager() throws Exception {
30          ServiceReference sr =
31                                          bc.getServiceReference(CommunityManager.class.getName());
32          if (sr == null) {
33              log.fatal("Could not find CommunityManager - aborting.");
34              throw new Exception("No CommunityManager found");
35          }
36          return (CommunityManager) bc.getService(sr);
37      }
38  }
```

4.2 Metadata Alignment

The Metadata Alignment component is responsible for generating a mapping between two given ontologies. It provides this functionality by aggregating evidences based on different measures. In this section we give an overview of the methods this component provides, its integration into the NEPOMUK architecture and its relation to other NEPOMUK components.

4.2.1 Service Description

The WSDL describing the functionality of the Metadata Alignment is shown in Table 3

Similar to the CommunityManager wsdl file, the service description starts with the obligatory namespace declarations and defines in lines 14 to 55 the XML Schema for the types underlying the messages used to communicate with Metadata alignment. For example the type “CreateGenericAlignment” which is part of the “CreatGenericAlignmentRequest” message is a sequence of two Models as defined in http://model.rdf2go.ontoware.org both of which occur

Table 3: WSDL describing the functionality of Metadata Alignment.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    targetNamespace="http://nepomuk.semanticdesktop.org/comp/PhaseAlignment.wsdl"
    xmlns:tns="http://nepomuk.semanticdesktop.org/comp/PhaseAlignment.wsdl"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:ns1="http://model.rdf2go.ontoware.org"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">

<wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://nepomuk.semanticdesktop.org/comp/PhaseAlignment.wsdl">
        <xsd:element name="createGenericAlignment">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="in0" nillable="true"
                        type="ns1:Model"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="in1" nillable="true"
                        type="ns1:Model"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="createGenericAlignmentResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="out" nillable="true"
                        type="ns1:Model"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="createPIMOAlignment">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="in0" nillable="true"
                        type="xsd:string"/>
                    <xsd:element maxOccurs="1" minOccurs="1" name="in1" nillable="true"
                       type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="createPIMOAlignmentResponse">
            <xsd:complexType>
                <xsd:sequence>
                    <xsd:element maxOccurs="1" minOccurs="1" name="out" nillable="true"
                        type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:schema>
</wsdl:types>

<wsdl:message name="createPIMOAlignmentRequest">
    <wsdl:part name="parameters" element="tns:createPIMOAlignment"/>
</wsdl:message>
<wsdl:message name="createGenericAlignmentRequest">
    <wsdl:part name="parameters" element="tns:createGenericAlignment"/>
</wsdl:message>
<wsdl:message name="createPIMOAlignmentResponse">
    <wsdl:part name="parameters" element="tns:createPIMOAlignmentResponse"/>
</wsdl:message>
<wsdl:message name="createGenericAlignmentResponse">
    <wsdl:part name="parameters" element="tns:createGenericAlignmentResponse"/>
</wsdl:message>

<wsdl:portType name="PhaseAlignmentPortType">
    <wsdl:operation name="createGenericAlignment">
        <wsdl:input name="createGenericAlignmentRequest"
            message="tns:createGenericAlignmentRequest"/>
        <wsdl:output name="createGenericAlignmentResponse"
            message="tns:createGenericAlignmentResponse"/>
    </wsdl:operation>
    <wsdl:operation name="createPIMOAlignment">
        <wsdl:input name="createPIMOAlignmentRequest"
            message="tns:createPIMOAlignmentRequest"/>
        <wsdl:output name="createPIMOAlignmentResponse"
            message="tns:createPIMOAlignmentResponse"/>
        </wsdl:operation>
</wsdl:portType>
```

Table 4: WSDL describing the functionality of Metadata Alignment (cont.).

```
 84 <wsdl:binding name="PhaseAlignmentHttpBinding" type="tns:PhaseAlignmentPortType">
 85     <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
 86             <wsdl:operation name="createGenericAlignment">
 87             <wsdlsoap:operation soapAction=""/>
 88                 <wsdl:input name="createGenericAlignmentRequest">
 89                         <wsdlsoap:body use="literal"/>
 90                 </wsdl:input>
 91                 <wsdl:output name="createGenericAlignmentResponse">
 92                         <wsdlsoap:body use="literal"/>
 93                 </wsdl:output>
 94             </wsdl:operation>
 95             <wsdl:operation name="createPIMOAlignment">
 96             <wsdlsoap:operation soapAction=""/>
 97                 <wsdl:input name="createPIMOAlignmentRequest">
 98                         <wsdlsoap:body use="literal"/>
 99                 </wsdl:input>
100                 <wsdl:output name="createPIMOAlignmentResponse">
101                         <wsdlsoap:body use="literal"/>
102                 </wsdl:output>
103                 </wsdl:operation>
104     </wsdl:binding>
105
106 <wsdl:service name="PhaseAlignment">
107     <wsdl:port name="PhaseAlignmentHttpPort" binding="tns:PhaseAlignmentHttpBinding">
108         <wsdlsoap:address location="http://localhost:8181/soap/PhaseAlignment"/>
109         </wsdl:port>
110 </wsdl:service>
111
112 </wsdl:definitions>
```

exactly once.

Lines 70 to 83 describe specifications of calling the metadata aligner. For example a call to “createGenericAlignment” requires as input a “createGenericAlignmentRequest” as defined in lines 60 to 62 and outputs “createGenericAlignmentRequest” as defined in lines 66 to 68. Therefore it is possible to generate an alignment between two ontologies by calling “createGenericAlignment”.

Finally, lines starting at 84 define a binding which allows components to access a Metadata alignment implementation via SOAP requests. The concrete implementation is reachable on the local desktop via HTTP at `http://localhost:8181/soap/PhaseAlignment`.

4.2.2 Architecture of Metadata Alignment

Since existing ontology languages like RDF/S or OWL do not fulfill all requirements given on a Semantic Desktop, a new language, PIMO is used in NEPOMUK. The language contains a core upper ontology, defining basic classes for things, concepts, resources, persons, etc. and also stops at these basic entities. Extending the ontology definitions of classes and relations is possible by PIMO-domain ontologies. The core application area of the PIMO-language is to allow individual persons to express their own mental models in a structured way, the different mental models can then be integrated based on matching algorithms or on domain ontologies. Based on the core upper ontology elements, each user can extend his personal mental model in an open manner. Currently the metadata alignment component does not use the PIMOService, but in future it should use it to access PIMO ontologies stored by the users.

The work in metadata alignment is divided between three main modules. The Ontology Adapter module basically allows the component to access ontologies represented in different languages. As described in section 6, two ontologies

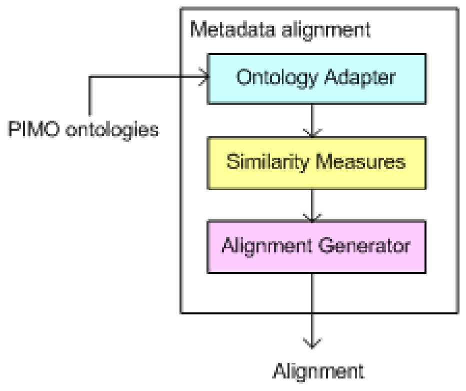


Figure 6: Architecture of Metadata Alignment

are aligned by gathering similarity measures based on different evidences and aggregating them. Thus the second module of the metadata alignment provides different similarity measures. The third module is the Alignment Generator, which takes as input the available Similarity Measures and aggregates them in a specific manner. The output is a mapping between the two input ontologies expressed as mapping ontology facts. A simple view of the metadata alignment component and its three modules and their relationships is shown in figure 6.

4.2.3 Relation to other components

PIMOService. The PimoService component stores personal information models and allows qualified access to the personal information model. Our metadata alignment component should use this component to access PIMOs.

Middleware. The NEPOMUK middleware manages the registration and calling of all services provided in NEPOMUK. The metadata aligner registers itself to the middleware at startup enabling other components to find and use it. Upon a request to PimoService, an implementation of this service will be provided by middleware.

5 Community Detection

To accomplish the community detection and labeling task provided by the CommunityManager and its complement, the FolkPeer, we describe here how the folksonomy structure of the NEPOMUK network can be analyzed to detect communities of users within the network. We start by introducing social resource sharing systems and formalize the underlying datastructure called folksonomy. Then we present the FolkRank algorithm and its application to community detection, followed by a description of the Trias algorithm for mining frequent closed tri-sets of a folksonomy.[14]

5.1 Social Resource Sharing and Folksonomies

Social resource sharing systems are web-based systems that allow users to upload their resources, and to label them with arbitrary words, so-called tags. The systems can be distinguished according to what kind of resources are supported. Flickr, for instance, allows the sharing of photos, del.icio.us the sharing of bookmarks, CiteULike[15] and Connotea[16] the sharing of bibliographic references, and 43Things[17] even the sharing of goals in private life. BibSonomy,[18] allows to share simultaneously bookmarks and BibTeX entries.

In their core, these 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 had uploaded, together with the tags he had 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.

A Formal Model for Folksonomies A folksonomy describes the users, resources, and tags, and the user-based assignment of tags to resources. We present here a formal definition of folksonomies.

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

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 ith dimension.

Users are typically described by their user ID, and tags may be arbitrary strings. What is considered as a resource depends on the type of system. For instance, in del.icio.us, the resources are URLs, and in Flickr, the resources

[14] The FolkRank has first been published in [30], TRIAS in [31].
[15] `http://www.citeulike.org/`
[16] `http://www.connotea.org/`
[17] `http://www.43things.com/`
[18] `http://www.bibsonomy.org`

are pictures. On the social semantic desktop resources range from files (documents, music, pictures) over e-mails and wiki pages to arbitrary resources which can be referenced by URIs like URLs, document authors, other users, or project tasks.

In the described analysis approach, we do not make use of the subtag/supertag relation for sake of simplicity. I. e., $\prec = \emptyset$, and we will simply note a folksonomy as a quadruple $\mathbb{F} := (U, T, R, Y)$. This structure is known in Formal Concept Analysis [70, 26] as a triadic context [38, 64]. An equivalent view on folksonomy data is that of a tripartite (undirected) hypergraph $G = (V, E)$, where $V = U \dot\cup T \dot\cup R$ is the set of nodes, and $E = \{\{u, t, r\} \mid (u, t, r) \in Y\}$ is the set of hyperedges.

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

5.2 Adapting PageRank for Folksonomies

A community of users surrounding a user or relevant to a given topic can technically be described as a set of users which 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. Similiarly, 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 neccessary for the relevant application. Thus, applications can decide how many users of the community they use—the task manager might highlight only few (five to ten) relevant persons while other applications need more. Nevertheless, the Community Manager does not return all users of the network, since FolkRank gives a natural way to cut the set of users into users which belong to the community and which do not belong to it. We will see at the end of Section 5.3 how this is possible. 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 the user several communities he is in and along the way support different aspects of his interest.

In the previous Section 5.1 we have shown that a folksonomy induces a graph structure. We will exploit that structure for ranking in this section. Our FolkRank algorithm is inspired by the seminal PageRank algorithm [11]. The PageRank weight-spreading approach cannot be applied directly on folksonomies because of the different nature of folksonomies compared to the web graph (undirected triadic hyperedges instead of directed binary edges). In the following we discuss how to overcome this problem.

Adaptation of PageRank

We implement the weight-spreading ranking scheme on folksonomies in two steps. First, we transform the hypergraph between the sets of users, tags, and resources into an undirected, weighted, tripartite graph. On this graph, we apply a version of PageRank that takes into account the edge weights.

Converting the Folksonomy into an Undirected Graph. First we convert the folksonomy $\mathbb{F} = (U, T, R, Y)$ into an undirected tripartite graph $\mathbb{G}_{\mathbb{F}} = (V, E)$ as follows.

1. The set V of nodes of the graph consists of the disjoint union of the sets of tags, users and resources: $V = U \dot{\cup} T \dot{\cup} R$. (The tripartite structure of the graph can be exploited later for an efficient storage of the – sparse – adjacency matrix and the implementation of the weight-spreading iteration in the FolkRank algorithm.)
2. All co-occurrences of tags and users, users and resources, tags and resources become undirected, weighted edges between the respective nodes: $E = \{\{u, t\}, \{t, r\}, \{u, r\} \mid (u, t, r) \in Y\}$, with each edge $\{u, t\}$ being weighted with $|\{r \in R : (u, t, r) \in Y\}|$, each edge $\{t, r\}$ with $|\{u \in U : (u, t, r) \in Y\}|$, and each edge $\{u, r\}$ with $|\{t \in T : (u, t, r) \in Y\}|$.

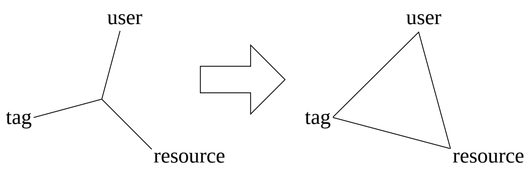


Figure 7: Converting a hyperedge into three edges.

Folksonomy-Adapted Pagerank. The original formulation of PageRank [11] reflects the idea that a page is important if there many pages linking to it, and if those pages are important themselves. The distribution of weights can thus be described as the fixed point of a weight passing scheme on the web graph. This idea was extended in a similar fashion to bipartite subgraphs of the web in HITS [33] and to n-ary directed graphs in [73]. We employ the same underlying principle for our ranking scheme in folksonomies. The basic notion is that a resource which is tagged with important tags by important users becomes important itself. The same holds, symmetrically, for tags and users. Thus we have a graph of vertices which are mutually reinforcing each other by spreading their weights.

Like PageRank, we employ the random surfer model, a notion of importance for web pages that is based on the idea that an idealized random web surfer normally follows hyperlinks, but from time to time randomly jumps to a new webpage without following a link. This results in the following definition of the rank of the vertices of the graph the entries in the fixed point $\vec{w}$ of the weight spreading computation $\vec{w} \leftarrow dA\vec{w} + (1-d)\vec{p}$, where $\vec{w}$ is a weight vector with one entry for each web page, A is the row-stochastic[19] version of the adjacency matrix of the graph $G_{\mathbb{F}}$ defined above, $\vec{p}$ is the random surfer component, and $d \in [0, 1]$ is determining the influence of $\vec{p}$. In the original PageRank, $\vec{p}$ is used to outweigh the loss of weight on web pages without outgoing links. Usually, one will choose $\vec{p} = \mathbf{1}$, i.e., the vector composed by 1's. In order to compute personalized PageRanks, however, $\vec{p}$ can be used to express user preferences by giving a higher weight to the components which represent the user's preferred web pages.

Formally, we spread the weight as follows:

$$\vec{w} \leftarrow \alpha\vec{w} + \beta A\vec{w} + \gamma\vec{p} \tag{1}$$

where A is the row-stochastic version of the adjacency matrix of $\mathbb{G}_{\mathbb{F}}$, $\vec{p}$ is a preference vector, $\alpha, \beta, \gamma \in [0, 1]$ are constants with $\alpha + \beta + \gamma = 1$. The constant

[19] I. e., each row of the matrix is normalized to 1 in the 1-norm.

α is intended to regulate the speed of convergence, while the proportion between β and γ controls the influence of the preference vector.

We call the iteration according to Equation 1 – until convergence is achieved – the Adapted PageRank algorithm. Note that, if $||\vec{w}||_1 = ||\vec{p}||_1$ holds,[20] the sum of the weights in the system will remain constant. The influence of different settings of the parameters α, β, and γ is discussed below.

As the graph $G_{\mathbb{F}}$ is undirected, part 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) to a ranking that is simply based on edge degrees, as we will see now. The reason for applying the more expensive PageRank approach nonetheless is that its random surfer vector allows for topic-specific ranking, as we will discuss in the next section.

Evaluation

In order to evaluate our retrieval technique, and because NEPOMUK data was not yet available, we have analyzed the popular social bookmarking sytem del.icio.us, which is a server-based system with a simple-to-use interface that allows users to organize and share bookmarks on the internet. It is able to store in addition to the URL a description, an extended description, and tags (i. e., arbitrary labels).

For our experiments, we used data from the del.ico.us system we collected in the following way. Initially we used `wget` starting from the top page of del.ico.us to obtain nearly 6900 users and 700 tags as a starting set. Out of this dataset we extracted all users and resources (i. e., del.icio.us' MD5-hashed urls). From July 27 to 30, 2005, we downloaded in a recursive manner user pages to get new resources, and resource pages to get new users. Furthermore we monitored the del.icio.us start page to gather additional users and resources. This way we collected a list of several thousand usernames which we used for accessing the first 10000 resources each user had tagged. From the collected data we finally took the user files to extract resources, tags, dates, descriptions, extended descriptions, and the corresponding username.

We obtained a core folksonomy with $|U| = 75,242$ users, $|T| = 533,191$ tags and $|R| = 3,158,297$ resources, related by in total $|Y| = 17,362,212$ TAS.[21] After inserting this dataset into a MySQL database, we were able to perform our evaluations, as described in the following sections.

As expected, the tagging behavior in del.icio.us shows a power law distribution, see Figure 8. This figure presents the percentage of tags, users, and resources, respectively, which occur in a given number of TAS. For instance, the rightmost '+' indicates that a fraction of $2.19 \cdot 10^{-6}$ of all tags (i. e. one tag) occurs 415950 times—in this case it is the empty tag. The next '+' shows that one tag ("web") occurs 238891 times, and so on. One observes that while the tags follow a power law distribution very strictly, the plot for users and resources levels off for small numbers of occurrences. Based on this observation, we estimate to have crawled most of the tags, while many users and resources are still missing from the dataset. A probable reason is that many users only try posting a single resource, often without entering any tags (the empty tag is the most frequent one in the dataset), before they decide not to use the system anymore. These users and resources are very unlikely to be connected with others at all (and they only appear for a short period on the del.icio.us start page), so that they are not included in our crawl.

Results for Adapted PageRank

We have evaluated the Adapted PageRank on the del.ico.us dataset described in the previous section. As there exists no 'gold standard ranking' on these

[20] . . . and if there are no rank sinks – but this holds trivially in our graph $G_{\mathbb{F}}$. [21] 4,313 users additionally organised 113,562 of the tags with 6,527 so-called *bundles*. The bundles will not be discussed here; they can be interpreted as one level of the $\prec$ relation.

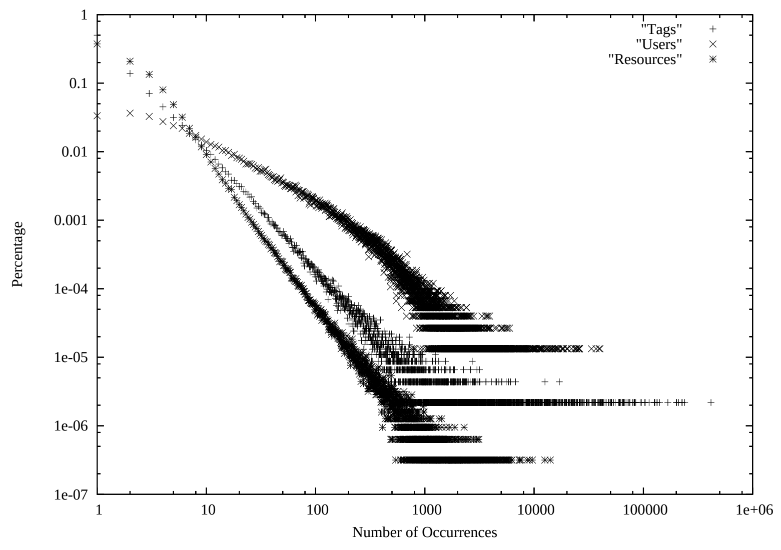


Figure 8: Number of TAS occurrences for tags, users, resources in del.icio.us

data, we evaluate our results empirically.

First, we studied the speed of convergence. We let $\vec{p} := \mathbf{1}$ (the vector having 1 in all components), and varied the parameter settings. In all settings, we discovered that $\alpha \neq 0$ slows down the convergence rate. For instance, for $\alpha = 0.35, \beta = 0.65, \gamma = 0$, 411 iterations were needed, while $\alpha = 0, \beta = 1, \gamma = 0$ returned the same result in only 320 iterations. It turns out that using γ as a damping factor by spreading equal weight to each node in each iteration speeds up the convergence considerably by a factory of approximately 10 (e. g., 39 iterations for $\alpha = 0$, $\beta = 0.85$, $\gamma = 0.15$).

Table 5 shows the result of the adapted PageRank algorithm for the 20 most important tags, users and resources computed with the parameters $\alpha = 0.35, \beta = 0.65, \gamma = 0$ (which equals the result for $\alpha = 0, \beta = 1, \gamma = 0$). Tags get the highest ranks, followed by the users, and the resources. Therefore, we present their rankings in separate lists.

As we can see from the tag table, the most important tag is "system:unfiled" which is used to indicate that a user did not assign any tag to a resource. It is followed by "web", "blog", "design" etc. This corresponds more or less to the rank of the tags given by the overall tag count in the dataset. The reason is that the graph $G_{\mathbb{F}}$ is undirected. We face thus the problem that, in the Adapted PageRank algorithm, weights that flow in one direction of an edge will basically 'swash back' along the same edge in the next iteration. Therefore the resulting is very similar (although not equal!) to a ranking based on counting edge degrees.

The resource ranking shows that Web 2.0 web sites like Slashdot, Wikipedia, Flickr, and a del.icio.us related blog appear in top positions. This is not surprising, as early users of del.ico.us are likely to be interested in Web 2.0 in general. This ranking correlates also strongly with a ranking based on edge counts.

The results for the top users are of more interest as different kinds of users appear. As all top users have more than 6000 bookmarks; "notmuch" has a large amount of tags, while the tag count of "fritz" is considerably smaller.

Table 5: Folksonomy Adapted PageRank applied without preferences (called *baseline*)

Tag	ad. PageRank
system:unfiled	0,0078404
web	0,0044031
blog	0,0042003
design	0,0041828
software	0,0038904
music	0,0037273
programming	0,0037100
css	0,0030766
reference	0,0026019
linux	0,0024779
tools	0,0024147
news	0,0023611
art	0,0023358
blogs	0,0021035
politics	0,0019371
java	0,0018757
javascript	0,0017610
mac	0,0017252
games	0,0015801
photography	0,0015469
fun	0,0015296

User	ad. PageRank
shankar	0,0007389
notmuch	0,0007379
fritz	0,0006796
ubi.quito.us	0,0006171
weev	0,0005044
kof2002	0,0004885
ukquake	0,0004844
gearhead	0,0004820
angusf	0,0004797
johncollins	0,0004668
mshook	0,0004556
frizzlebiscuit	0,0004543
rafaspol	0,0004535
xiombarg	0,0004520
tidesonar02	0,0004355
cyrusnews	0,0003829
bldurling	0,0003727
onpause_tv_anytime	0,0003600
cataracte	0,0003462
triple_entendre	0,0003419
kayodeok	0,0003407

URL	ad. PageRank
`http://slashdot.org/`	0,0002613
`http://pchere.blogspot.com/2005/02/absolutely-delicious-complete-tool.html`	0,0002320
`http://script.aculo.us/`	0,0001770
`http://www.adaptivepath.com/publications/essays/archives/000385.php`	0,0001654
`http://johnvey.com/features/deliciousdirector/`	0,0001593
`http://en.wikipedia.org/wiki/Main_Page`	0,0001407
`http://www.flickr.com/`	0,0001376
`http://www.goodfonts.org/`	0,0001349
`http://www.43folders.com/`	0,0001160
`http://www.csszengarden.com/`	0,0001149
`http://wellstyled.com/tools/colorscheme2/index-en.html`	0,0001108
`http://pro.html.it/esempio/nifty/`	0,0001070
`http://www.alistapart.com/`	0,0001059
`http://postsecret.blogspot.com/`	0,0001058
`http://www.beelerspace.com/index.php?p=890`	0,0001035
`http://www.techsupportalert.com/best_46_free_utilities.htm`	0,0001034
`http://www.alvit.de/web-dev/`	0,0001020
`http://www.technorati.com/`	0,0001015
`http://www.lifehacker.com/`	0,0001009
`http://www.lucazappa.com/brilliantMaker/buttonImage.php`	0,0000992
`http://www.engadget.com/`	0,0000984

To see how good the topic-specific ranking by Adapted PageRank works, we combined it with term frequency, a standard information retrieval weighting scheme. To this end, we downloaded all 3 million web pages referred to by a URL in our dataset. From these, we considered all plain text and html web pages, which left 2.834.801 documents. We converted all web pages into ASCII and computed an inverted index. To search for a term as in a search engine, we retrieved all pages containing the search term and ranked them by tf$(t) \cdot \vec{w}[v]$ where tf(t) is the term frequency of search term t in page v, and $\vec{w}[v]$ is the Adapted PageRank weight of v.

Although this is a rather straightforward combination of two successful retrieval techniques, our experiments with different topic-specific queries indicate that this adaptation of PageRank does not work very well. For instance, for the search term "football", the del.icio.us homepage showed up as the first result. Indeed, most of the highly ranked pages have nothing to do with football.

Other search terms provided similar results. Apparently, the overall structure of the—undirected—graph overrules the influence of the preference vector. In the next section, we discuss how to overcome this problem.

5.3 FolkRank—Community Detection in Folksonomies

In order to reasonably focus the ranking around the topics or users defined in the preference vector, we have developed a differential approach, which compares the resulting rankings with and without preference vector. This resulted in our new FolkRank algorithm.

The FolkRank Algorithm

The FolkRank algorithm computes a topic-specific ranking in a folksonomy as follows:

1. The preference vector $\vec{p}$ is used to determine the topic. It may have any distribution of weights, as long as $||\vec{w}||_1 = ||\vec{p}||_1$ holds. Typically a single entry or a small set of entries is set to a high value, and the remaining weight is equally distributed over the other entries. Since the structure of folksonomies is symmetric, we can define a topic by assigning a high value to either one or more tags and/or one or more users and/or one or more resources.
2. Let $\vec{w_0}$ be the fixed point from Equation (1) with $\beta = 1$.
3. Let $\vec{w_1}$ be the fixed point from Equation (1) with $\beta < 1$.
4. $\vec{w} := \vec{w_1} - \vec{w_0}$ is the final weight vector.

Thus, we compute the winners and losers of the mutual reinforcement of resources when a user preference 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.

Whereas the Adapted PageRank provides one global ranking, independent of any preferences, FolkRank provides one topic-specific ranking for each given preference vector. Note that a topic can be defined in the preference vector not only by assigning higher weights to specific tags, but also to specific resources and users. These three dimensions can even be combined in a mixed vector. Similarly, the ranking is not restricted to resources, it may as well be applied to tags and to users. We will show below that indeed the rankings on all three dimensions provide interesting insights.

Comparing FolkRank with Adapted PageRank

To analyse the proposed FolkRank algorithm, we generated rankings for several topics, and compared them with the ones obtained from Adapted PageRank. We will here discuss one set of ranking results for the tag "boomerang". Our other experiments all provided similar results.

The leftmost part of Table 6 contains the ranked list of tags according to their weights from the Adapted PageRank by using the parameters $\alpha = 0.2, \beta = 0.5, \gamma = 0.3$, and 5 as a weight for the tag "boomerang" in the preference vector $\vec{p}$, while the other elements were given a weight of 0. As expected, the tag "boomerang" holds the first position while tags like "shop" or "wood" which are related are also under the Top 20. The tags "software", "java", "programming" or "web", however, are on positions 4 to 7, but have nothing to do with "boomerang". The only reason for their showing up is that they are frequently used in del.icio.us (cf. Table 5). The second column from the left in Table 6 contains the results of our FolkRank algorithm, again for the tag "boomerang". Intuitively, this ranking is better, as the globally frequent words disappear and related words like "wood" and "construction" are ranked higher.

Table 6: Ranking results for the tag “boomerang” (two left at top: Adapted PageRank and FolkRank for tags, middle: FolkRank for URLs) and for the user “schm4704” (two right at top: Adapted PageRank and FolkRank for tags, bottom: FolkRank for URLs)

Tag	ad. PRank
boomerang	0,4036883
shop	0,0069058
lang:de	0,0050943
software	0,0016797
java	0,0016389
programming	0,0016296
web	0,0016043
reference	0,0014713
system:unfiled	0,0014199
wood	0,0012378
kassel	0,0011969
linux	0,0011442
construction	0,0011023
plans	0,0010226
network	0,0009460
rdf	0,0008506
css	0,0008266
design	0,0008248
delicious	0,0008097
injuries	0,0008087
pitching	0,0007999

Tag	FolkRank
boomerang	0,4036867
shop	0,0066477
lang:de	0,0050860
wood	0,0012236
kassel	0,0011964
construction	0,0010828
plans	0,0010085
injuries	0,0008078
pitching	0,0007982
rdf	0,0006619
semantic	0,0006533
material	0,0006279
trifly	0,0005691
network	0,0005568
webring	0,0005552
sna	0,0005073
socialnetworkanalysis	0,0004822
cinema	0,0004726
erie	0,0004525
riparian	0,0004467
erosion	0,0004425

Tag	ad. PRank
boomerang	0,0093549
lang:ade	0,0068111
shop	0,0052600
java	0,0052050
web	0,0049360
programming	0,0037894
software	0,0035000
network	0,0032882
kassel	0,0032228
reference	0,0030699
rdf	0,0030645
delicious	0,0030492
system:unfiled	0,0029393
linux	0,0029393
wood	0,0028589
database	0,0026931
semantic	0,0025460
css	0,0024577
social	0,0021969
webdesign	0,0020650
computing	0,0020143

Tag	FolkRank
boomerang	0,0093533
lang:de	0,0068028
shop	0,0050019
java	0,0033293
kassel	0,0032223
network	0,0028990
rdf	0,0028758
wood	0,0028447
delicious	0,0026345
semantic	0,0024736
database	0,0023571
guitar	0,0018619
computing	0,0018404
cinema	0,0017537
lessons	0,0017273
social	0,0016950
documentation	0,0016182
scientific	0,0014686
filesystem	0,0014212
userspace	0,0013490
library	0,0012398

Url	FolkRank
http://www.flight-toys.com/boomerangs.htm	0,0047322
http://www.flight-toys.com/	0,0047322
http://www.bumerangclub.de/	0,0045785
http://www.bumerangfibel.de/	0,0045781
http://www.kutek.net/trifly_mods.php	0,0032643
http://www.rediboom.de/	0,0032126
http://www.bws-buhmann.de/	0,0032126
http://www.akspiele.de/	0,0031813
http://www.medco-athletics.com/education/elbow_shoulder_injuries/	0,0031606
http://www.sportsprolo.com/sports%20prolotherapy%20newsletter%20pitching%20injuries.htm	0,0031606
http://www.boomerangpassion.com/english.php	0,0031005
http://www.kuhara.de/bumerangschule/	0,0030935
http://www.bumerangs.de/	0,0030935
http://s.webring.com/hub?ring=boomerang	0,0030895
http://www.kutek.net/boomplans/plans.php	0,0030873
http://www.geocities.com/cmorris32839/jonas_article/	0,0030871
http://www.theboomerangman.com/	0,0030868
http://www.boomerangs.com/index.html	0,0030867
http://www.lmifox.com/us/boom/index-uk.htm	0,0030867
http://www.sports-boomerangs.com/	0,0030867
http://www.rangsboomerangs.com/	0,0030867

Url	FolkRank
http://jena.sourceforge.net/	0,0019369
http://www.openrdf.org/doc/users/ch06.html	0,0017312
http://dsd.lbl.gov/ hoschek/colt/api/overview-summary.html	0,0016777
http://librdf.org/	0,0014402
http://www.hpl.hp.com/semweb/jena2.htm	0,0014326
http://jakarta.apache.org/commons/collections/	0,0014203
http://www.aktors.org/technologies/ontocopi/	0,0012839
http://eventseer.idi.ntnu.no/	0,0012734
http://tangra.si.umich.edu/ radev/	0,0012685
http://www.cs.umass.edu/ mccallum/	0,0012091
http://www.w3.org/TR/rdf-sparql-query/	0,0011945
http://ourworld.compuserve.com/homepages/graeme_birchall/HTM_COOK.HTM	0,0011930
http://www.emory.edu/EDUCATION/mfp/Kuhn.html	0,0011880
http://www.hpl.hp.com/semweb/rdql.htm	0,0011860
http://jena.sourceforge.net/javadoc/index.html	0,0011860
http://www.geocities.com/mailsoftware42/db/	0,0011838
http://www.quirksmode.org/	0,0011327
http://www.kde.cs.uni-kassel.de/lehre/ss2005/googlespam	0,0011110
http://www.powerpage.org/cgi-bin/WebObjects/powerpage.woa/wa/story?newsID=14732	0,0010402
http://www.vaughns-1-pagers.com/internet/google-ranking-factors.htm	0,0010329
http://www.cl.cam.ac.uk/Research/SRG/netos/xen/	0,0010326

A closer look reveals that this ranking still contains some unexpected tags; "kassel" or "rdf" are for instance not obviously related to "boomerang". An analysis of the user ranking (not displayed) explains this fact. The top-ranked user is "schm4704", and he has indeed many bookmarks about boomerangs. A FolkRank run with preference weight 5 for user "schm4704" shows his different interests, see the rightmost column in Table 6. His main interest apparently is in boomerangs, but other topics show up as well. In particular, he has a strong relationship to the tags "kassel" and "rdf". When a community in del.ico.us is small (such as the boomerang community), already a single user can thus provide a strong bridge to other communities, a phenomenon that is equally observed in small social communities.

A comparison of the FolkRank ranking for user "schm4704" with the Adapted PageRank result for him (2nd ranking from left) confirms the initial finding from above, that the Adapted PageRank ranking contains many globally frequent tags, while the FolkRank ranking provides more personal tags. While the differential nature of the FolkRank algorithm usually pushes down the globally frequent tags such as "web", though, this happens in a differentiated manner: FolkRank will keep them in the top positions, if they are indeed relevant to the user under consideration. This can be seen for example for the tags "web" and "java". While the tag "web" appears in schm4704's tag list—but not very often, "java" is a very important tag for that user. This is reflected in the FolkRank ranking: "java" remains in the Top 5, while "web" is pushed down in the ranking.

The ranking of the resources for the tag "boomerang" given in the middle of Table 6 also provides interesting insights. As shown in the table, many boomerang related web pages show up (their topical relatedness was confirmed by a boomerang aficionado). Comparing the Top 20 web pages of "boomerang" with the Top 20 pages given by the "schm4704" ranking, there is no "boomerang" web page in the latter. This can be explained by analysing the tag distribution of this user. While "boomerang" is the most frequent tag for this user, in del.icio.us, "boomerang" appears rather infrequently. The first boomerang web page in the "schm4704" ranking is the 21st URL (i. e., just outside the listed TOP 20). Thus, while the tag "boomerang" itself dominates the tags of this user, in the whole, the semantic web related tags and resources prevail. This demonstrates that while the user "schm4704" and the tag "boomerang" are strongly correlated, we can still get an overview of the respective related items which shows several topics of interest for the user.

This example—as well as the other experiments we performed—shows that FolkRank provides good results when querying the folksonomy for topically related elements. Overall, our experiments indicate that topically related items can be retrieved with FolkRank for any given set of highlighted tags, users and/or resources.

Our results also show that the current size of folksonomies is still prone to being skewed by a relatively small number of perturbations—a single user, at the moment, can influence the emergent understanding of a certain topic in the case that a sufficient number of different points of view for such a topic has not been collected yet. With the growth of folksonomy-based data collections on the web and in social communities such as NEPOMUK, the influence of single users will fade in favor of a common understanding provided by huge numbers of users.

As detailed above, our ranking is based on tags only, without regarding any inherent features of the resources at hand. This allows to apply FolkRank to search for pictures and other multimedia content, as well as for all other items that are difficult to search in a content-based fashion. The same holds for intranet applications, where in spite of centralized knowledge management efforts, documents often remain unused because they are not hyperlinked and difficult to find. Full text retrieval may be used to find documents, but traditional IR methods for ranking without hyperlink information have difficulties finding the most relevant documents from large corpora.

Applying FolkRank for Community Detection

The original PageRank paper [11] already pointed out the possibility of using the random surfer vector $\vec{p}$ as a personalization mechanism for PageRank computations. The results of Section 5.3 show that, given a user, one can find a set of tags, resources and particulary users of interest to him. Likewise, FolkRank yields a set of related users and resources for a given tag. Following these observations, FolkRank can be used to detect communities of users which share a common interest or have interests that are related to those of a given user. These commmunities can be utilized at different points in the folksonomy system:

- Documents from related users that are of potential interest to a user can be suggested to him. This kind of recommendation pushes potentially useful content to the user and increases the chance that a user finds useful resources that he did not even know existed by "serendipitous" browsing.
- When using a certain tag, other related tags can be suggested. This can be used, for instance, to speed up the consolidation of different terminologies and thus facilitate the emergence of a common vocabulary among the community.
- The precision of document search can be improved by restricting the search space to documents of the relevant community.
- Other users that work on related topics can be made explicit, improving thus the knowledge transfer within organizations and fostering the formation of communities.

As already mentioned, FolkRank computes the winners and losers of the mutual reinforcement of users when a user or tag preference is given, compared to the baseline without a preference vector. To this end we regard the winners as belonging to the community surrounding the user or tag and the loosers as not belonging to it. With proper normalization the FolkRank vector $\vec{w}$ sums up to 0 and thus, users with a weight $\vec{w}[x] > 0$ are regarded as winners and similiarly, users with a weight $\vec{w}[x] \leq 0$ as loosers. Consequently, the set of users returned by the Community Manager contains all users with positive weight in the FolkRank weight vector.

5.4 Conceptual Clustering of Folksonomies

This section focuses on an approach to conceptually cluster folksonomies and thus find groups of users with common conceptualization. As the three dimensions of a folksonomy do not carry any structure, there is no systematic way of browsing the data, e. g., in a top-down manner. Hence, support is needed for discovering the most significant concepts (and thus communities) of the folksonomy. For two-dimensional binary data, a solution to this task are iceberg concept lattices [65]. In this section, we will extend this conceptual hierarchical clustering technique to the three-dimensional nature of folksonomies.

Our algorithm solves the problem of frequent closed itemset mining for this kind of data. It will return a tri-ordered set of (frequent) 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—called (frequent) 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. They are thus the three-dimensional version of closed itemsets.

We can additionally impose minimum support constraints on each of the three dimensions 'users', 'tags', and 'resources'. In the dyadic case, they equal the minimum support and minimal length thresholds from association rule mining. By setting higher values, we can focus on the largest conceptual components of the folksonomy before having a more detailed look with lower thresholds.

Formal Concept Analysis

FCA formalizes the notion of a '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 [70, 26]. This understanding of 'concept' is first mentioned explicitly in the Logic of Port Royal [4]. To allow a formal description of extensions and intensions, FCA starts with a (formal) context $\mathbb{K} := (G, M, I)$ which consists of a set G of objects, a set M of attributes, 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.

We define (following [70]), for $A \subseteq G$, $A' := \{m \in M \mid \forall g \in A \colon (g, m) \in I\}$; and dually, for $B \subseteq M$, $B' := \{g \in G \mid \forall m \in B \colon (g, m) \in I\}$.

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

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

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 $\mathbb{K}$ [70].

The concept lattice is a hierarchical conceptual clustering of the data which can be visualised by a Hasse diagram. This visualisation technique has been used in many applications for qualitative data analysis [25].

Closed Itemset Mining

In terms of Formal Concept Analysis, the task of mining frequent itemsets [1] is described as follows: Given a formal context $\mathbb{K} = (G, M, I)$ and a threshold $\mathrm{minsupp} \in [0, 1]$, determine all subsets B of M where the support $\mathrm{supp}(B) := \frac{\mathrm{card}(B')}{\mathrm{card}(G)}$ (with B' as defined above) is larger than the threshold $\mathrm{minsupp}$. Here, M is the set of items while G is the set of transactions.

The set of these frequent itemsets itself is usually not considered as a final result of the mining process, but rather an intermediate step. Its most prominent use is certainly association rules [1]. The task of mining association rules is to determine all pairs $X \rightarrow Y$ of subsets of M such that the support $\mathrm{supp}(X \rightarrow Y) := \mathrm{supp}(X \cup Y)$ is above the threshold $\mathrm{minsupp}$, and the confidence $\mathrm{conf}(X \rightarrow Y) := \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)}$ is above a given threshold $\mathrm{minconf} \in [0, 1]$. Association rules are for instance used in warehouse basket analysis, where the warehouse management is interested in learning about products 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 [2], 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 [51, 74, 63] 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, as 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 $\cdot'$ operators defined in the previous subsection. That is, for an itemset B, the set B'' is the smallest concept intent containing B. This closure operator will be used in the Trias algorithm in Section 5.5.

In FCA, the equivalent notion is that of an iceberg concept lattice [65], which is the $\bigvee$–semi-lattice $\{(A, B) \in \mathfrak{B}(\mathbb{K}) \mid \frac{\mathrm{card(A)}}{\mathrm{card(G)}} \geq \mathrm{minsupp}\}$ with the order defined in Section 5.4. The iceberg concept lattice visualises the most frequent concepts of a dataset [65], and allows for an efficient visualisation of a basis (condensed set) of association rules [66, 53]. These bases allow to reduce the number of rules significantly without losing any information.

Triadic Concept Analysis

Inspired by the pragmatic philosophy of Charles S. Peirce with its three universal categories [55], Rudolf Wille and Fritz Lehmann extended Formal Concept Analysis in 1995 with a third category [37]. They defined a triadic formal context as a quadruple $\mathbb{K} := (G, M, B, Y)$ where G, M, and B are sets, and Y is a ternary relation between G, M, and B, i. e., $Y \subseteq G \times M \times B$. This is exactly the structure of a folksonomy (as defined above) without the $\prec$ relation. The elements of G, M, and B are called objects, attributes, and conditions, respectively, and $(g, m, b) \in Y$ is read "object g has attribute m under condition b". A triadic concept of $\mathbb{K}$ is a triple (A_1, A_2, A_3) with $A_1 \subseteq G$, $A_2 \subseteq M$, $A_3 \subseteq B$, and $A_1 \times A_2 \times A_3 \subseteq Y$ such that none of its three components can be enlarged without violating this condition. This is the natural extension of the definition of a formal concept[22] to the triadic case.

With the three dimensions one obtains three quasi-orders $\lesssim_1$, $\lesssim_2$, and $\lesssim_3$ on the set of all tri-concepts: $(A_1, A_2, A_3) \lesssim_i (B_1, B_2, B_3)$ iff $A_i \subseteq B_i$, for $i = 1, 2, 3$. For two tri-concepts $\mathfrak{a}$ and $\mathfrak{b}$, $\mathfrak{a} \lesssim_i \mathfrak{b}$ and $\mathfrak{a} \lesssim_j \mathfrak{b}$ imply $\mathfrak{b} \lesssim_k \mathfrak{a}$, for $\{i, j, k\} = \{1, 2, 3\}$. These three quasi-orders are the triadic version of the lattice of closed itemsets in standard frequent itemset mining. In the triadic case, the relationship between the three quasi-orders is unfortunately weaker, which makes the mining more complex.

Lehmann and Wille present in [37] 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.[23] With the rise of social resource sharing systems on the web, triadic data became recently interesting for many researchers. In particular, one needs knowledge discovery and information retrieval methods that are able to handle very large datasets. In [59], we discussed how to compute association rules from a triadic context based on projections. A first step towards truly 'triadic association rules' has been done in [24].

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 1 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

[22] In terms of association rules: a closed itemset together with all its related transactions, see [70, 26] for details. [23] See `http://www.bibsonomy.org/tag/triadic+fca`

can establish minimum support thresholds on all of them. The general problem of mining frequent tri-sets is then the following:

Problem 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-minsupp and t-minsupp). 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-minsupp$:= 0$).

As in the classical case, our thresholds are antimonotonic constraints: If the tri-set (A_1, B_1, C_1) with A_1 being maximal for $A_1 \times B_1 \times C_1 \subseteq Y$ 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 directions.

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 5.4) from the triadic Galois connection [8] induced by a triadic context.

Lemma 1 (cf. [37]) Let (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 $i = 1, 2$.[24] 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.

Problem 2 (Mining all frequent tri-concepts) 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-concepts consists in determining all tri-concepts (A, B, C) of $\mathbb{F}$ with $\frac{|A|}{|U|} \geq u\text{-}minsup$, $\frac{|B|}{|T|} \geq t\text{-}minsup$, and $\frac{|C|}{|R|} \geq r\text{-}minsup$.

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

Once Problem 2 is solved, we obtain the answer to Problem 1 in a straightforward enumeration as $\{(A, B, C) \mid \exists \text{ frequent tri-concept } (\hat{A}, \hat{B}, \hat{C})\colon A \subseteq \hat{A}, B \subseteq \hat{B}, C \subseteq \hat{C}, |A| \geq \tau_u, |B| \geq \tau_t, |C| \geq \tau_r\}$.

5.5 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 1. A prior version was used for analysing psychological

[24] This holds in particular if the tri-sets are tri-concepts.

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

Algorithm 1: The TRIAS algorithm

```
1   FirstFrequentConcept($\mathbb{K}, \tau$)
2       $A := \emptyset'$
3       $B := A'$
4       if $|A| < \tau$ then
5           NextFrequentConcept($(A, B), \mathbb{K}, \tau$)
6       endif
7       return $(A, B)$
```

Algorithm 2: The *FirstFrequentConcept* function

studies [34]. That application varied from Trias as it aimed at 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 U, T, and R with natural numbers, i. e. $U = \{1, \ldots, |U|\}$ (and symmetrically for T, 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, whereas the condition in line 9 of Algorithm 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.

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 NextClosure algorithm [23, 26] with frequency pruning for implementing the FirstFrequentConcept and NextFrequentConcept routines (see Algorithms 2 and 3, resp.) for both the outer and the inner loop. This algorithm has the advantage that it needs almost only the space for the data in main memory.

NextClosure computes concepts in lectic order. This means that, for a given concept (A, B), NextClosure 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.

```
1   NextFreqentConcept$((A,B),(G,M,I),\tau)$
2       while defined$(i)$ begin
3           $A := (B \oplus i)'$
4           if $|A| \geq \tau$ then
5               $D := A'$
6               if $B <_i D$ then
7                   $B := D$
8                   return true
9               endif
10          endif
11          $i :=$ max$(M \setminus B \cap \{1,\ldots,i-1\}$
12      end
13      return false
```

Algorithm 3: The *NextFreqentConcept* function

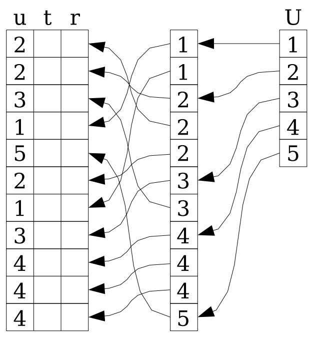


Figure 9: Accessing triples in sorted order

To find the next concept we define for $B \subseteq M$ and $i \in M$: $B \oplus i := (B \cap \{1,\ldots,i-1\}) \cup \{i\}$.

By applying the closure operator $X \mapsto X''$ to $B \oplus i$ the algorithm computes for a given B the set $D := (B \oplus i)''$. This is the lectically next intent, if $B <_i D$ holds, where $B <_i D$ means, 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 $A := (B \oplus i)'$ fullfills the minimal support criterion before computing the intent $D := A'$. This is done in line 5 of Algorithm 3 by considering the extent A only if it is large enough.

Taking a closer look on the function $\cdot'$ revealed that it demands the computation of several set intersections at a time. Since profiling showed that this is the main bottleneck of the algorithm we optimized this by first ordering the sets to be intersected by size (with the smallest set first). Then the algorithm recursively intersects them with a procedure used for merge sort. This is possible since every itemset of the binary context can be accessed as an ordered list due to the method described as follows.

Because two sortings of Y are needed, instead of storing both, we just store the permutations for every order and an additional offset table which allows constant time access to every triple. The chosen approach is exemplified in Figure 9. The table on the left contains the unsorted triples Y of which only the values from U are shown here. The table in the middle describes the permutation which allows to access the triples in lexicographic order. Finally, the third table contains for every element $u \in U$ an offset which points to the position in the second table, which points to the first triple of that user in the

A	fischer gnat
B	css design web

C	`http://www.quirksmode.org/`
	`http://webhost.bridgew.edu/etribou/layouts/`
	`http://www.picment.com/articles/css/funwithforms/`
	`http://www.alistapart.com/articles/sprites/`

A	bibi poppy
B	women cinema film

C	`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/`

Table 7: Examples of frequent tri-concepts of del.icio.us

Y list.

Together all this allows constant time access to the sorted tag-resource set of every user. Consequently, this method is applied also for accessing the columns of the respective context.

Evaluation

In order to evaluate our approach, we have analyzed the popular social bookmarking system del.icio.us. For detecting communities of users which have the same tagging behaviour, we ran the Trias algorithm on a snapshot consisting of all users, resources, tags and tag assignments we could download (cf. Section 5.2) that were entered to the system on or before June 15, 2004. The resulting folksonomy consists of $|U| = 3,301$ users, $|T| = 30,416$ different tags, $|R| = 22,036$ 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 more than one user, tag and resource, i. e., with $\tau_u := \tau_t := \tau_r := 2$. There were $1,062$ such tri-concepts. Table 7 shows two examples. They may be exploited further for extracting relations between tags or for recommending a user to get in touch with the other one, as they both use the same terminology for the same URLs and are thus likely to be on a similar line of thought.

As in the dyadic case, the size of the result may grow exponentially in the worst case. Biedermann has shown in [9] 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.

For measuring the runtime and the number of frequent concepts we have evaluated the performance of Trias on the del.icio.us dataset described before. From the base set we created monthly snapshots as follows. $\mathbb{F}_0$ contains all tag assignments performed on or before Dec 15, 2003, together with the involved users, tags, and resources; $\mathbb{F}_1$ all tag assignments performed on or before Jan 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 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 the implementations on a dual-core Opteron system with 2 GHz and 8 GB RAM.

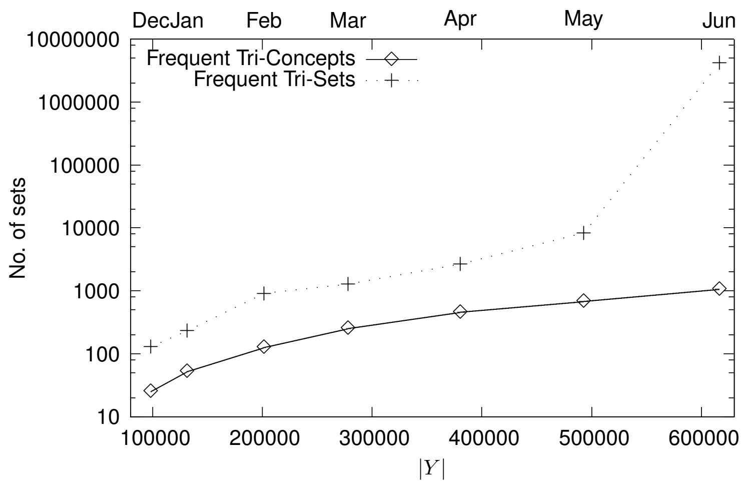


Figure 10: Number of frequent tri-sets vs. number of frequent tri-concepts

Figure 10 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 Dec 2003 to 616,819 in June 2004. What can be seen is the massive increase of frequent tri-sets in June 2004 while the number of frequent tri-concepts grows at a different level. This can be explained when looking further on the size of the tri-concepts found which also grows from month to month, since more and more users appear and start agreeing on a common vocabulary. Especially such large concepts as shown in Table 7 do not appear until June 2004 but they are responsible for the steep increase of frequent tri-sets. Overall 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.

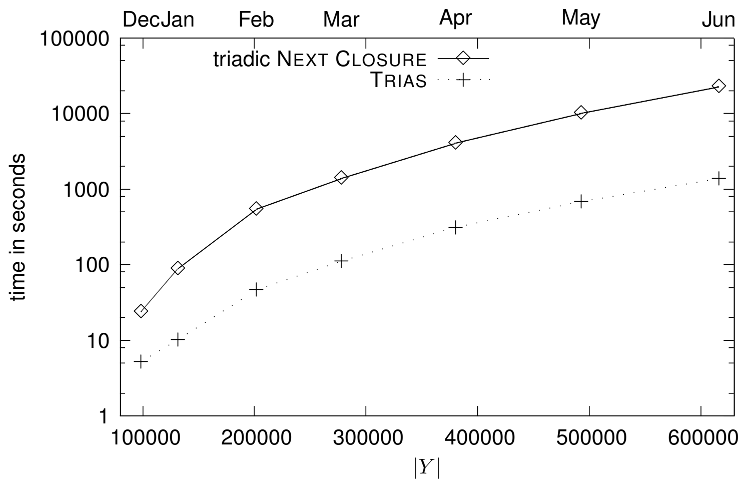


Figure 11: Runtime of triadic NEXT CLOSURE and TRIAS algorithm on del.icio.us datasets

A comparison of the speed improvement gained from not computing all tri-concepts with an algorithm like Next Closure and afterwards pruning the non-frequent concepts but using the Trias algorithm for directly mining frequent tri-concepts is shown in Figure 11. 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 the results seen we can conclude that the Trias algorithm provides an efficient method to mine frequent tri-concepts in large scale conceptual structures. Those tri-concepts can be regarded as communities consisting of users which tagged the same resources with the same tags. Hence, one can use those tri-concepts to support users in finding other users with similiar conceptualizations by suggesting them users of concepts they appear in. This could be refined when a user is looking for other users related to a certain topic, e.g., "football". Then one could extract the users which appear together with the user and the tag "football" in a tri-concept.

Insofar, the method proposed here provides some flexibility for the community detection task of D5.1. As presented in this section, we analyzed both Trias and FolkRank and chose to implement the FolkRank in the Community Manager component. FolkRank has some advantages compared to Trias in terms of speed and also allows more control on the sizes of detected communities. Hence, it is the favourite of the analyzed algorithms.

6 Metadata Alignment

Overview

The main task in metadata alignment is to identify relationships between elements of the input ontologies, most basically between the ontologies' classes. These relationships are necessary to determine which actions to perform in order to create a merged ontology (see Figure 12). Within the scope of NEPOMUK, we will not formally infer class relationships, but gather evidence for such relationships. This is due to the fact that the PIMO ontologies we want to align are ontologies created by end users, therefore mostly having a kind of ad-hoc character instead of being completely formally sound.

In the literature, metadata alignment operations are mainly based on two sources of evidence:

Term–based evidence considers similarities in the the textual description (i.e., the "name") of concepts in the source ontologies. Examples are the Chimaera ontology environment [45, 44] and Protégé's PROMPT tab [50].

Topology–based evidence considers the structure of the source ontologies, e.g., by determining similarities of the graphs representing concepts and their relationships, as done by the Similarity Flooding algorithm [46].

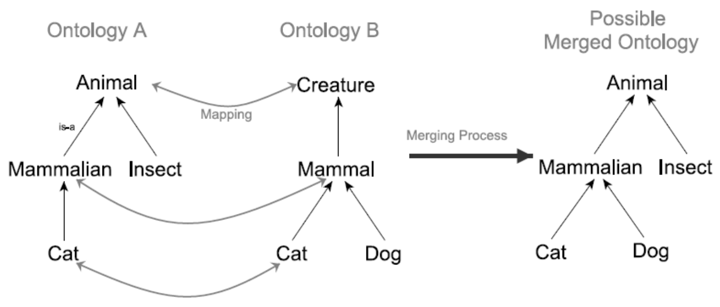


Figure 12: Merging Ontologies

The advantage of term–based evidence is that a variety of well–understood algorithms for the determination of string similarities makes this approach easy to implement. However, all precision and recall problems known from string–based information retrieval (e.g., due to synonyms or homonyms) directly apply here. Therefore, thesauri or lexica are sometimes incorporated as additional background knowledge to alleviate these problems.
Most methods that focus on topology–based evidence are, strictly speaking, hybrid: Approaches using formal logic apply matching and unification procedures that rely on some common vocabulary (i.e., they rely basically on term identity); the similarity flooding algorithm presented in [46] is hybrid as it derives its initial activation values from term similarities. For an overview of some merging tools and algorithms along with the kind of information they exploit, see Table 8. The Metadata Alignment component is designed to allow custom tailored combinations of alignment algorithms to produce the best matching procedure of ontology alignment for each application scenario. Thus, its core consists of a set of interfaces, providing a framework to integrate various types of modules to a single procedure.

Each module belongs to one of the following levels:

Ontology level Since all kinds of ontologies and alike should serve as "alignees", the Metadata Alignment component defines an ontology interface. On-

Table 8: Comparison of ontology merging approaches

References	Term	Topology	Instance
COMA [19]	x	x	
Chimaera[44]	x		
CAIMAN[35]		x	x
Similarity Flooding[46]	x	x	
PROMPT[50]	x		

tology Adapter implement this interface in order to make a certain kind of ontological data accessible for the system.

Similarity level The basis of most alignment approaches is some sort of similarity between the entities of the ontologies to be aligned. Thus, we have created an interface to such "Similarity Measures", to make them exchangable.

Procedure/alignment level The highest level is a complete alignment procedure that is able to calculate an alignment for a given pair of ontologies using the similarity values generated on similarity level. This is certainly the single most important level from a user's point of view.

Modules

In the following, an overview of the modules present in the Metadata Alignment component is given. For each module type a specialized API is available[25]. New modules can be created by implementing this API. Custom parameters can be passed to the modules upon initialization. More in-depth technical documentation is available as JavaDoc as well as on the Phase component's wiki pages[26]. As is the case with most NEPOMUK components, the Phase component is Open Source.

6.1 Ontology Adapters

Ontology adapters provide are a unified interface for several kinds of input ontologies. Note that even with ontologies using the same representation language, the granularity level on which to perform the actual matching can be chosen. For example, NEPOMUK ontologies are expressed in NRL which is in turn RDFS-based. However, since most user concepts are expressed as PIMO instances (within NRL), it may be desirable not to perform alignment on the (rather basic) RDFS class level but on PIMO level. Currently, we use the RDFS-based adapter, but in future ontologies optimized for NEPOMUK domain will be taken into account.

RDFS Ontologies This adapter allows accessing RDFS based ontologies, including ontologies currently present in NEPOMUK.

OWL Ontologies This adapter enables the metadata alignment component to access OWL ontologies, represented as RDF files.

Protégé Ontologies Provides an interface to all Protégé projects.

Document Classification Stores DCSs are integrated document management and classification systems. Their document topic taxonomy is treated as a simple kind of ontology.

[25] See `org.semanticdesktop.nepomuk.comp.phasealignment.model.*` packages

[26] `http://dev.nepomuk.semanticdesktop.org/wiki/PhaseAlignment`

Composite ontology This is a meta adapter allowing to create modified views on other ontologies by expanding or contracting subgraphs. This is handy for similarity algorithms that have problems with very large ontologies.

6.2 Similarity Measures

In the following subsections we first describe the different similarity measures which are implemented to be used as evidence for the ontology alignment algorithm and then we describe each implemented method in detail. These measures can be divided into three main categories.

Element-based methods are used to measure the correspondence of two elements of two ontologies at a local level, i.e., only comparing one element with another one and not considering the topology and structure of the ontologies.

Structure-based In structure-based matching, the topologies of the respective ontologies are used to compute similarity information. The more sophisticated the structure of the ontologies is, the more information can be gathered using this technique: Flat ontologies do not contain enough structural "fingerprints" to be suitable for structure-based matching.

Instance-based Determines the similarity between concepts by calculation the similarity between sets of example objects. Currently we do not utilize this kind of similarity to induce mappings.

6.2.1 String Based Similarity

This similarity measure is element-based and determines the similarity of two ontology entities by calculating a simple string alikeness on the labels of these entities. In our implementation, we incorporated similarity based on n-grams, which is fast and language independent. n-gram distance is defined as follows: Let $ngram(s, n)$ be the set of all substrings of s (augmented with $n - 1$ irrelevant characters at the beginning and the end) of length n, the n-gram distance is a dissimilarity measure between two strings s and t:

$\delta(s, t) = |ngram(s, n) \cap ngram(t, n)|$

The normalized version of this function is:

$\overline{\delta}(s, t) = \frac{|ngram(s,n) \cap ngram(t,n)|}{n \times min(|s|,|t|)}$

This function is quite efficient when some characters are only missing, and not replaced by other characters.

6.2.2 Acronym Matcher

This similarity measure is also element-based and determines the similiarity of two ontology entities by analyzing the entities' labels for one being an acronym of the other. In practice, this is implemented using an algorithm that compares both labels using a variant of a longest common subsequences algorithm, assigning penalty points for certain characteristics that make similarity less likely. For example, "Resource Description Framework" is a perfect match for "RDF" whereas any additional letters in the acronym or any additional words in the name would diminish similarity. Several rules are implemented and have been tested in example scenarios.

6.2.3 Similarity Flooding

Similarity Flooding [46] is a structure-based algorithm that is already used in the area of databases for schema mapping. It is a generic approach for determining matching nodes in graphs, taking two graphs as input and returning a set of match pairs with corresponding similarity values. It uses a propagation mechanism to promote a given or confirmed similarity to the neighboring nodes. Similarity Flooding roughly consists of the following steps:

1. Create graph representations for the ontologies.
2. Set up an initial map (set up the initial similarity value of match pairs).
3. Perform the core Similarity Flooding algorithm.
   - Create pairwise connectivity graph.
   - Create induced propagation graph.
   - Compute similarity values.

As Similarity Flooding works on graphs, we first need to create graphs (models) representing the structure of the ontologies we want to align. Then, we need to set up initial similarity values for each pair of graph nodes where a pair are two nodes, one from the model representing ontology A, and one from the model representing ontology B. The Similarity Flooding algorithm then creates a pairwise connectivity graph. Each node of the pairwise connectivity graph represents a potential match pair of both of the models' nodes, therefore the pairwise connectivity graph combines the two models and gives a notion on how similarity propagates. The induced propagation graph is a refined version of the pairwise connectivity graph, consisting of the same nodes, but featuring additional edges. A weight value is assigned to each edge. This value denotes how well similarity propagates. When computing similarity values, we perform a fixpoint calculation, starting by assigning the initial map's similarity values to the nodes of the induced propagation graph, and then propagating similarity along the graph's edges until the similarity values do not change more than an arbitrarily-chosen value between iterations. We will now discuss the steps of Similarity Flooding in more detail.

Creating graph representations This can be done for an ontology in several ways. Due to the way Similarity Flooding constructs its propagation graph, it is beneficial to use a graph representation that uses a large amount of distinctively labeled edges. The edges' labels must stay generic, i.e. unrelated to the frame names used in the ontology.

In figure 13, two models representing the same ontology are shown. Model A uses differently named edges representing the type of the respective node. Model B uses a separate type edge annotating nodes with their respective type. Therefore Model B needs two nodes (for the nodes representing class and templateslot types) and five edges (for associating nodes with type nodes) in excess of Model A.

While this difference is not very big, the complexity difference of the pairwise connectivity graphs is big. Here, the pairwise connectivity graph is created from Model A and Model A' (representing the same model but using corresponding German identifiers as node names), shown in figure 14 . The same

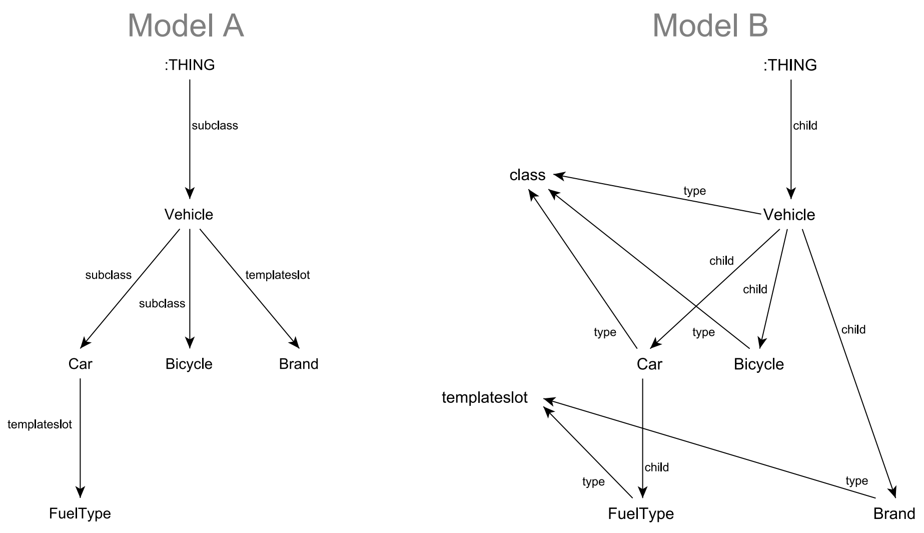


Figure 13: One Ontology, Two Representation Styles

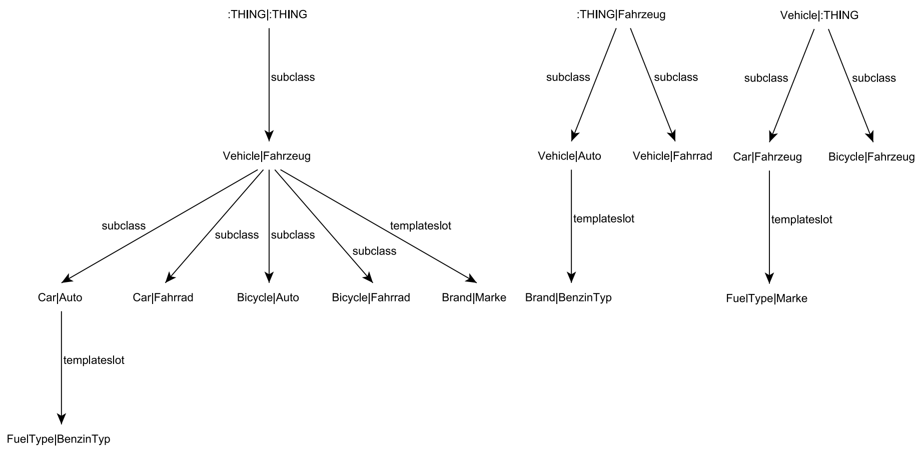


Figure 14: Pairwise Connectivity Graph for Model A/A'

is done with Model B (figure 15). The pairwise connectivity graph for Model B/B′ uses twice the number of nodes the pairwise connectivity graph for Model A/A′ uses. The number of edges is about four times as high.

Due to these differences in complexity, we chose to use the way to generate models from ontologies shown by Model A (figure 13).

Setting up the initial map The initial similarity value of all match pairs is set to 0.5. The original Similarity Flooding algorithm uses a similarity value determined by a string comparison. However, in our application Similarity Flooding shall only use structural information as name-based matching is already performed by the separate name-based matcher. Moreover, we will use the matches the user manually confirmed or rejected to set up a more meaningful initial map for the following matching iterations.

Creating the pairwise connectivity graph Every edge in the model is represented as a triple (s, p, o) with s being the source node, o the target node and p the label of the edge. The pairwise connectivity graph's edges are defined by $((x, y), p, (x', y')) \in PCG(A, B) \Leftrightarrow (x, p, x') \in A \wedge (y, p, y') \in B$ where A and B are the Models A and B and PCG(A, B) is the pairwise connectivity

```
for every edge $(x, p, x')$ from Model A
    for every edge $(y, p, y')$ from Model B
        add edge $(x|y, p, x'|y')$ to PCG
    end
end
```

Algorithm 4: The PCG generating algorithm

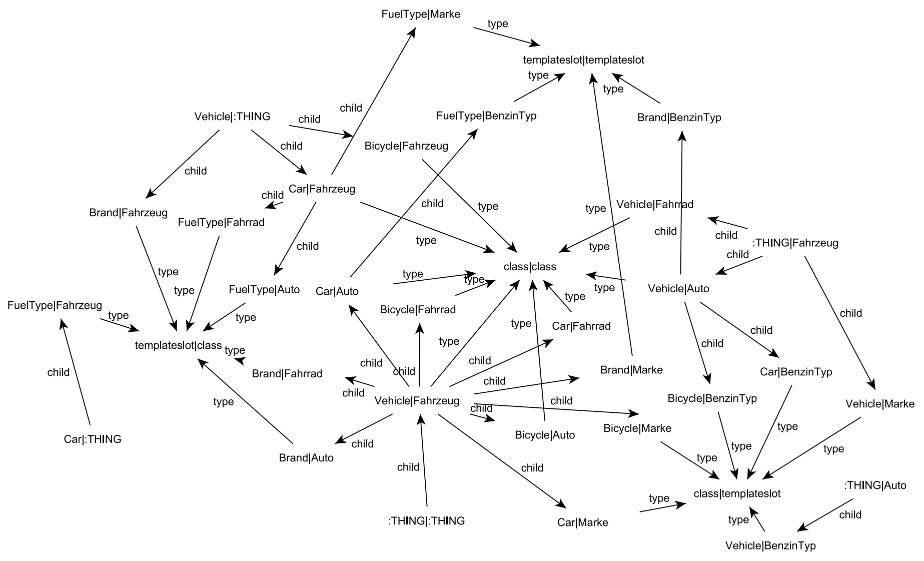


Figure 15: Pairwise Connectivity Graph for Model B/B'

graph derived from the models. Note that every node of PCG(A, B) is an element from $A \times B$, representing a match pair. The pairwise connectivity graph can be constructed by the following algorithm 4.

Here, only edges are considered, nodes must be created once referenced by any edge. For an example, see figures 13 and 14 . Clearly, the more equally-labeled edges are in the models, the more edges the PCG will consist of.

Creating the induced propagation graph The induced propagation graph (IPG) is constructed from the pairwise connectivity graph by adding a new edge for every existing edge pointing in the opposite direction. Weights are associated with the edges, indicating how well the similarity of a match pair propagates to its neighbors and back. The weights can be computed by the algorithm 5 . Figure 16 shows an example IPG corresponding to the PCG shown in figure 14.

There are other ways to compute the IPG (see [46]).

Computing similarity values The calculation following the previous steps is an interative fixpoint calculation, finding a vector of similarity values corresponding to the match pairs (nodes) from the IPG that represents a fixpoint. This means that the vector does not change even when performing a new iteration of the algorithm. A similarity value $\sigma(x, y) \geq 0$ gets assigned to each node in the IPG labelled x|y. Similarity values are zero for match pairs not occurring in the IPG.

$\sigma(x, y)^i$ denotes the similarity value of the match pair (x, y) in the $i - th$ iteration of the algorithm. σ^0 therefore denotes all similarity values of the initial

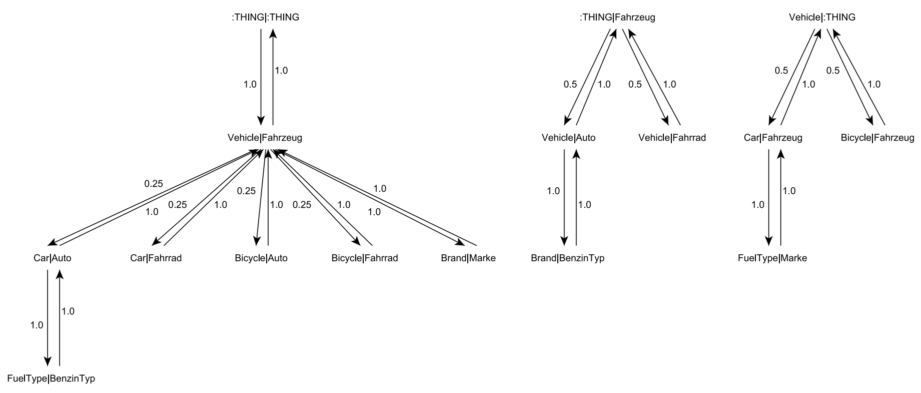


Figure 16: Example Induced Propagation Graph

```
1   initialize all edges' weights with zero
2   for every node $d$ in the IPG
3       for each group $g$ of equally-labeled edges originating from $d$ in the PCG
4           for each edge $p$ in group $g$
5               add $1/|g|$ to the weight of the edge corresponding to $p$ in the IPG
6           end
7       end
8       for each group $h$ of equally-labeled edges pointing to $d$ in the PCG
9           for each edge $q$ in group $h$
10              add $1/|h|$ to the weight of the edge corresponding to reversed $q$
11              in the IPG
12          end
13      end
14  end
```

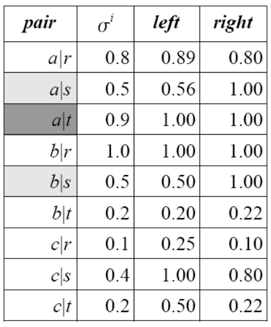
Algorithm 5: The ICG generating algorithm

map. $\sigma(x,y)^{i+1}$ can be derived from $\sigma(x,y)^i$ by calculating $normalize(\sigma^0 + \sigma(x,y)^i + \vartheta(\sigma^0 + \sigma(x,y)^i))$ where normalize divides all elements of the vector passed to it by the maximum element of the vector (so the normalized vector's maximum element is 1.0).$\vartheta(x)$ denotes adding to every $\sigma(x,y)^i$ the weighted $\sigma(x,y)^i$-value of its neighbors in the IPG. A description how to implement this using matrix-vector multiplication is given in the following paragraphs. There are other variants of the fixpoint formula [46]. However, this formula weights the initial map's similarity values σ^0 higher than other variants of the fixpoint formula which fits our needs due to the initial map denoting the class matches the user confirmed. The algorithm stops once the Euclidean length of $\Delta(\sigma(x,y)^i, \sigma(x,y)^{i-1})$becomes less than a chosen ϵ (a fixpoint is found).

This algorithm can be implemented using a vector V representing the similarity values for every node (match pair) of the IPG. Also, there is a matrix G representing the weights of the IPG's edges. The order of the matrices' rows and columns must match the order of the vector's elements meaning if $G_{1,1}$ is associated with node g of the IPG, V_1 also must correspond to node g. $x' = \vartheta(x)$ then simply means multiplying G with x and storing the result in x'.

σ^i finally denotes match pair similarity $\sigma(x,y)^i = \sigma(y,x)^i$, therefore Similarity Flooding yields undirected similarity. In order to obtain directed similarity, it is possible to compute relative similarity by normalizing the maximum per-node similarity to 1. See figure 17 for an example.

The node a occurs paired with the node r, s, and t $(a|r, a|s, a|t)$. As the similarity value 0.9 of $a|t$ is the best similarity value node a can achieve, the similarity values for $a|r$, $a|s$ and $a|t$ get divided by 0.9. Therefore, in



pair	σ^i	*left*	*right*
$a\|r$	0.8	0.89	0.80
$a\|s$	0.5	0.56	1.00
$a\|t$	0.9	1.00	1.00
$b\|r$	1.0	1.00	1.00
$b\|s$	0.5	0.50	1.00
$b\|t$	0.2	0.20	0.22
$c\|r$	0.1	0.25	0.10
$c\|s$	0.4	1.00	0.80
$c\|t$	0.2	0.50	0.22

Figure 17: Computing Relative Similarity

the left-orientated column of the example, the optimal pair from a's point of view, a|t, gets assigned the maximum similarity value of 1. After calculating the left-orientated similarity values, the right-orientated similarity values are computed in the same manner. For example, from s's point of view both the $a|s$ and the $b|s$ pair are optimal (with their maximum similarity value being 0.5). Therefore, the right-orientated similarity values are computed by dividing all pairs that include node s by 0.5.

6.2.4 Graph Matching

This measure is structure-based, using a classifier to determine entity similarity. Similar to Similarity Flooding, the actual algorithm takes a graph as input, so the input ontologies have to be transformed into generic graphs first. Then, an association graph is contructed, relating both separate graphs to each other. The graph matching module implements two different approaches to generate association graphs, namely the Complete Similar Edges algorithm, and the At Least One Edge algorithm. Using association graphs, maximum common subgraphs (MCSs) can be determined. An MCS refers to structurally equal parts of the two input graphs.

Completely Similar Edge Association Graph (CSE) The CSE association graph is a strict version of an association graph. It is suitable in case of regarding only taxonomies (Taxonomy graph) since then only a limited number of distinct edges types are present (mainly the `subclass-of` edge type).

Definition 2 The CSE association graph of two graphs g_1 and g_2 is a non-labeled, undirected graph $G = (V, E)$ where

- $V = (u_1, u_2)|u_1 \in V_1 \wedge u_2 \in V_2 \wedge similar(\alpha_1(u_1), \alpha_2(u_2))$
- E: An edge between two vertices (u_1, v_1) in g_1 exists and edge (u_2, v_2) in g_2 with
  1. the type of (u_1, v_1) is equal to the type of (u_1, v_1) and
  2. $similar(\alpha((u_1, v_1)), \alpha((u_2, v_2)))$ and vice versa,

  or if there is neither an edge between (u_1, v_1) nor between (u_2, v_2).

Hence, the number of edges between (u_1, v_1) and (u_2, v_2) must be the same.

At least One Edge Association Graph (ALOE) This type of association graph is less strict version than CSE and is more suitable for ontology matching since it is better suited for handling graphs with many different edge types (that ontologies typically correspond to).

Definition 3 The ALOE association graph of two graphs g_1 and g_2 is a non-labeled, undirected graph $G = (V, E)$ where

- $V = (u_1, u_2)|u_1 \in V_1 \wedge u_2 \in V_2 \wedge similar(\alpha_1(u_1), \alpha_2(u_2))$
- E: An edge between two vertices (u_1, v_1) and (u_2, v_2) of the association graph exists, either if, For at least one edge (u_1, v_1) in g_1 exists an edge (u_2, v_2) in g_2 with

  1. the type of (u_1, v_1) is equal to the type of (u_1, v_1) and
  2. $similar(\alpha((u_1, v_1)), \alpha((u_2, v_2)))$

  or if there is neither an edge between (u_1, v_1) nor between (u_2, v_2).

Hence, the number of edges must not be the same. The function $similar()$ denotes a similarity measure for vertices and edges, which can be based on a string distance (e.g. Levenshtein distance), a distance from a dictionary (e.g. wordnet), or more generally, a similarity measure from a previous alignment process, or even a combination of several similarity measures.

Originally, the definition of the maximum common subgraph of two graphs is strict in the sense that only exact subgraphs are taken into consideration. It is impractical to apply this definition to ontologies, since exact matching subparts of ontologies are rare in the NEPOMUK use case—ontologies for the same domain are created by end users without any knowledge of each other so while it is likely that similar structures are created, exact matches are very unlikely to occur. Therefore, a weaker definition of the MCS is used in our implementation, allowing minor derivations.

6.3 Alignment Generators

Alignment generators take the data computed by similarity measures and compute an actual alignment. Alignments can be created to fulfill several different requirements—a typical requirement for alignments is that only two classes may get marked as equivalent (i.e., any class in ontology A can be marked as equivalent to at maximum one class in ontology B and vice versa). Again, this alignment can be refined further—for example, class mappings can be derived from similarity using a threshold algorithm or more advanced techniques such as algorithms creating alignments that fulfill stable marriage requirements[27].

Currently, a number of simple alignment generators are implemented in the metadata alignment component.

Phase-Tab Algorithm This is an alignment generator using the input of three different similarity measures as input, namely string-based similarity (implemented using n-grams), structure-based similarity (implemented using similarity flooding), and instance-based similarity (implemented using an external document classification component that compares text documents attached to concepts). Similarities are combined using a

[27] In a *stable marriage* or *stable matching* no element of the first matched set (classes of ontology A in our case) prefers an element of the second matched set (classes of ontology B in our case) that also prefers the first element. See `http://en.wikipedia.org/wiki/Stable_Marriage_Problem` for details.

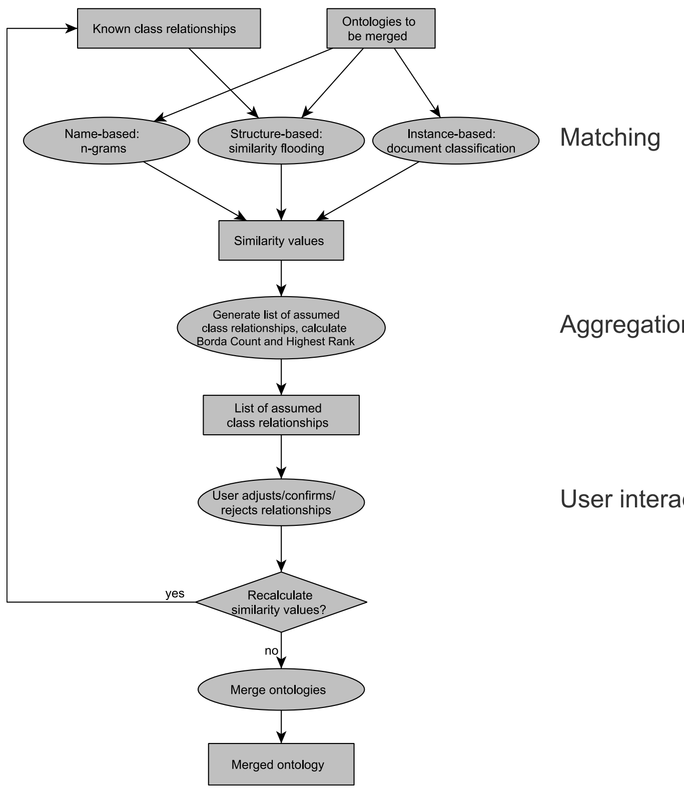

Known class relationships
Ontologies to be merged
Name-based: n-grams
Structure-based: similarity flooding
Instance-based: document classification
Matching
Similarity values
Generate list of assumed class relationships, calculate Borda Count and Highest Rank
Aggregation
List of assumed class relationships
User adjusts/confirms/ rejects relationships
User interaction
yes
Recalculate similarity values?
no
Merge ontologies
Merged ontology

Figure 18: The Phase-Tab Algorithm

modified Borda count[28] method. This algorithm, or more specifically, its similarity flooding-based similarity measure, can incorporate user feedback, enhancing its results. In NEPOMUK the interfaces required for user feedback can be added to components that do structural recommendation or search. For completely automated mapping, we found that using pseudo relevance feedback also yields results that are quite acceptable. See figure 18 for a representation of data flow in this algorithm.

Simple Borda Count This generator uses a Borda count algorithm [69] to combine an arbitrary set of similarity measures to one alignment. The Borda count for a match pair is the sum of the number of pairs ranked below it by each matcher. Compared to the Highest Rank Method which basically implements a maximum operator, the Borda Count Method can be compared to an average operator. Matchers which assign a low rank to a match pair get ignored by the Highest Rank Method while the Borda Count Method takes this decision in account when computing the overall ranking.

Simple Evidence Converts a single similarity measure using a threshold algorithm into an alignment. Useful for testing.

6.4 Evaluation

A preliminary evaluation of the Phase component has been done using the Ontology Alignment Evaluation Initiative Test library[29] that provides a frame-

[28] The Borda count is a single winner election method. In our case, it is used for combining similarities that use different scales - in this case, other methods such as arithmetic mean fail. See [69] for details. [29] `http://oaei.ontologymatching.org/tests/`

work and extensive test ontologies as well as a gold standard for expected mappings. Test results can be found on the Phase component's website[30]. It has to be noted that these results have been achieved without extensive optimizing. In the NEPOMUK context, we expect to enhance results primarily due to adapting to the application domain.

[30] `http://dev.nepomuk.semanticdesktop.org/wiki/PhaseAlignment`

7 Conclusions

In this deliverable we have implemented the basic functionalities for the semantic social networking and knowledge exchange contributions of WP5000 to the Social Semantic Desktop NEPOMUK. We developed, implemented, and analyzed methods for community detection and labeling, as well for (semi-) automatic metadata alignment.

The Community Manager plays the central role for the community support challenges of WP5000 and with the implementation done within this deliverable, it allows the extraction of communities of users inside the NEPOMUK P2P network. Those communities can be defined as belonging to a user or by their affinity to a certain topic.

Besides its importance for WP5000, the Community Manager provides large potential benefit for the case studies. The members of the Mandriva community will be able to harness the community detection for discovering users who run into similiar problems as they do (and to contact them to discuss possible workarounds), using the tags attributed automatically or by hand to their questions. The knowledge workers at SAP can discover other collegues interested in similiar topics as they are by querying the Community Manager for users interested in a specific topics or using their user name as centre of the community, or the consultants at PRC are able to quickly find associates which have knowledge in a certain field.

While we have shown how communities around given tags or users can be inferred by FolkRank, some open questions remain to be researched:

- What exactly constitutes a community in a folksonomy?
- Can different kinds of communities be distinguished?
- Which elements of a folksonomy should be used with FolkRank to start a community?

Those questions lead to the next subtasks of the Social Network Analysis task of WP5000, which include the analysis of the structure within communities, e.g., finding leaders and followers, or finding communities with similar structures. This will be augmented by algorithms to detect trends and threads within and across communities of users. There we want to support users of the Social Semantic Desktop in finding relevant topics in communities and in gaining an understanding of their relation to each other.

In the metadata alignment component we presented a framework that collects and integrates heuristic evidences for ontology mappings. The framework employs three basic sources of evidence for ontology mappings, namely term-based, topology-based and instance-based evidence. In implementing the alignment generator, we use a rather simple voting schema for the aggregation of evidence, the so-called Borda Count. Further work can comprise:

- more flexible aggregation of evidence, e.g., by using logic-based evidence integration and by adaptive aggregations functions whose parameters may be learned,
- support for other types of relationships, e.g., by additional heuristics for the instance-based evidence generation, and
- richer mapping languages that also allow for mapping composed concepts.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of Data (SIGMOD'93), pages 207–216. ACM Press, May 1993.

[2] 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, September 1994.

[3] 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, March 1995.

[4] 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.

[5] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. SIGKDD Explorations, Special Issue on Scalable Algorithms, 2(2):71–80, 2000.

[6] R. J. Bayardo. Efficiently mining long patterns from databases. In Proceedings of the 1998 ACM SIGMOD international conference on Management of Data (SIGMOD'98), pages 85–93. ACM Press, June 1998.

[7] 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, number 1257 in LNAI, pages 304–317, Heidelberg, 1997. Springer.

[8] K. Biedermann. Triadic Galois connections. In K. Denecke and O. Lüders, editors, General algebra and applications in discrete mathematics, pages 23–33, Aachen, 1997. Shaker Verlag.

[9] K. Biedermann. Powerset trilattices. In M.-L. Mugnier and M. Chein, editors, Conceptual Structures: Theory, Tools and Applications, number 1453 in LNAI, pages 209–224, Heidelberg, 1998. Springer.

[10] J.-F. Boulicaut, A. Bykowski, and C. Rigotti. Approximation of frequency queries by means of free-sets. In PKDD '00: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, pages 75–85, London, UK, 2000. Springer-Verlag.

[11] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30(1-7):107–117, April 1998.

[12] 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 Press.

[13] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In PKDD, pages 74–85, 2002.

[14] C. Carpineto and G. Romano. Concept Data Analysis. Wiley, 2004.

[15] C. Cattuto, V. Loreto, and L. Pietronero. Collaborative tagging and semiotic dynamics, May 2006. `arXiv:cs.CY/0605015`.

[16] A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. Physical Review E, 70:066111, 2004.

[17] Connotea Mailing List. `https://lists.sourceforge.net/lists/listinfo/connotea-discuss`.

[18] F. Dau and R. Wille. On the modal unterstanding 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.

[19] H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[20] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach, 2003.

[21] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins. Visualizing tags over time. In Proc. of the 15th International WWW Conference, 2006.

[22] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. Physical Review E, 72:027104, 2005.

[23] 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.

[24] 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. Springer, 2004.

[25] B. Ganter, G. Stumme, and R. Wille, editors. Formal Concept Analysis – Foundations and Applications, volume 3626 of LNAI, Heidelberg, 2005. Springer.

[26] B. Ganter and R. Wille. Formal Concept Analysis: Mathematical foundations. Springer, 1999.

[27] H. Halpin, V. Robu, and H. Shepard. The dynamics and semantics of collaborative tagging. In Proceedings of the 1st Semantic Authoring and Annotation Workshop (SAAW'06), 2006.

[28] T. Hammond, T. Hannay, B. Lund, and J. Scott. Social Bookmarking Tools (I): A General Review. D-Lib Magazine, 11(4), April 2005.

[29] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Computer Science Department, April 2006.

[30] 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 LNAI, pages 411–426, Heidelberg, June 2006. Springer.

[31] 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, December 2006. IEEE Computer Society.

[32] M. Kamber, J. Han, and Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In Proc. of the 3rd KDD Int'l Conf., August 1997.

[33] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5):604–632, 1999.

[34] 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, 1994.

[35] M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings, 2001.

[36] R. Lambiotte and M. Ausloos. Collaborative tagging as a tripartite network, Dec 2005. `arXiv:cs.DS/0512090`.

[37] 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. Springer Verlag, 1995.

[38] 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 Computer Science, pages 32–43. Springer, 1995.

[39] B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In Proceedings of the 3rd international conference on Knowledge Discovery and Data mining (KDD'97), pages 227–230. AAAI Press, August 1997.

[40] D. Lin and M. Kedem. A new algorithm for discovering the maximum frequent set. In Proceedings of the 6th Int'l Conf.on Extending Database Technology (EDBT), pages 105–119, March 1998.

[41] B. Lund, T. Hammond, M. Flack, and T. Hannay. Social Bookmarking Tools (II): A Case Study - Connotea. D-Lib Magazine, 11(4), April 2005.

[42] H. Mannila. Methods and problems in data mining. In Proceedings of the 6th biennial International Conference on Database Theory (ICDT'97), Lecture Notes in Computer Science, Vol. 1186, pages 41–55. Springer-Verlag, January 1997.

[43] A. Mathes. Folksonomies – Cooperative Classification and Communication Through Shared Metadata, December 2004. `http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html`.

[44] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000), 2000.

[45] D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In Seventh International Conference, 2000.

[46] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In Proc. 18th ICDE Conf., 2001.

[47] 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, ISWC 2005, volume 3729 of LNCS, pages 522–536, Berlin Heidelberg, November 2005. Springer-Verlag.

[48] 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, ISWC 2005, volume 3729 of LNCS, pages 522–536. Springer-Verlag, November 2005.

[49] M. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E, 69:026113, 2004.

[50] N. F. Noy and M. A. Musen. Anchor-prompt: Using non-local context for semantic matching. In Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001), Seattle, WA, 2001.

[51] 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, Octobre 1999.

[52] 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), Lecture Notes in Computer Science, Vol. 1540, pages 398–416. Springer-Verlag, January 1999.

[53] 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.

[54] 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.

[55] C. S. Peirce. Collected Papers. Harvard Universit Press, Cambridge, 1931–1935.

[56] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks, Feb 2004. `arXiv:cond-mat/0309488v2`.

[57] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. VLDB Journal: Very Large Data Bases, 10(4):334–350, 2001.

[58] 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.

[59] 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: Proc. of the 10th IFCS Conf., Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Berlin, Heidelberg, 2006. Springer.

[60] P. Schmitz. Inducing ontology from flickr tags. In Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May 2006.

[61] P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. J. Data Semantics IV, 3730:146–171, 2005.

[62] 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, January 1998.

[63] G. Stumme. Conceptual knowledge discovery with frequent concept lattices. FB4-Preprint 2043, TU Darmstadt, 1999.

[64] 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. Springer, 2005.

[65] G. Stumme, R. Taouil, Y. Bastide, N. Pasqier, and L. Lakhal. Computing iceberg concept lattices with titanic. J. on Knowledge and Data Engineering, 42(2):189–222, 2002.

[66] 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 LNAI, pages 335–350. Springer, Heidelberg, 2001.

[67] TagDB Mailing List. `http://lists.tagschema.com/mailman/listinfo/tagdb`.

[68] 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.

[69] M. van Erp and L. Schomaker. Variants of the borda count method for combining ranked classi er hypotheses, 2000.

[70] 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.

[71] R. Wille. The basic theorem of triadic concept analysis. Order, 12:149–158, 1995.

[72] 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-Verlag.

[73] W. Xi, B. Zhang, Y. Lu, Z. Chen, S. Yan, H. Zeng, W. Ma, and E. Fox. Link fusion: A unified link analysis framework for multi-type interrelated data objects. In Proc. 13th International World Wide Web Conference, New York, 2004.

[74] 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, October 1999.

[75] 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.

[76] A. V. Zhdanova and P. Shvaiko. Community-driven ontology matching. In European Semantic Web Conference, pages 34–49, Budva, Montenegro, 2006.

A Abbreviations

ALOE	At least One Edge
API	Application Programming Interface
CSE	Completely Similar Edge
DCS	Document Classification Stores
ER	Entity-Relation
FCA	Formal Concept Analysis
FOAF	Friend of a Friend
HTML	Hypertext Markup Language
HTTP	Hyptertext Transfer Protocol
IPG	Induced Propagation Graph
MCS	Maximum Common Subgraphs
NOA	NEPOMUK Annotation Ontology
NRL	NEPOMUK Representation Language
OSGi	Open Services Gateway Initiative
OWL	Web Ontology Language
P2P	Peer-to-Peer
PCG	Pairwise Connectivity Graph
PIMO	Personal Information Management Ontology
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSDL	Web Service Description Language
XML	Extensible Markup Language