Integrated Project

Priority 2.4.7

Semantic based knowledge systems

The Social Semantic Desktop

NEPOMUK

Metadata Sharing and Recommendation Application

First Prototype

Deliverable D5.2

Version 1.0

17.12.2007

Dissemination level: PU

Nature	Prototype
Due date	31.12.2007
Lead contractor	LEIBNIZ UNIVERSITAET HANNOVER
Start date of project	01.01.2006
Duration	36 months

Authors

Gianluca Demartini, L3S Research Center
Robert Jäschke, L3S Research Center
Rodolfo Stecher, L3S Research Center
Parisa Haghani, EPFL Ecole Polytechnique Fédérale de Lausanne

Mentors

Leo Sauermann, DFKI German Research Center for Artificial Intelligence DFKI GmbH
Roman Schmidt, EPFL Ecole Polytechnique Fédérale de Lausanne

Contributors

Vasilios Darlagiannis, EPFL Ecole Polytechnique Fédérale de Lausanne
Philippe Cudré-Mauroux, EPFL Ecole Polytechnique Fédérale de Lausanne
Claudia Niederée, L3S Research Center

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

Versions

Version	Date	Reason
0.1	2007-08-24	First draft
0.2	2007-10-09	Added sections on recommending tags
0.3	2007-10-16	Added sections on architecture
0.4	2007-10-19	Added scenarios to requirements section and worked on architecture section
0.5	2007-10-25	Added introduction and modified architecture section
0.6	2007-10-31	Added executive summary
0.7	2007-12-05	Incorporated mentors comments, shortened parts of section 6.5
0.8	2007-12-07	Added Appendix: Specification of Exchange Formats & Workflows for Metadata Sharing and Recommendations
1.0	2007-12-17	Final version; finalisation by IMC

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 as well as for supporting knowledge exchange in these semantic social networks. The goal of the second deliverable is to provide the basic functionality for metadata sharing and recommendation. The provided functionality is packaged in three components (as they have been identified in the Nepomuk architecture): i) Metadata Sharing, ii) Metadata Recommender, and iii) Expert Recommender. These components can be adopted and support several scenarios focusing on social interactions among desktops. The components' functionality is provided to other Nepomuk components or applications via well defined APIs, which have been tested by prototype applications.

Shortly, our work resulted in the following:

- The Metadata Sharing[1] component, being responsible for building a semantic overlay infrastructure based on the P2P[2] infrastructure provided by WP4000;
- The Metadata Recommender[3] component, which provides the user with recommendations built using the metadata in the network of Nepomuk peers. By analyzing the overlap between metadata and resources, it provides recommendation for: i) metadata enrichment, ii) metadata-based recommendation of new resources, and iii) recommendation for tags;
- The Expert Recommender[4] component, which provides the users with a list of persons who are experts on a given topic.

The main task of the Metadata Sharing component is to provide a semantic overlay infrastructure on top of a P2P overlay network which separates the logical layer – where data, schemas and schema mappings are managed – from the physical layer consisting of the structured P2P network. It fosters semantic interoperability through pair-wise schema mappings and query reformulation. Thus, heterogeneous but semantically related information sources can all be queried transparently using iterative query reformulation.

For the realization of the Metadata Sharing component:

- GridVine, the first metadata sharing system addressing simultaneously both scalability and semantic heterogeneity is described;
- A general overview of our infrastructure is provided with details on both the P2P access structure used for indexing information and routing messages and the higher semantic layer responsible for managing structured content;
- The decentralized information integration based on pair-wise schema mappings between heterogeneous but semantically related schemas is given;
- Several distributed query resolution mechanisms are discussed and we illustrate the performance of our infrastructure.

To accomplish the recommendation tasks provided by the Metadata Recommender and by the Expert Recommender:

[1] Located on the Nepomuk SVN Server `https://dev.nepomuk.semanticdesktop.org` in the directory /repos/trunk/java/org.semanticdesktop.nepomuk.comp.metadatasharing [2] Abbreviations are listed in the Appendix A. [3] Located on the Nepomuk SVN Server `https://dev.nepomuk.semanticdesktop.org` in the directory /repos/trunk/java/org.semanticdesktop.nepomuk.comp.metadatarecommender [4] Located on the Nepomuk SVN Server `https://dev.nepomuk.semanticdesktop.org` in the directory /repos/trunk/java/org.semanticdesktop.nepomuk.comp.expertrecommender

- We describe how we can provide recommendations to enrich the metadata description of the current desktop resources using the shared metadata in the Nepomuk network of peers, and how we can provide recommendations of new resources similar to a given resource;
- We present a model for finding experts in a given topic using the resources on the desktop; our model also allows to retrieve both people and document together;
- We describe how the folksonomy structure of the Nepomuk network of peers can be used to find suitable tag recommendations for resources the user wants to annotate.

The implemented Metadata Recommender and Expert Recommender components allow the recommendation of new resources, persons, and of additional metadata using the Nepomuk P2P network.

Summarizing, we have developed the necessary functionalities for the semantic social networking and knowledge exchange. We have designed, analyzed and implemented methods for sharing of metadata and for recommending resources, persons, and metadata. In the rest of the project the focus will be put on extending the available functionalities, integration and deeper evaluation.

Together with Deliverable 5.2 we deliver a report on "Specification of Exchange Formats & Workflows for Metadata Sharing and Recommendations" which was written in the first year of the project. In the description of work it was not mentioned to be part of a deliverable but an intermediate report. We hereby hand this report in later, as promised to the reviewers. Since it has been written at the very beginning of the project, the terminology partly has changed. The report is delivered as an appendix to this document (cf. Appendix B).

Table of contents

1 Introduction

In the context of Nepomuk- **The Social Semantic Desktop** - the **social** aspect is brought in mainly by WP5000: the social networking components are utilized to build and maintain topic- and content-specific interconnections between distributed individual workspaces. After the first deliverable of WP5000, which provided a first version of a prototype of community detection algorithm and (semi)-automatic metadata alignment, the aim of this deliverable is to provide a first version of a documented and tested prototype of a metadata sharing and recommendation application. The first two deliverables provide prototypes of several components which will make the semantic desktop richer in functionality related to social aspects. Finally, in the third deliverable an integrated prototype of social infrastructure and software will be delivered.

While in the first year of the Nepomuk project the focus has been put on the semantic aspects of the Desktop, in the second and third year WP5000 aims at providing social behavior to the Desktop, thus allowing the user to benefit from being part of a community. This deliverable exploits advantages of the social dimension in order to provide new ways of interaction and richer information sources for the user. Prototype applications for metadata sharing and for metadata recommendations are delivered. The user has several means of sharing and searching for metadata in the P2P community and her experience is improved with: i) metadata-based recommendation of resources, ii) metadata enrichment functionalities, iii) experts recommendation, and iv) tag recommendations.

More particularly, in this deliverable, we describe GridVine, the first metadata sharing system addressing simultaneously both scalability and semantic heterogeneity. Built following the P2P paradigm, it eliminates all central components and concentrates rather on bottom-up and decentralized processes. GridVine is a semantic overlay infrastructure built on top of a P2P access structure, which separates the logical layer – where data, schemas and schema mappings are managed – from a physical layer consisting of the structured P2P network supporting decentralized indexing, key load-balancing and efficient routing. Our system is totally decentralized, yet it fosters semantic interoperability through pair-wise schema mappings and query reformulation. In GridVine, heterogeneous but semantically related information sources can all be queried transparently using iterative query reformulation.

Moreover, in this deliverable we describe metadata recommendation solutions. We present a metadata-based recommendation of resources based on the similarity of the resources metadata in the P2P network. We present a metadata enrichment algorithm which create additional metadata for the desktop resources looking for similar items in the P2P network. We present a model for finding and recommending people who are experts on a given topic. We present algorithms for recommending tags to the user.

The deliverable is structured as follows. In Section 2 we describe the requirements and objectives of the metadata sharing and of the metadata recommendation component. Then, in Section 3 we illustrate the state of the art in the field of metadata sharing, metadata recommendation, people search, and tag recommendation. After this, in Section 4, we describe the architecture of the metadata sharing and recommendation components also describing the interaction with other Nepomuk components. In Section 5 we describe how the Metadata Sharing component is build on top of a P2P overlay providing global sharing and search support for metadata in RDF format. We elaborate on the decentralized nature of our approach for fostering semantic interoperability by introducing a bottom-up approach. In Section 6 we show our approach to provide recommendations to the user. In particular we describe methodologies for metadata enrichment, metadata based recommendation of files, people recommendation, and tags recommendation. Section 7 concludes the document summarizing and discussing the obtained results.

2 Requirements and Objectives

2.1 Metadata Sharing

In the Metadata Sharing component we concentrate on enabling global search for structured data on top of individual desktops. Leveraging on the P2P paradigm, the whole process is done in a bottom-up and totally decentralized manner.

While semantic web technologies have gained popularity to help organize large amounts of data, these solutions typically operate in predefined communities and do not enable data sharing outside the data's original boundaries. However there is a need for enabling structured data sharing and search in large scales. Since enforcing a global schema for structured data over large network of users is not possible, mappings between schemas of different users should exist and properly shared to enable semantic interoperability.

A good example of functionalities of the Metadata Sharing component can be demonstrated through a scenario similar to the one presented in D5.1 (For more information about the personas in this scenario see [52]). Kim and Alistair are among a large network of users who publish metadata about their digital assets through a decentralized infrastructure. This metadata is in RDF format, however each of Kim and Alistair as well as other users, either define and customize his/her own schema for the metadata shares or use an existing popular schema. To enable global search over their data, Kim and Alistair use some alignment generators to automatically produce some mappings among their schemas and some popular schemas and publish them as well. 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 his schema for computer gadgets is different of the one Kim uses. However a two way mapping between both Kim's and Alistair's schemas for computer gadgets and a popular schema for computer gadgets exist. If the system can route Kim's query, utilizing the published mappings between schemas, Alistair's data could also be searched and the search function could provide Kim with the answer he needs.

For the Metadata Sharing component we see the following requirements:

- Scalability: A large network of independent users with the need to share structured data in a decentralized manner.
- Semantic Interoperability: Independent end-users who can create both data and schemas to organize their information in customized ways and need to share this information outside their original boundaries.

Several recent research efforts concentrate on the above requirements, but they don't take both requirements into consideration at the same time. In the Metadata Sharing component we address simultaneously both scalability and semantic interoperability and set the following as our objective:

- utilize the P2P access structure as a decentralized and self-organizing media to support higher-level semantic services
- eliminate all central components and concentrate rather on a bottom-up decentralized process to enable sharing structured data. Users should be able to provide pair wise schema translations and share these translations.

- devise techniques to discover translation paths in the graph of translations, to support query routing to other semantic domains by applying available translations, but also to assess the semantic quality obtained from these translations.

2.2 Metadata Recommendation

While content based recommendations of resources and recommendations of simple data items have been investigated, functionalities such as recommendations based on metadata and recommendation of people are still missing to the Desktop user. In this section we first describe the scenarios from which we extract the users' requirements helping us in formalizing the objectives.

In the scenario Dirk: Deal with e-mails (see [38] section 3.2.2), "Dirk adjusts his mailbox to focus on topics of his interest". When Dirk enters the topics, the tag recommender can help him by suggesting related tags. This way Dirk can enrich his topic list to not miss mails he is interested in.

In another scenario, involving Dirk, Hans and Peter (see [38] section 3.2.2), Dirk wants to make a paper accessible to other project partners because he finds it relevant for their work. Hence, he makes it available and adds keywords to it. While he enters the keywords, the annotation tool asks the Metadata Recommender for tags related to the keywords Dirk has already entered. It shows those additional tags, such that Dirk can add them — either by clicking on them or by using the autocompletion feature. While Dirk has already entered the tag "Graph Topology", the Metadata Recommender returned (besides other tags) also the tag "Self Organizing Network", which Dirk decides to add to the paper. Since Hans has subscribed to the tag "Self Organizing Network", he is notified that the paper is available now. Hans retrieves the paper and also the annotations of it. Since he is not satisfied with the available metadata for this resource, he would like to extend it. Therefore, he queries the network for similar metadata descriptions of resources and receives back a list of metadata descriptions. He selects some of the entries and adds them to the annotations of the considered paper updating his annotation of it.

The scenario Claudia: Writes Deliverable (see [38] section 3.2.3) describes that "...all data, figures and documents are accessible and Claudia can easily have a deeper look into them.". Claudia searches her community by selecting the previous deliverable report for the project. After Claudia selected it, the system asks other users in her community (e.g. Dirk, who also works in the same project) for documents having similar metadata descriptions. The peers in his community, including Dirk, send then the shared metadata descriptions of the documents having similar annotations back to Claudia. Claudia can then request directly the documents she is interested in from the owner of the document in order to compile the deliverable.

In the SAP scenarios, Ambrosia is mostly working with coordination of the acquisition process for projects. For this, she needs to "find the right person for the right task" (see [38] Section 3.2.5).

We can conclude from those scenarios that there is a need for recommendations in several situations when working on and with the social semantic desktop. A user annotating or searching for a resource will certainly appreciate metadata suggestions. This speeds up the annotation process and allows for more efficient retrieval. Hence, we need a component which recommends metadata for given resources of a user. On the other hand, recommendations can help the user to find new resources she might find interesting, as described in the "Claudia: Writes Deliverable" scenario. This can be accomplished by a recommender system which on request or automatically recommends those resources to the user while she is looking for interesting and useful resources. Also recommendations of people who are experts on certain

topics would be useful for Ambrosia.

Therefore, recommender components on the social semantic desktop need to address the following types of recommendations:

1. informational resources,
2. tags, and
3. people.

These tools will thus allow both metadata exchange between community members and recommendations based on the items shared by other similar users.

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 D5.2. First, we present an analysis of what has been done in the area of Peer Data Management Systems, then we proceed with analysis of what has been done in the field of metadata recommendation and people recommendation. We conclude with an overview on publications relevant to the recommendation of tags.

3.1 Metadata Management and Sharing

There is a long tradition in data integration of pursuing the design of systems and methods that allow transparent access to disparate and heterogeneous systems through a single interface. Federated Databases were developed towards that goal. They allow the retrieval of data from multiple noncontiguous databases with a single query, even when the constituent databases are heterogeneous. Thus, federated databases provide solutions to integrate data coming from heterogeneous databases interconnected via a computer network. They come in different flavors (see Sheth and Larson [67] for a taxonomy) but often revolve around a central mediator [69] component, storing a global schema. The mediator is responsible for reformulating the query in terms of all the other schemas used by the individual databases.

With the explosion and decentralization of information production, however, it rapidly became clear that this centralized approach – requiring the definition of a central, global schema – could not be enforced in the large. The way GridVine semantically gossips the queries from one schema to the other is typical of a new generation of data integration systems called Peer Data Management Systems (PDMSs). PDMSs emerged as an attempt to decentralize the mediator architecture and allow the systems to scale gracefully with the number of heterogeneous sources. They do not require the definition of a global schema as they consider loosely-structured networks of mappings between pairs of schemas to iteratively disseminate a query from one database to all the other related databases.

Research on PDMSs is developing in several compelling directions. The complexity of iteratively reformulating queries to reach distant and heterogeneous databases in a PDMS is studied in the context of the Piazza [40] project. Hyperion [7] is a system inspired by the Local Relational Model [15] mapping data at both the instance and schema levels to enable global search capabilities in decentralized environments. SomeWhere [4] is a PDMS offering reasoning services through a distributed consequence finding algorithm. SQPeer [50] proposes a publish-subscribe mechanism and several compile and run-time optimization techniques to execute query plans in decentralized Semantic Web environments.

Our own efforts in GridVine focus on scalability and efficiency through the use of a structured overlay layer and on query diffusion based on probabilistic analyses of the mappings to determine the correctness of the semantic routes in the network [28].

3.2 Metadata(-based) Recommendation

In the past there has been no investigation on the possible algorithms for recommending metadata in a social infrastructure. However, some relevant work can be found in other related areas, such as that e.g. of meta-search engines. An overview of them can be found in [61]. General ranking in social

networks has been investigated e.g. in [37]. The area of recommendations for music in platforms like e.g. iTunes [5] or for diverse resources as books, DVDs and CDs at Amazon [6] bases the decisions on analyzing user behavior (e.g. past visited or bought items). This is a different scenario as the one we consider in this deliverable, since we consider the metadata annotations of the resources to compute similarity and thus deciding about the resources to recommend.

Several techniques applied in different fields can be reused and adapted to the recommendation scenario based on similarities of metadata descriptions considered in this work. For example, many algorithms have been developed in the field of databases for data cleaning and integration. This techniques aim at detecting that two different records represent the same real world thing, by analyzing their attributes or they schemas. Many of them are based on costly text similarity techniques in combination with strategies to lower the required number of comparisons. Several approaches exist and they are different approaches, going for lower processing times or better matching quality as can be seen for example in [5, 39, 20]. Similarly, also the information about the relations between documents and the resulting graph structure has been employed to match data from different sources [46, 48].

Much of the described work use a text similarity function as one evidence for matchings. In [23], the authors show that a Soft TFxIDF similarity function based on the Jaro-Winkler text similarity measure is the most appropriate for this type of matching. This is one of the approaches we build upon for the first recommendation prototype.

In Section 6 we will present the details of the specific scenario we consider and explain the approach we follow.

3.3 Recommendation of People

In this section we briefly describe already existing approaches for finding a ranked list of persons that are experts on a requested topic. The topic of recommending human experts on a given topic is a relatively new one, some first interesting Expert Search (ES) systems have been proposed in the last years and some initiatives have been started [74]. These systems use different information sources and features such as social network information [17], co-occurrences of terms and changes in the competencies of people over time [76], rule-based models and FOAF[7] data [53]. For the web, a different context from the enterprise search one, one of the approaches proposed in [72] focuses on scenarios like Java Online Communities where experts help newcomers or collaborate with each other, and investigated several algorithms that build on answer-reply interaction patterns, using PageRank and HITS authority models as well as additional algorithms exploiting link information in this context.

Most similar system to our approach is the Enterprise PeopleFinder [59, 60] also known as P@noptic Expert [26]. It first builds a candidate profile attaching all documents related to that candidate in one big document giving different weights to the documents based on their type. The system uses the document terms as topics of expertise and candidate name matching (i.e. whether a name appears into the document or not) for connecting documents and experts. Relationships between candidates and documents are binary, a given document is either related to a candidate or not.

A interesting distinction has been made between expert finding and expert profiling in [11]. The former approach aims at first retrieving the documents relevant to the query and then extract the experts from them. The latter approach builds a profile for each candidate and then matches the query with

[5] http://www.apple.com/itunes/

[6] http://www.amazon.com/

[7] http://www.foaf-project.org/

the profiles without considering the documents anymore [12]. Our model merges these two approaches: it builds a weighted profile for each candidate and keeps in the model all the documents without losing information.

All systems mentioned up to now use different ad-hoc techniques but do not formally define a retrieval model for experts. Some first steps in this direction have been made: probabilistic models [36] and language models [8, 9, 10] have been proposed. Our work continues this line of work, and shows how to model expert search based on a vector space based model. The advantage of building on this model is that, as we will show here, we can easily include existing VSM based refinements and solutions to improve expert search, and to query both for documents and candidates in a uniform way.

Another model for ES proposed in [56, 57] views expert search as a voting problem. The documents associated to a candidate are viewed as votes for this candidate's expertise also including relevance feedback techniques. Again, the relationships between candidates and documents are only binary and not continuous.

An important step in expert search is to extract the names of the candidates from the documents in the collection and match them with the list of the given candidates (e.g. the list of employees in a company). Possible solutions to the problem of measuring similarity between two named entities are presented in [22]. How to pre-process a document collection in order to extract names from documents such as email has been proposed in [18].

3.4 Recommendation of Tags

General overviews on the rather young area of folksonomy systems and their strengths and weaknesses are given in [42, 55, 58]. In [62], Mika defines a model of semantic-social networks for extracting lightweight ontologies from del.icio.us. Recently, work on more specialized topics such as structure mining on folksonomies—e. g. to visualize trends [34] and patterns [65] in users' tagging behavior—as well as ranking of folksonomy contents [44], analyzing the semiotic dynamics of the tagging vocabulary [19], or the dynamics and semantics [41] have been presented.

The literature concerning the problem of tag recommendations in folksonomies is still sparse. The existent approaches usually lay in the collaborative filtering and information retrieval areas. AutoTag [63], e.g., is a tool that suggests tags for weblog posts using information retrieval techniques. Xu et al. [71] introduce a collaborative tag suggestion approach based on the HITS algorithm [49]. A quality measure for tags, derived from collective user authorities, is iteratively adjusted by a reward-penalty algorithm. Benz et al. [14] introduce a collaborative approach for bookmark classification based on a combination of nearest-neighbor-classifiers. There, a keyword recommender plays the role of a collaborative tag recommender, but it is just a component of the overall algorithm, and therefore there is no information about its individual effectiveness. The standard tag recommenders, in practice, are services that provide the most-popular tags used for a particular resource. This is usually done by means of tag clouds where the most frequent used tags are depicted in a larger font or otherwise emphasized.

The approaches described above address important aspects of the problem, but they still diverge on the notion of tag relevance and evaluation protocols used. Xu et al. [71], e.g., present no quantitative evaluation, while in [63], the notion of tag relevance is not entirely defined by the users but partially by experts.

In this deliverable we will describe the technique we implemented to recommend tags to users of Nepomuk (cf. Section 6.5) and will compare it with state-of-the-art recommender systems. We will focus on a quantitative evalu-

ation which shows that our approach can outperform common recommendation algorithms and therefore is a good candidate for recommending tags in Nepomuk.

4 Metadata Sharing and Recommendation Architecture

The knowledge exchange and the social aspects of Nepomuk are mainly examined by WP5000. With the Deliverable 5.2 following the Deliverable 5.1 [32], we provide the next steps towards the social services of Nepomuk. In this section we present first the general architecture and interactions of the WP5000 components and then we present in detail the architecture of each component delivered in D5.2.

In Figure 1 the interaction between WP5000 components is presented. As described in the legend, a straight line connecting two components shows that our two components exchange data in order to interoperate: it is a necessary condition for them to work; while the dotted line shows optional exchange, for example, the Metadata Recommender might receive information from the Social Ranker. The inner box depicts the components delivered with D5.1 (i.e., the Community Manager and the Metadata Aligner). The D5.2 box shows the status of WP5000 components in this deliverable, and how these components interact with other semantic desktops using the distributed search infrastructure provided in WP4000. The Social Ranker component will be the focus of WP5000 in the future. It will receive input from the Community Manager to generate community-specific rankings. As seen in the diagram, there exists also an optional connection to the Metadata Recommender. The ranker can profit from recommendations made to enhance metadata as well es from recommendations for query extension. The Messaging infrastructure is delivered by WP6000 in D6.2.

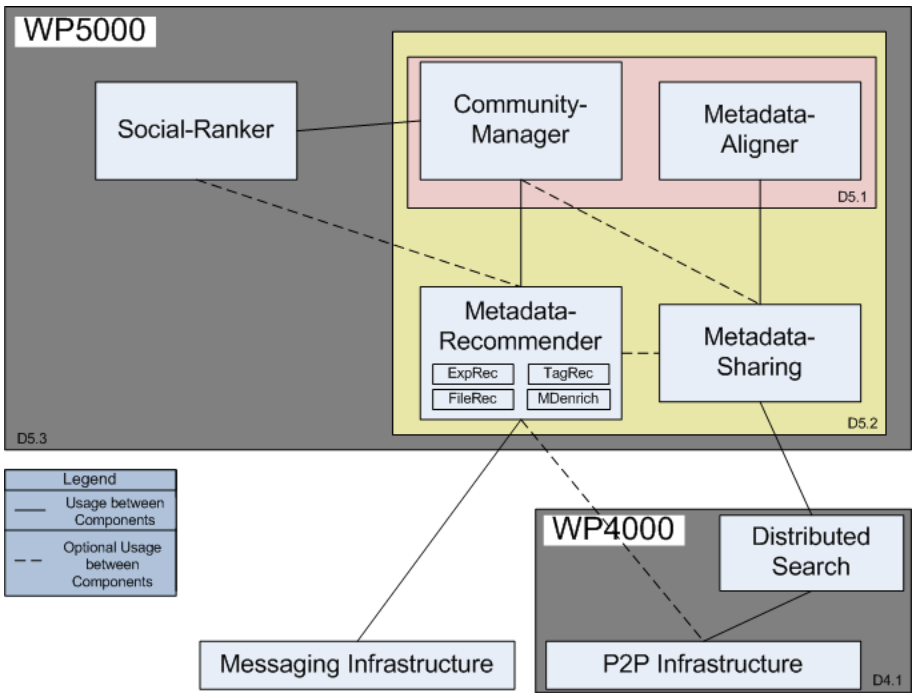


Figure 1: Interaction of components from WP5000 and their affiliation to the deliverables

4.1 Metadata Sharing

The Metadata Sharing component is responsible for enabling global search over all the data which users wish to share and for fostering semantic interoperability. 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.

4.1.1 Service Description

The MetadataSharing component, provides a service for inserting, deleting and searching metadata in RDF format. This data can represent either an RDF instance, an RDFS schema or a simple OWL mapping relating semantically similar classes and properties from two different schemas using owl:equivalentClass and owl:equivalentProperty properties. The three main services of this component can be summarized as follows:

- insert: this is the method used for inserting triples in the system to be shared among all users
- delete: this is the method for deleting an existing triple from the system
- search: this is the main service provided by the metadata sharing component performing a search among all shared triples. It takes a search criteria (at least one part of the triple (subject, predicate, object)), and uses the underlying overlay network and the fact that each triple is indexed three times to retrieve all data triples corresponding to this criteria.

The WSDL file describing the functionality of the Metadata Sharing component is available in the Nepomuk SVN Server in the directory */repos/trunk/java/org.semanticdesktop.nepomuk.services.metadatasharing*

4.1.2 Architecture of Metadata Sharing

The Metadata Sharing component is a semantic overlay built over a distributed hash table (DHT), enabling global search. In the semantic overlay RDF, RDFS and OWL managers are modules responsible for inserting and deleting RDF instances, schemas, and mapping between those schemas. These modules access the insert handler of the responsible peer for those data through the DHT. The data is then indexed and stored in the DHT Local store of that peer, enabling global search over this data. Since we are using a light version of P-Grid as our underlying DHT, which does not provide storage functionalities, the network database is also managed by the Metadata Sharing component. A query (atomic or conjunctive) is directed to the search manager module which accesses the query handler of the peer responsible for the data queried for, again through the DHT. The query handler accesses the network database to retrieve data. The queries can be propagated using the mappings indexed in the network database so that relevant data annotated according to a different schema can also be retrieved. A simple view of the Metadata Sharing components is shown in Figure 2. In this figure, the solid lines show local accesses to different modules of the Metadata Sharing component, while dashed lines show accesses which are directed by the DHT to the corresponding module of the responsible peer.

Relation to other components

In Nepomuk, tasks are distributed among components and our Metadata Sharing components is dependent on some services provided by other components, while it provides services to some other ones. Here we describe the Metadata Sharing component's relation to other Nepomuk components.

RDF Repository

The RDF repository is responsible for storing local RDF data. If the user wants to share its data globally and enable global search over this data, it can insert those data to the Metadata Sharing component. So the Metadata Sharing

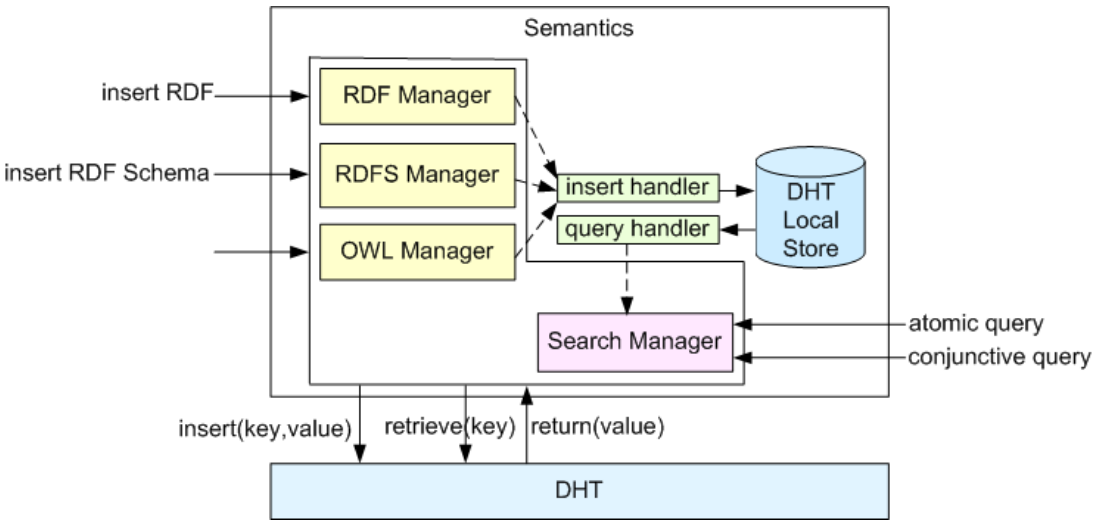


Figure 2: Architecture of Metadata Sharing component- solid lines show local accesses, dashed lines show accesses which are directed by the DHT to a module.

component can get the local data which each user wishes to share, from the RDF Repository.

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

Metadata Alignment To foster semantic interoperability and provide global search, the Metadata Sharing components may need to have some mappings between schemas inserted to it. It can use the functionality of the Metadata Alignment component to ask for a mapping to be created between two specific RDF schemas.

Distributed Index The Metadata Sharing component builds over a light version of P-Grid, the distributed index component, as its underlying DHT.

Components which need data shared by users Components like the Community Manager or Metadata Recommender, which need data as input for analyzing it, can use the service provided by the Metadata Sharing component to do global search for data.

4.2 Metadata Recommendation Architecture

The Metadata Recommendation components are responsible for providing recommendation functionalities to the Nepomuk user. All these are based on the metadata generated by other components in the semantic desktop and stored in the local RDF repository or via the Metadata Sharing component described in this deliverable. The services which provide recommendations are:

- Metadata based Recommendation of Files
- Metadata Enrichment
- Recommendation of Persons
- Recommendation of Tags

In this section we give an overview of the methods these components provide, their integration into the Nepomuk architecture, and their relation to other Nepomuk components.

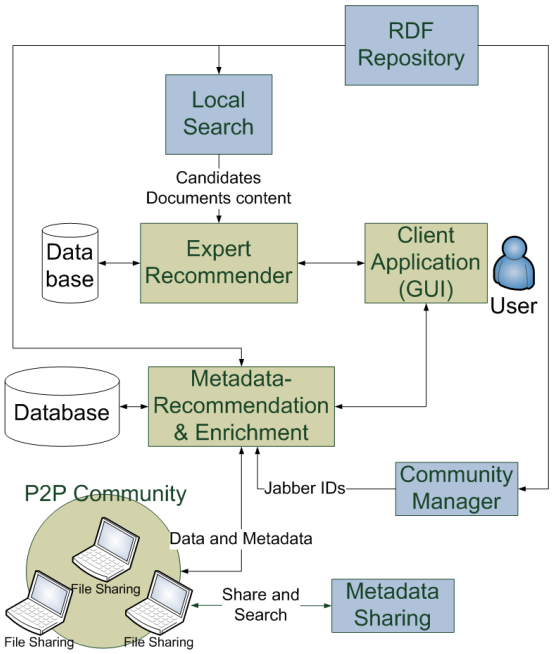


Figure 3: Architecture of Metadata Recommendation components

The first two services provided are the Metadata based Recommendation of Files and the Metadata Enrichment. From the architectural point of view these two services are highly correlated and, thus, are represented in Figure 3 as a single box (i.e., Metadata Recommendation & Enrichment). The goal of this component is to provide the user with recommendation of new interesting files and new interesting metadata for the resources already present on the desktop. In order to do so, the component needs to communicate both with the local RDF repository and with remote peers in the community via the communication infrastructure (i.e., it is using the messaging service to communicate with peers). The community in which to search for metadata can be provided as a list of users by the Community Manager component (see Deliverable 5.1 [32]).

The service of the Expert Recommendation aims at providing the user with a list of experts (i.e., people) on a given topic. The experts are selected among a list of persons referral to in the desktop. In order to do so, the component (i.e., Expert Recommender in Figure 3) needs to extract, out of the RDF Repository, some information about the content of documents and emails and also a list of expert candidates. These functionalities are provided to the Expert Recommender by the Local Search component which provides a simplified access to the repository.

The Tag Recommendation service builts upon the FolkPeer of the Community Manager component, introduced in Deliverable 5.1 [32]. We have extended the FolkPeer to supply tag recommendations based on the analysis of the folksonomy graph structure (cf. Section 6.5). Thus, FolkPeer now provides tag recommendations which are made available to components via the Metadata Recommender component introduced in this deliverable.

4.2.1 Service Description

The services provided by the Metadata Recommendation & Enrichment are exposed in the corresponding interface. The available methods in this API are:

- recommendSimilarMetadataOfResource(resourceUri)
- recommendAdditionalMetadata(resourceUri)
- recommendTags(resourceUri)
- recommendTags(tag)

The client applications request the recommendation of additional metadata (i.e., Metadata Enrichment) or the recommendation of similar resources (i.e., Metadata based Recommendation of Files) by providing the URI of a resource. This URI is used for retrieving the metadata of the resource from the local repository, and this metadata is used for the computation of the recommendations. Similiarly, tag recommendations can be retrieved for resources by providing the resource URI to the tag recommender. It is also possible to ask the tag recommender for related tags for a given tag by giving this tag to the recommendTags method instead of a URI.

The service provided by the Expert Recommender component can be accessed with three different inputs. The interface of this component contains three methods which are:

- findPeople(text)
- findSimilarPeople(personUri)
- findPeopleFromDocument(documentUri)

The resources (i.e., persons) retrieved by this functionality (i.e., Recommendation of Persons) are presented to the client application as a ranked list of results which are described by an URI. The different ways in which is possible to tailor the recommendations are some input text, or another person, or the content of a document.

4.2.2 Relations to other components

In Nepomuk, tasks are distributed among components and our Metadata Recommendation components depend on services provided by other components, while it provides services to some other ones. Here we describe the Metadata Recommendation components' relation to other Nepomuk components.

RDF Repository The RDF repository is responsible for storing local RDF data. All the recommendations provided by the Metadata Recommendation components are built based on the content of the RDF Repository. Also in the scenario of metadata enrichment, the final recommendations are originally coming from the RDF Repository of remote peers.

Middleware The Nepomuk middleware manages the registration and calling of all services provided in Nepomuk. The metadata sharing components register themselves to the middleware at startup enabling other components to find and use them.

Distributed Index The Metadata Recommendation & Enrichment component uses the P-Grid functionalities in order to communicate with remote peers from which to get metadata for recommendations.

Metadata Sharing In order to gather data as input for analyzing it, the Metadata Recommender uses the service provided by the Metadata Sharing component to do global search of data.

Community Manager The Metadata Recommender uses the FolkPeer included in the Community Manager component (cf. Deliverable 5.1 [32]) to gather information from the network of peers and to calculate tag recommendations on the metadata graph built on the FolkPeer.

Components using the recommendations Other than general Nepomuk end-user client applications, several components

are candidates for including recommendations generated by the Metadata Recommender: for example the Kaukolu Wiki (i.e., a semantic wiki with semi-automatic metadata creation, see [51] Section 3.3), when navigating wiki pages, can recommend tags or additional metadata for a page; the Personal Task Manager (the first prototype will be delivered in "D3.2 First Task Management Prototype"), when the user looks for appropriate tags to add to a task, or when she looks for experts to invite for a meeting, can use the recommendation components to provide appropriate recommendations.

5 Metadata Sharing

5.1 Overview

GridVine [2] is a semantic overlay infrastructure based on a peer-to-peer access structure. Built following the principle of data independence, it separates a logical layer – where data, schemas and schema mappings are managed – from a physical layer consisting of a structured peer-to-peer network supporting decentralized indexing, key load-balancing and efficient routing. Our system is totally decentralized, yet it fosters semantic interoperability through pair-wise schema mappings and query reformulation. In GridVine, heterogeneous but semantically related information sources can all be queried transparently using iterative query reformulation. We discuss a reference implementation of the system and several mechanisms for resolving queries in a collaborative fashion. Promoting both scalability and interoperability, GridVine provides a complete solution for building higher-layer distributed knowledge management applications.

In the following, we describe GridVine, the first system addressing simultaneously both scalability and semantic heterogeneity. Built following the Peer-to-Peer (P2P) paradigm, it eliminates all central components and concentrates rather on bottom-up and decentralized processes. Instead of requesting services from centralized servers, the participating Nepomuk desktops collaboratively contribute resources to support higher-level semantic applications. This ensures graceful scalability, as new clients entering into the system can in turn provide resources and act as servers intermittently to support shared services. The self-organizing and decentralized P2P access structure supports several higher-level semantic services: (i) distributed search, (ii) persistent storage, (iii) and semantic integration. The higher-level services recursively use the underlying P2P access structure to locate relevant information and distribute the load of the machines in a dynamic way.

A general overview of our infrastructure is given below, with details on both the P2P access structure used for indexing information and routing messages and the higher semantic layer responsible for managing structured content. Integration of data being central to our approach, we also detail decentralized information integration based on pair-wise schema mappings between heterogeneous but semantically related schemas. We discuss distributed query resolution mechanisms and illustrate the performance of our infrastructure deployed in situ on several hundreds of machines scattered around the world.

A first prototype of GridVine had been initially designed for the research purposes of LSIR/EPFL. Nevertheless, this was only a very preliminary version with several functional limitations and incompatible with the advancements and the identified requirements in Nepomuk. During this work, we have addressed many of the initial shortcomings and integrated the latest system with the components developed in WP4000 and WP5000. In the following section, we describe the latest state of the GridVine architecture.

5.2 GridVine: A Three-Tier Semantic Overlay Network

A key aspect of our approach is to apply the principle of data independence by separating a logical from a physical layer. This principle is well-known from the database area and has largely contributed to the success of modern database systems, by opening the door to optimization of logical higher-level primitives at the physical layer. In the present case, we generalize Codd's notion of data independence to networked environments beyond storage systems by separating a logical semantic layer – responsible for structured data storage, integration and query resolution – from a self-organizing P2P infrastructure –

liable for indexing, load-balancing and efficient routing.

Figure 4 gives a conceptual overview of our architecture. The base layer, called Internet Layer in the figure, represents the various machines connected to the Internet (Nepomuk desktops) and sharing structured information through our infrastructure. These machines self-organize into a structured P2P overlay layer for efficient routing of messages and index load-balancing. We use P-Grid (more information in deliverables D4.1 [30], D4.2 [29] and a short summary in the following section) to arrange the Nepomuk desktops into a virtual binary search tree at the overlay layer. Finally, the semantic mediation layer sits on top of this architecture and takes advantage of the overlay layer to efficiently share and integrate structured information across the network.

Structured information (data, schemas, mappings) is managed and stored through a database at the semantic mediation layer, but indexed and load-balanced by the overlay layer. Distinguishing the semantic mediation layer from the overlay layers allows us to offer higher level primitives at the uppermost layer, while benefiting from an efficient, decentralized and load-balanced access structure maintained by the P2P overlay. Note that the three layers are uncorrelated: the organization of the machines at the Internet layer is independent of the organization of the Nepomuk desktops (peers) at the overlay layer, which is itself dissociated from the structure of the information at the semantic layer. We describe both the overlay and the semantic mediation layer in more detail in the following.

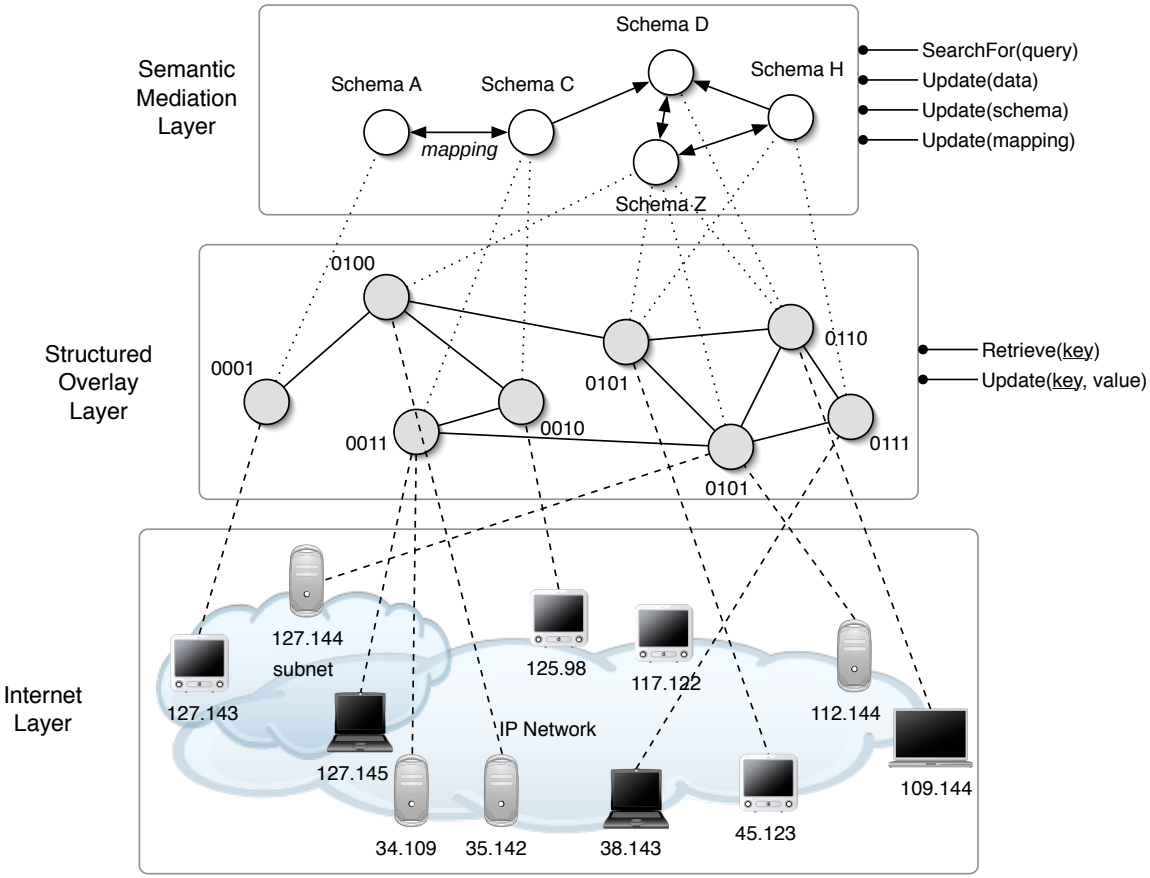


Figure 4: The GridVine Semantic Overlay Network: in our architecture, the semantic mediation layer is indexed using a P2P overlay layer, which is itself built on top of the Internet layer.

5.2.1 Organizing Peers and Load-Balancing the Index at the Overlay Layer

GridVine takes advantage of the P-Grid P2P access structure[8] at the intermediate overlay layer. P-Grid is a self-organizing and distributed access structure, which associates logical peers representing Nepomuk desktops in the network with keys from a key space representing the underlying data structure. Each peer is responsible for some part of the overall key space and maintains additional routing information to forward queries to neighbouring peers. As the number of machines taking part in the network and the amount of shared information evolves, P-Grid peers opportunistically organize their routing tables according to a dynamic and distributed binary search tree.

Each Nepomuk desktop (peer) $p \in \mathcal{P}$ is associated with a leaf of the binary tree. Each leaf corresponds to a binary string $\pi \in \Pi$. Thus, each peer p is associated with a path $\pi(p)$. For each level of the tree, each peer stores references to some other peers that do not pertain to the peer's subtree at that level, which enables the implementation of prefix routing for efficient search. The data items that are shared in the system are all indexed by keys. The key $key(d)$ of a data item d is generated using an order-preserving hash function $Hash()$. Each peer is responsible for storing the keys falling under its current key space $key \in \pi(p)$. The partition of the key space is load-balanced [3] in such a way that all peers are responsible for the same amount of data, irrespective of the actual distribution of the keys. In addition, peers also maintain references $\sigma(p)$ to peers having the same path, i.e., their replicas that duplicate their content, to ensure persistence of storage and resilience to network churn.

P-Grid supports two basic operations: $Retrieve(\underline{key})$ for searching for a certain key and retrieving the associated data value and $Update(\underline{key}, value)$ for inserting, updating or deleting keys. Different values can be associated with the keys depending on the application. In the context of GridVine, key values either point to unstructured content (e.g., images) managed by the overlay layer, or to structured data managed by a database at the semantic layer. Since P-Grid uses a binary tree, $Retrieve(\underline{key})$ is intuitively efficient, i.e., $\mathcal{O}(log(|\Pi|))$, measured in terms of number of messages required to resolve a search request in a balanced tree. For skewed distributions, it has been shown [1] that due to the probabilistic nature of the P-Grid approach, the expected search cost remains logarithmic, independently of how the P-Grid is structured. As a result, P-Grid can be seen as a persistent and load-balanced index layer, which supports efficient key look-ups in a totally decentralized manner. For further information on P-Grid, please refer to D4.1 [30], D4.2 [29].

5.2.2 Sharing Information at the Semantic Mediation Layer

GridVine takes advantage of the efficient and distributed index structure maintained at the overlay layer to globally manage semantic information at the semantic mediation layer. We support various operations to maintain the semantic layer, including instance, schema and schema mapping insertion. Capitalizing on the popularity of Semantic Web standards, we provide those mechanisms within the standard syntactic framework of RDFS and OWL. This requires mapping semantic data and operations to the two operations provided at the overlay layer, and hence mapping semantic information to routable keys.

RDF stores information as triples t representing various statements; triples always take the following form:

$$t_i = \{t_{subject}, t_{predicate}, t_{object}\}$$

where $t_{subject}$(the subject) is the resource about which the statement is made,

[8] `http://www.p-grid.org/`

$t_{predicate}$ (the predicate) represents a specific property in the statement and t_{object} (the object) is the value (resource or literal) of the predicate in the statement. All resources in our system are identifiable through Uniform Resource Identifiers (URIs), which are created on-the-fly upon insertion when missing. A structured overlay network allows to implement application-specific addressing spaces. In our case, we introduce the specific URI schemes $pgrid$: for elements managed by the overlay layer (index keys, unstructured content), and $pgrids$: for structured elements managed by the semantic layer (instances, schemas, schema mappings). This does not exclude the use of other URI schemes in conjunction with P-Grid's specific ones.

We map each triple at the semantic layer to routable keys at the overlay layer in order to enable efficient and load-balanced indexing of information. The granularity of the index as well as the exact mechanism used for mapping information at the semantic layer to keys at the overlay layer are naturally of utmost importance since they both directly influence the query processing capabilities of the overall system. We want to support searches on individual statements and thus have to index each triple separately. As most RDF query languages are based on constraint searches on the triples' subject, predicate or object, we have to reference each individual triple three times, generating separate keys based on their subject, predicate and object values. Thus, the insertion of each triple t is performed as follows:

$$Update(t) \equiv Update(\underline{Hash(t_{subject})}, t), \\ Update(\underline{Hash(t_{predicate})}, t), Update(\underline{Hash(t_{object})}, t).$$

In that way, each triple is associated to three keys at the overlay layer. Update and delete operations can be implemented using the same mechanism, which explains the generic name (Update) we give to this primitive. Each peer p maintains a local database DB_p at the semantic layer to store the triples whose keys fall under its key space. Since RDFS statements can be written as ternary relations, the physical schemas of the local databases can all be identical and consist of three attributes $S_{DB} = (subject,\ predicate,\ object)$. The local databases support three standard relational algebra operators: projection π, selection σ and (self) join $\bowtie$.

GridVine also supports the sharing of schemas defining classes of resources and their related properties. Each schema is associated with a unique key and indexed in an atomic operation:

$$Update(RDF\_Schema) \equiv \\ Update(\underline{\pi(p) : Hash(Schema\_Name)}, Schema\_Definition).$$

where the logical address $\pi(p)$ of the Nepomuk desktop p posting the schema is concatenated to a hash of the schema name to create a unique key for the schema whenever necessary. As class and property definitions can also be written as triples in RDFS, we use the same database to store both triples and schemas.

5.3 Integrating Data at the Semantic Mediation Layer

GridVine's semantic layer allows peers to efficiently share knowledge in a global manner. Nepomuk desktops can query for any information at the semantic layer by issuing series of $Retrieve(\underline{key})$ operations on the corresponding keys at the overlay layer (see Section 5.4). Sharing information syntactically aligned as RDF triples in the network does not however ensure global interoperability. On the contrary, GridVine being a totally decentralized system, any peer in the network is free to come up with new schemas to structure its own information.

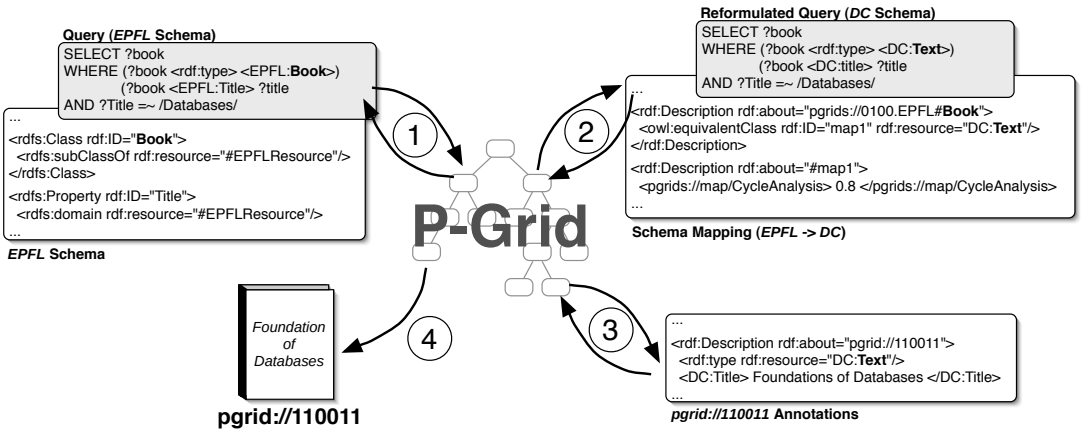


Figure 5: A simple example of query reformulation using a schema mapping.

To integrate all semantically related but syntactically heterogeneous information shared by the Nepomuk desktops, GridVine supports the definition of pair-wise schema mappings. A mapping allows the reformulation of a query posed against a given schema into a new query posed against a semantically similar schema. By iterating this process over several mappings, a query can traverse a sequence of schemas at the mediation layer and retrieve all relevant results, irrespective of their schemas. Given the provision of a sufficient number of mappings, GridVine fosters in that way global semantic interoperability in a totally decentralized fashion.

We encode schema mappings using simple OWL statements relating semantically similar classes and properties from two different schemas using owl:equivalentClass and owl:equivalentProperty properties. Schema mappings are indexed at the key space corresponding to the source schema at the overlay layer – or at the key spaces corresponding to both schemas if the mapping is bidirectional:

$$Update(Schema\_Mapping) \equiv Update(\underline{Source\_Schema\_key}, Schema\_Mapping).$$

Figure 5 shows a simplified example of query reformulation in GridVine; a peer issues a query to retrieve a book annotated with a given schema (EPFL schema) (1), reformulates the query thanks to a schema mapping (2), finds a relevant resource annotated with the second schema (3), and finally retrieves the relevant resource by querying the structured overlay layer (4).

5.4 Resolving Queries in GridVine

Indexing each triple based on three keys at the overlay layer enables us to resolve complex higher-level queries at the semantic layer. A triple pattern [66] is an expression of the form $(s,\ p,\ o)$ where s and p are URIs or variables, and o is a URI, a literal or a variable. The simplest semantic queries supported by GridVine retrieve information based on a single triple pattern:

$$SearchFor(x? \ :\ (s,\ p,\ o)),$$

where $x?$, the distinguished variable the query has to return, also appears in the triple pattern $(s,\ p,\ o)$. For instance, the following triple pattern query:

$$SearchFor(x_2? \ :\ (x_1?,\ pgrids://0100:EPFL\#Title,\ x_2?))$$

retrieves the $Title$ of all the images annotated with the $pgrids : //0100 : EPFL$ schema. We call such a query an atomic query since it only contains one triple pattern. In GridVine, atomic queries are resolved by first locating relevant peers thanks to the index provided at the overlay layer, and then by processing structured data handled at those peers at the semantic layer. A peer issuing an atomic query q first has to determine the key space $\underline{key}$ where it can find the answers. This can be determined by taking a hash of one of the constant terms $const$ in the triple pattern:

$$\underline{key} = \underline{Hash(const)}.$$

In our example, $\underline{key} = \underline{Hash(pgrids : //0100 : EPFL\#Title)}$. When two constant terms appear in the triple pattern, both can be used to retrieve the results. Once the key space is discovered, the peer simply forwards the query to the peer(s) responsible for that space using $Retrieve(\underline{key}, q)$. As all triples are indexed on their subject, predicate and object in GridVine, the query can be directly answered by the peer(s) responsible for this key space, which stores the corresponding triples in its database. Thus, atomic query resolution boils down to a standard P-Grid look-up generating $\mathcal{O}(log(|\Pi|))$ messages. Once arrived at its final destination(s) $\underline{key}$, the query is resolved with a local relational query on the local database DB_{dest}. Defining $pos(term)$ as the position of a term (variable or constant) in a triple pattern, i.e., $pos(term)$ either takes $subject$, $predicate$ or $object$ as value, the set of results $Results$ is obtained as follows:

$$Results = \pi_{pos(x)}\ \sigma_{pos(const)=const}\ (DB_{dest}).$$

In our example, the query is forwarded to the peer(s) responsible for $Hash(pgrids : //0100 : EPFL\#Title)$, which can retrieve the results by issuing a query $\pi_{object}\ \sigma_{predicate=pgrids://0100:EPFL\#Title}$ on its local database. Once retrieved by the destination peer(s), the results are sent back to the original issuer of the query. Conjunctive and disjunctive queries can be resolved in a similar manner [2], by iteratively resolving each triple pattern contained in the query and aggregating the sets of results through local join operations [54]. Queries on value ranges are also supported, as they can be natively processed by our overlay network.

5.4.1 Self-Organizing Mappings

GridVine's mediation layer allows the peers to share structured data in a scalable manner. Sharing information syntactically aligned as triples does not however ensure global interoperability. On the contrary, GridVine being a totally decentralized system, any peer in the network is free to come up with new schemas to structure its own data.

To integrate all semantically related but syntactically heterogeneous information shared by the peers, GridVine supports the definition of pairwise schema mappings. Mappings allow the reformulation of a query posed against a given schema into a new query posed against a semantically similar schema. By iterating this process over several mappings, a query can traverse a sequence of schemas at the mediation layer and retrieve all relevant results, irrespective of their schemas. Given the provision of a sufficient number of mappings, GridVine fosters in that way global semantic interoperability in a totally decentralized fashion.

GridVine allows for the definition of both equivalence and inclusion (subsumption) GAV mappings. For the sake of this demonstration, mappings relate semantically similar $predicates$ defined in different schemas. Queries are then reformulated by replacing the predicates with the definition of their equivalent or subsumed predicates (view unfolding). Schema mappings are inserted at the key space corresponding to the source schema at the overlay layer – or at

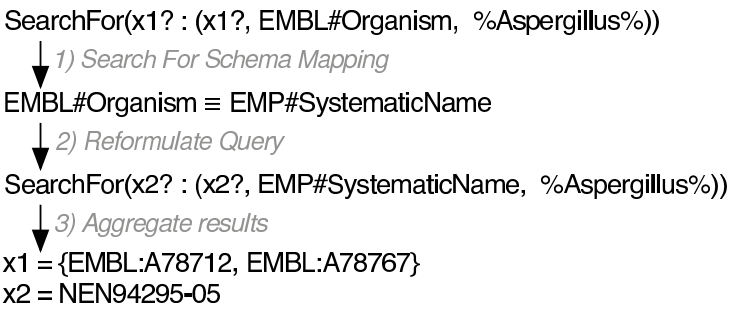


Figure 6: A simple example of query reformulation using a schema mapping.

the key spaces corresponding to both schemas if the mapping is bidirectional:

$$Update(Schema\_Mapping) \equiv Update(\underline{Source\_Schema\_Key}, Schema\_Mapping).$$

Figure 6 shows a simple example of query reformulation.

GridVine offers several functionalities to organize the network of mappings at the mediation layer in an automated way. The system ensures that the network of schemas and mappings at the mediation layer is connected in order to enforce global interoperability. It creates additional mappings whenever necessary, tries to assess the quality of the mappings and discards the mappings that are detected as being erroneous.

5.4.2 Connectivity at the Mediation Layer

GridVine maintains information about the graph of schemas and mappings. Upon inserting a new schema, GridVine asks for the manual definition of schema mappings between the new schema and some already inserted schemas. The system then periodically ensures that the network of schemas and mappings forms a strongly connected graph, such that a query posed locally using any schema can be disseminated globally to all related schemas using the network of mappings.

Repeatedly crawling a decentralized and potentially large graph of schemas connected by mappings would be costly in our setting. In GridVine, the peers determine the degree of connectivity of the mediation layer from the degree distribution of its schemas. Each peer storing a schema definition is responsible for updating the number of incoming and outgoing mappings attached to its schema:

$$Update(Domain\_Connectivity) \equiv Update(\underline{Hash(Domain)}, \{Schema, InDegree, OutDegree\})$$

where $Domain$ is the name of the application domain related to the mediation layer. The peer p responsible for $Hash(Domain)$ at the overlay layer can then locally derive the degree distribution of the graph of schemas by aggregating these numbers. It evaluates the connectivity of the mediation layer by computing a connectivity indicator ci_{domain}:

$$ci_{domain} = \sum_{j,k} (jk - k)\ p_{jk}$$

where p_{jk} stands for the probability of a schema to have in-degree j and out-degree k. $ci_{domain} \geq 0$ indicates the emergence of a giant connected component in the graph of schemas and mappings [27]. Thus, the mediation layer is not strongly connected as long as $ci_{domain} < 0$.

5.4.3 Creation & Deprecation of Mappings

Peers responsible for a schema periodically inquire about the connectivity of the mediation layer by issuing a query to the corresponding key space. $ci < 0$ indicates that some of the schemas shared at the mediation layer cannot always be accessed by following series of mappings. In that case, more mappings are needed to ensure global interoperability. This triggers the automatic creation of additional schema mappings to reinforce the existing network. The exact method used to choose the pair of schemas and to create the mapping depends on the application domain.

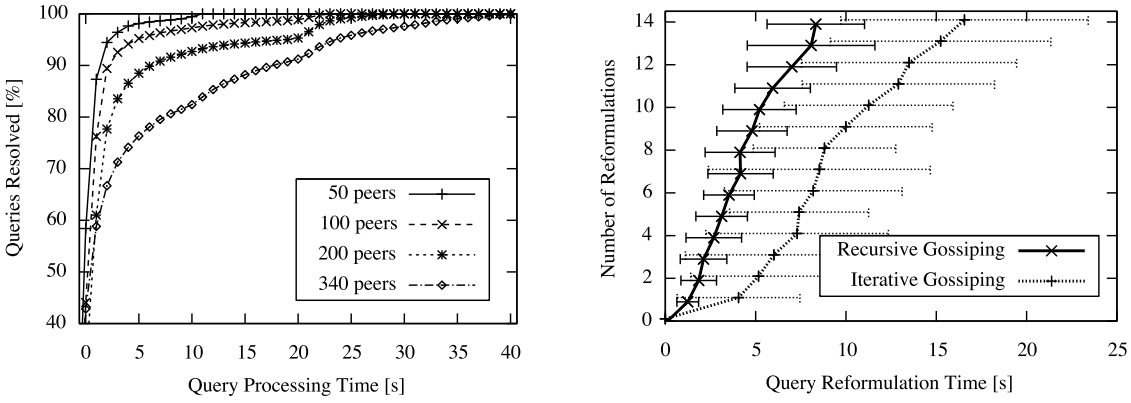
The quality of the mappings created in this way is periodically assessed as the networks of peers, schemas, and mappings evolve. GridVine uses a Bayesian analysis comparing transitive closures of mappings to assess the quality of the mappings [28]. The mappings manually created by the users are always considered as correct in this analysis, while probabilistic correctness values are inferred for mappings that were created automatically. A mapping detected as incorrect is marked as deprecated in the system, and is from then on ignored, both for the reformulation of the queries and for the connectivity analysis. The deprecation of mappings fosters the creation of a new topology of mappings, which will ensure the global interoperability of the system eventually.

5.4.4 Performance Evaluation

We carried out several large-scale experiments in various settings in order to validate the design of our infrastructure. We detail two of these experiments below. Figure 7 a) shows the distribution of query processing time in several large-scale networks of 50 to 340 peers scattered around the world. Each peer ran on a distinct machine on the PlanetLab network[9]. We inserted between 4000 and 80000 triples (depending on the network size) in the system and monitored the resolution of thousands of atomic queries issued by all the peers over several hours. For the largest network consisting of 340 peers, 43% of the queries were answered within one second only, and 75% within five seconds. Note that the machines used for the experiment were heavily loaded due to the processes inherent to the PlanetLab infrastructure and to several other experiments running concurrently. Despite the heavily skewed distribution of the keys generated when indexing the triples, the partition of the key space at the overlay layer, and thus the storage load at the semantic layer, remained balanced thanks to P-Grid proactive load-balancing mechanisms. Although several peers went offline during the test, all queries were answered properly due to the dynamic index replication triggered at the overlay layer.

Figure 7 b) illustrates a network-intensive deployment focusing on data integration. GridVine was this time deployed over 40 peers running on 20 cores in a local-area cluster and interconnected through a realistic network setting[10]. Each peer was setup to be responsible of a distinct schema. The schemas were related through a random graph of schema mappings in order to create a fully-connected semantic mediation layer. Several thousands of atomic queries were issued in various points of this network, and then reformulated in successive steps to be disseminated throughout the network. The figure shows the successive reformulation steps, averaged over the various queries and with confidence intervals set to 90% for up to 14 reformulations. Two approaches were tested: in iterative gossiping, the peer issuing the original query is responsible for retrieving all mappings and reformulating all queries by itself iteratively. In recursive gossiping, the reformulation process is iteratively delegated to those peers receiving reformulated queries. Recursive gossiping

[9] `http://www.planet-lab.org/` [10] ModelNet `http://modelnet.ucsd.edu/` network based on a client-stub topology, with 5ms delays and 200Kb/s links



a) **b)**

Figure 7: Two deployments of GridVine illustrating a) the cumulative distribution of query resolution time for networks ranging from 50 to 340 peers distributed globally and b) the reformulation steps for up to 14 query reformulations in a network 40 schemas stored on 40 different peers.

performs systematically better, as it distributes the reformulation load more evenly among the peers.

6 Metadata Recommendation

6.1 Overview

In this section we describe our methods for providing the user with different types of recommendations. Each of the following subsections present the details of the different recommendation types as presented in Section 2.2. We provide (i) recommendations of files or resources and (ii) recommendations for metadata enrichment based on the metadata of resources, (iii) recommendations of persons based on the content of the desktop, and (iv) recommendations of tags based on the folksonomy structure of the Nepomuk network of peers. Metadata enrichment describes the process of suggesting metadata extracted from resources to other resources which lack this metadata. In contrast, recommending tags provides user-generated annotations when the user is in a situation where he wants to enter tags by himself, e.g., during tagging a resource or when searching for a resource.

6.2 Metadata based Recommendation of Files

In this section we present the details of the metadata based recommendations of files or resources. The functionality provided is based on the scenario where the user wants to obtain, from the communities she belongs to, files which have similarity in their metadata descriptions (see Section 2.2). For this, she selects a local file which has a corresponding metadata description. This metadata description is then used for finding files in her own and in other desktops based on comparisons with their metadata descriptions.

Let us first introduce some general definitions used for the Recommendations of Files and for the Metadata Enrichment recommendations which is presented in Section 6.3:

- P_i is a peer in the community
- D is a document
- M is metadata expressed as an RDF graph
- p_i is a certain property appearing in M
- $m(D) := \{M | M \text{ describes } D\}$
  is the metadata describing document D
- $V_p^D := \{\text{values of } p | p \in m(D)\}$
  is the set of values of a certain property in M

The Recommendation of Files based on their metadata requires a series of steps and computations in order to find the expected resources. In the following each of the steps involved in this process will be explained in detail:

1. Analysis of the metadata of the selected resource: After the user selects a resource, its metadata is retrieved from the repository. Based on the discriminativeness information of each property at this peer, the properties are ranked and the top-K most discriminative properties with their values are selected for the next step. The discriminativeness value for each property is computed in regular intervals at each peer (each peer has its own values), by considering the existing property types in the metadata entries and the corresponding distribution of values. This is: the more the values of a certain property vary, the more they can be used for univocally identifying a resource; this is, the higher is the

discriminativeness of this property.

$$discr_{P_i}(p_j) := \frac{\text{number of distinct values of } p_j}{\text{number of all values of } p_j} \quad (1)$$

Note that the discriminativeness value of a property might be different in each peer of the network and it represents the strength of a property for identifying unequivocally a resource, for that specific user in that peer.

2. Query the Peers: A query is constructed with the top-K most discriminative properties contained in the metadata description and the corresponding values (a disjunction of property-values). The query is sent to peers in the user's community and also executed on the local storage of the user issuing the query. Metadata descriptions having any property-value matching with the query are retrieved for further processing. Querying a peer in the community involves also to compute the overlap of the query with each of the results at that peer (see next item).

3. Compute the Overlap: All results obtained from querying the local storage and the peers in the user's communities are analyzed. Each result is compared with the top-K property-values of the metadata of the originally selected file for finding matchings of corresponding values. This is performed using a string similarity function in order to allow partial matches (like the ones presented in [23]). The similarity measures are accumulated and normalized. After every property in the metadata result has been compared, a metadata overlap measure is computed by taking into account the discriminativeness of each matching property type, the accumulated similarity value and the number of compared properties. This is needed for considering the different annotation preferences of different users at different peers. In this scenario, the overlap measure computation procedure favors the resources that match better the property-values in the query. Using the discriminativeness, the overlap for the recommendation of resources can be defined as:

$$overlap_r(m(D_1), m(D_2)) := \frac{\sum_{p \in m(D_2)} discr_{P_2}(p) \cdot pvs(p, m(D_1), m(D_2))}{|p \in m(D_1) \cup p \in m(D_2)|} \quad (2)$$

$$overlap\_percentage_r(m(D_1), m(D_2)) := \frac{\sum_{p \in m(D_2)} discr_{P_2}(p) \cdot pvs(p, m(D_1), m(D_2))}{|p \in m(D_1)|} \quad (3)$$

where pvs is the property value similarity and it computes the similarity between the metadata of two documents D_1 and D_2 according to a given property p. This form of property similarity computation is used in the scenario where the recommendation targets to find similar resources. It is defined as

$$pvs(p_i, m(D_1), m(D_2)) := \frac{\sum_{v_j \in V_{p_i}^{D_1}} \max_{v_k \in V_{p_i}^{D_2}} (\{stringSim(v_j, v_k)\})}{\max(\left|V_{p_i}^{D_1}\right|, \left|V_{p_i}^{D_2}\right|)} \quad (4)$$

where $stringSim$ is a given string similarity function (e.g. the Jaro-Winkler measure [24]).

4. Rank the Results: The results obtained are ranked accordingly to the overlap measure. In addition to discarding results with very high overlap value, results with low overlap value are discarded as well. By doing this the files that are likely to be the same or that are likely to be non-relevant results are left aside. After this, the top-K most prominent results (considering the overlap measure) are returned.

To perform this tasks, the Metadata Recommendation requires accessing other Nepomuk components:

- The Community Manager needs to be accessed for finding the communities the user belongs to; and
- The RDFRepository at each peer in the community (which allows access) has to be queried for similar metadata, and each peer needs to keep statistics about this metadata e.g. the discriminativeness of properties.

The interactions with other components in Nepomuk can be seen in Figure 3.

6.2.1 Evaluation

The Evaluation of the Recommendation of Files and the Metadata Enrichment (see Section 6.3) has been performed using two peers with different, partially overlapping datasets.

The datasets used for the evaluation contain information about scientific publications: they contain metadata such as title, authors, publication place, publication date, keywords, etc. as available in DBLP[11], Citeseer[12] and other public available websites.

From an original pool of 600 publications from 1997 to 2007, three datasets were created, each of them containing metadata of 300 publications and corresponding to a different virtual user: Claudia, Dirk and Peter. The sets are overlapping to a certain extent, Claudia and Dirk have 149, Claudia and Peter have 84, and Dirk and Peter have 146 records in common. The publications where extracted mainly from the areas of cognitive informatics, interactive systems, software engineering, simulation and modeling, programming languages, computer graphics, information retrieval, network security, and computational biology. Each metadata description is identified with a different URI in each different virtual user's space.

For each different user, the publications were randomly selected from the pool of 600 available publications and were fixed for each of the users, thus representing an hypothetical desktop content. This yields three datasets with some overlap and with different naming schema. The fact that one metadata record corresponds to the same publication as another metadata record can only be seen by analyzing the local part of the identifier of each metadata record. These three different datasets were stored as different Nepomuk peers: Claudia's, Dirk's and Peter's. The client requesting the recommendations was run from one of the peers, and the others were accessed through the messaging functionality between peers available in Nepomuk. We consider to be Claudia the user asking for recommendations, so, always resources from Claudia are chosen for asking the system to recommend her with other files or additional metadata. Most of the previous works use text similarity functions as one evidence of the matching. It is shown in [23] that a Soft TFxIDF similarity function based on the Jaro-Winkler text similarity measure is the most appropriate for this type of matching.

The system-oriented experiments performed in order to evaluate the recommendation of files or resources used these three sets. The input to the system

[11] http://www.informatik.uni-trier.de/~ley/db/ [12] http://citeseer.ist.psu.edu/

are 16 identifiers (URIs) of different metadata records (a record is a set of triples describing the same resource). Each of this metadata record describes a different file in the dataset of Claudia. The 16 metadata records have the peculiarity that (i) a recommendation for the publication can be found in the ACM Digital Library (DL) [13], (ii) there exists at least a description of a publication in one of the three different user sets among the first 20 results of the "Find similar Articles" functionality provided by ACM DL. We looked into this first 20 results of (ii) and picked up all descriptions of publications (not equal to the one selected for giving similar articles) that are in at least one of the three user sets. For each of the 16 files selected from Claudia we stored this results, so, we knew which recommendations our system could give based on the recommendations the ACM DL gives for this selected file (see section 2.2 for details about the scenario). Then the system was asked to give File recommendations for these 16 selected publications. For each requested recommendation we consider: the recommendation found by our system to be the number of retrieved results; the expected maximum number of recommendations considering the top 20 ACM recommendation for each selected file to be the number of relevant results[14]; the recommendations that our system found among the top 20 ACM recommendations to be the intersection between relevant and retrieved results. Using these definitions, we computed the average effectiveness values for the 13 requests for which our system provided recommendations obtaining an average precision of 0.33 and an average recall of 0.5. This result shows that the developed system is able to find many of the relevant results available, but also that we need to improve the ranking in order to put at the top of the recommendations the most relevant results.

Additionally, we performed also a user-oriented evaluation for the recommendation of similar files. For this we took 10 files from Claudia and asked the system to compute recommendations. We analyzed the files for which the system found recommendations in two different ways:

1. analyzing the metadata available for each of this publications, and
2. analyzing the abstracts of the corresponding publications.

The scenario we have in mind is that the reader has read the original paper and wants to be recommended with additional interesting papers.

The judgment of 1) is pretty simple, here co-occurring strings in the values of metadata attributes yield an increase in the overlap measure between the metadata descriptions. The judgment was binary, it is relevant according to the analysis of the authors, publication title, conference, keywords, or any of the other metadata fields available, or it is not relevant. Here the result are 100 % correct.

The evaluation of 2) is more subjective, since the content of the abstracts has to be analyzed in order to state the relevance of the recommended publications to the selected one. Here we performed a three level judgment: not relevant at all, relevant, very relevant. The results obtained in these experiments are presented in Table 1.

For the 10 Files, 41 recommendations were given by the system in total. As it can be seen from Table 1, 49 % of the recommendations were very relevant publications, 18 % resulted in recommendations not related at all with the query, and the rest 33 % resulted in publications related to the selected one. These results gives us a recommendation rate of 82 % of relevant recommendations which we consider to be a good result.

[13] `http://portal.acm.org/dl.cfm` [14] Limiting the possible relevant results only at the top 20 ACM recommendations makes the precision and recall values low because in this way we consider only the highly relevant documents

	Not Relevant	Relevant	Very Relevant
User 1	9	12	20
User 2	6	15	20
Average	18 %	33 %	49 %

Table 1: File Recommendation Evaluation

6.3 Metadata Enrichment

This section presents the details of the recommendation of additional metadata for annotating one resource as presented in the scenario in Section 2.2. Imagine a user having partially annotated some resources and wanting to find additional metadata for this resources (e.g. missing authors, information related to publication place or date, etc.). The user poses a query to peers in his community and gets additional metadata entries to the ones that already exist for this resources. After the results have been presented, the user can select the entries of interest and add them to the metadata for each of this resources.

The steps for fulfilling this task are the same as the ones presented in Section 6.2, but the selection and computation algorithms are different. For a definition of the notation used, please refer to the definitions stated in Section 6.2. The overlap is again used for finding the most similar resources, but metadata descriptions having more entries than the query are favored because the goal is to add more entries to an existing metadata description. In this recommendation, the computation of the overlap, pvs is preceded by a metadata recommendation similarity expression which punishes query results which do not contain more information than the query. It is expressed as:

$$simMDReco(p_i, m(D_1), m(D_2)) := \frac{\min(\left|V_{p_i}^{D_1}\right|, \left|V_{p_i}^{D_2}\right|)}{\left|V_{p_i}^{D_1}\right|} \cdot pvs(p_i, m(D_1), m(D_2)) \quad (5)$$

So, the overlap for the scenario aiming to enhance metadata descriptions is computed as presented in Equations 6 and 7 .

$$overlap_e(m(D_1), m(D_2)) := \frac{\sum_{p \in m(D_2)} discr_{P_2}(p) \cdot simMDReco(p, m(D_1), m(D_2))}{|p \in m(D_1) \cup p \in m(D_2)|} \quad (6)$$

$$overlap\_percentage_e(m(D_1), m(D_2)) := \frac{\sum_{p \in m(D_2)} discr_{P_2}(p) \cdot simMDReco(p, m(D_1), m(D_2))}{|p \in m(D_1)|} \quad (7)$$

The remaining steps stay the same as already described in Section 6.2.

6.3.1 Evaluation

The setting of the peers and datasets for the evaluation of the Metadata Enrichment recommendation is the same as the one presented in Section 6.2.1.

Starting again from the three different sets for our users, each of the records in each of the user sets went through a deletion phase. In this deletion phase, 20% of the existing property-value pairs were randomly deleted. So, after this process we obtained three different sets, with some overlap in the publications described by the metadata, and with some differences due to the randomly applied deletion. For this evaluation, a set of 30 URIs from Mary's set were selected. Each of this URIs represents a metadata record that also exists in Peter's or in Dirk's (or in both) sets. Mary requested recommendations for metadata enhancement with these 30 selected URIs, and we computed the precision and recall of the obtained results. Then, the original set of Mary went through a deletion phase where 30% of the property-value pairs were randomly deleted and the same request for recommendations was performed again and precision and recall computed. This was performed iteratively, with deletions in Mary's set of 40%, 50%, and 60% in order to see when the deletion rate is too high in order to get good recommendations back. We performed this test several times, starting from the same fixed set of publications for each user, but with newly computed random deletions and computed the average values of precision and recall for the different deletion rates. It has to be mentioned that the system is not only finding matching resources so that its metadata description can be reused, but is also recommending only metadata descriptions that provide at least one more property-value pair to the metadata record that was selected for recommendations. This is, if the found description would not add any information to the selected one, then it is not sent back as a recommendation. The details of how the precision and recall vary depending on how much information was deleted in each of the different runs can be seen in figures 8(a) and 8(b).

From this detail we also computed the average values, which are presented in figure 9

The reader will notice that the precision and recall of our approach deminishes depending on the deletion rate of property-value pairs in the metadata records. This is the expected behavior since when more data is deleted, less information is available for finding resources.

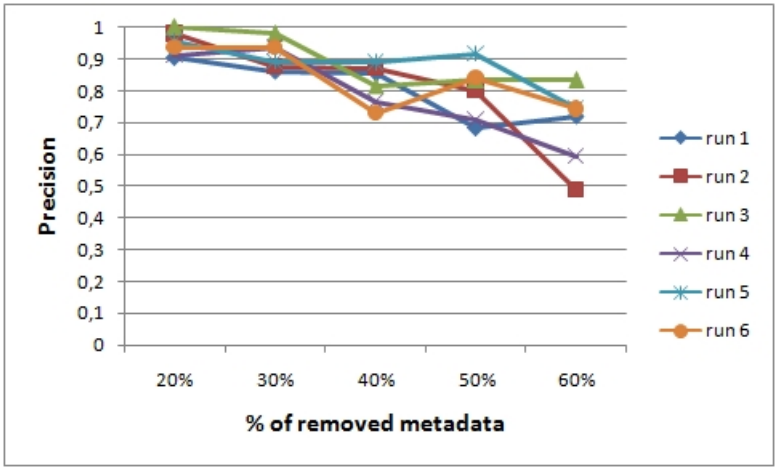
In addition, we performed a second evaluation for the Recommendations for Enhancing Metadata Descriptions by selecting other 15 publications from Mary. As a result the system recommended entries corresponding to 16 different metadata records. A screenshot of the selection of one publication can be seen in Figure 10 and its corresponding metadata recommendation can be seen in Figure 11. It can be noticed that the entries that are already present in the selected publication's metadata are not recommended, only entries with new values are recommended.

The results obtained show that out of the 16 obtained recommendations, 11 recommendations are accurate and correspond to the same originating publication, which means that they are correctly suggested and that the recommended metadata is metadata missing in the selected metadata record. From the remaining 5 not accurate recommendations obtained there is some evidence of matching, like same authors, same conference and dates, keywords, etc., but they describe different publications and should in general not be recommended. However, we noticed that in general, having overlap scores of 0.7 or above yield correct results. This will be considered for the future development of the recommendation techniques.

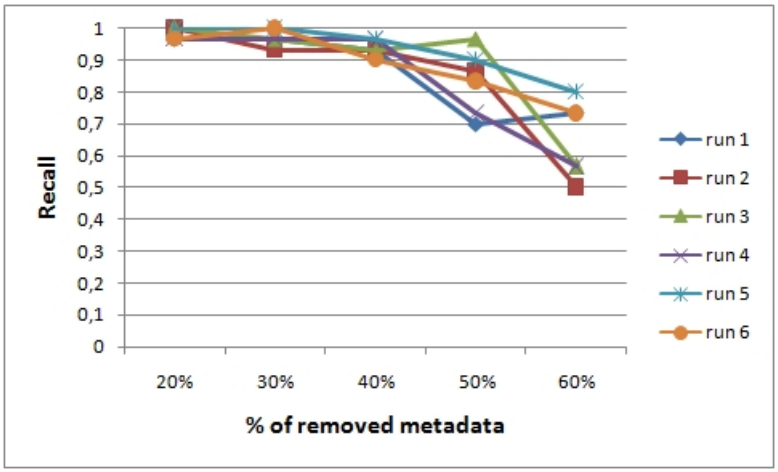
The overview of this second evaluation results are presented in Table 2.

6.4 Recommendation of Persons

The model proposed in this section builds on the well-known vector space model, and represents expert candidates as vectors in the same space to-



(a) Precision



(b) Recall

Figure 8: Metadata Enhancement Evaluation: Precision and Recall values for each run

	Number of Reco.	% of the provided Reco.
Accurate Reco.	11	73 %
Not Accurate Reco.	5	27 %

Table 2: Metadata Enrichment Recommendation Evaluation

gether with documents and queries, allowing us to retrieve relevant documents and experts with a single query. This allows us to re-use many techniques developed for information retrieval to improve expert search. The basic model is simple and easy to extend, e.g. using not only documents as expert evidence, but also prior knowledge of the user, personal opinion, time, geographical distance, or salary.

6.4.1 Expert Search: Problem Definition

We assume a collection of documents $D = d_1, \ldots, d_m$ and a list of expert candidates $C = c_1, \ldots, c_n$. Additionally, we have a set of topics, extracted from the collection of documents or predefined, $T = t_1, \ldots, t_l$ which will represent

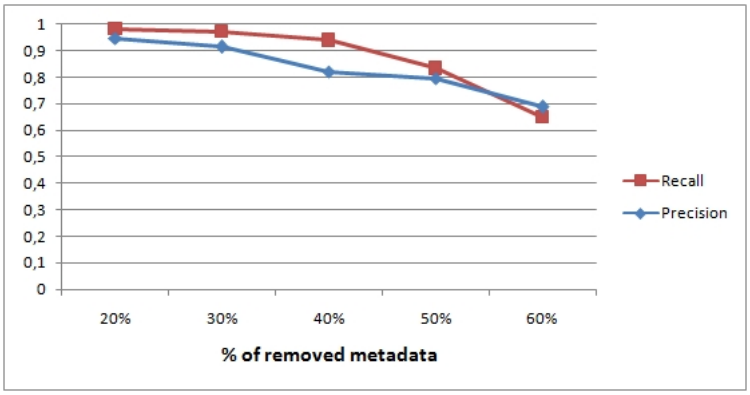


Figure 9: Metadata Enhancement Evaluation: Average values of Precision and Recall

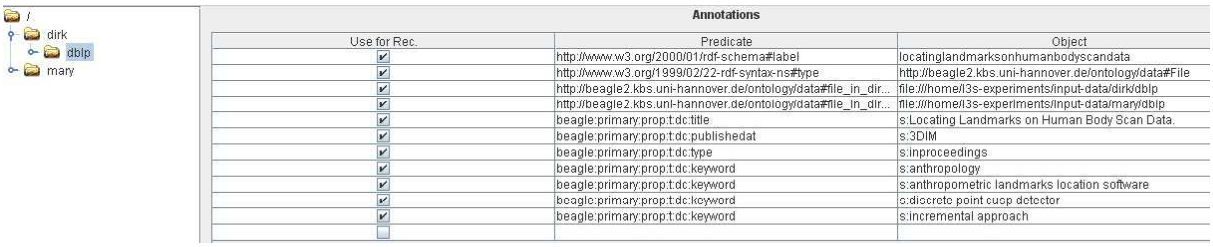


Figure 10: Screenshot of Metadata Enrichment Recommendation - File selection

the dimensions of the space, and a query q of the type "Find the experts on the topic X". The task is then to retrieve a list of candidates from C ranked by degree of expertise on q.

6.4.2 Formal Definition of the Basic Model

The model builds a multi-dimensional vector space S with inner product. We define on S a basis representing l expertise topics, $T = \vec{t_1}, \ldots, \vec{t_l}$.

We further represent a set of m documents $d_i \in D$, with $1 \leq i \leq m$ as linear combination of the basis vectors, as usually done in the standard IR vector space model. We then have $\vec{d_i} = d_{1,i}\vec{t_1} + \ldots + d_{l,i}\vec{t_l}$, where the $d_{k,i}$s are coefficients measuring how a topic vector $\vec{t_k}$ belongs to the document d_i. These coefficients can be based on TF$\times$IDF, if we represent the topic basis vectors $\vec{t_k}$ by terms.

Our model differs from the standard IR vector space model by also representing the candidates $c_j \in C$, $1 \leq j \leq n$ as vectors in this space. In order to do

Figure 11: Screenshot of Metadata Enrichment Recommendation - Recommendations provided

so, we define a function

$$\begin{aligned} f : D \times C &\to \mathbb{R} \\ (d_i, c_j) &\mapsto r_{i,j} \end{aligned} \tag{8}$$

which, for each candidate $c_j \in C$ assigns a weight $r_{i,j}$ to each document d_i. This weight is defined as representing how much candidate c_j is an expert on the subject addressed by document d_i. Another view is to see the document d_i an evidence of the expertise of candidate c_j. We can use information extraction techniques to estimate the probability that a candidate j's name appears in a document i, and use this as the weight $r_{i,j}$.

Given the set of documents-vectors D and the function f, we can find the coordinates of a candidate c_j in the vector space generated by the chosen topic basis T:

$$\vec{c_j} = \sum_{i=1}^{m} r_{i,j}\vec{d_i} = \sum_{i=1}^{m}\left(r_{i,j}\sum_{k=1}^{l} d_{k,i}\vec{t_k}\right) = \sum_{k=1}^{l}\left(\sum_{i=1}^{m} d_{k,i}r_{i,j}\right)\vec{t_k}$$

or in matrix form

$$\boxed{C = D \times R} \tag{9}$$

where

$$\begin{aligned} C &= [\vec{c_1}|\ldots|\vec{c_n}]\,,\ D = [\vec{d_1}|\ldots|\vec{d_m}] \\ R &= [r_{i,j}] \text{ with } 1 \le i \le m, 1 \le j \le n \end{aligned}$$

and n is the number of candidates, m is the number of documents, $\vec{d_i}$ is the vector of the ith document, $\vec{c_j}$ is the vector of the jth candidate, and $r_{i,j}$ is the relationship weight between the document i and the candidate j.

The query is represented as a vector containing the topics for which we need appropriate experts or documents, i.e. $\vec{q}$ as $\vec{q} = q_1\vec{t_1} + \ldots + q_n\vec{t_n}$.

We use a distance measure in order to rank candidates and documents based on increasing distance to the query vector. A default measure is the well known cosine similarity $sim(\vec{q}, \vec{v}) = \frac{\vec{q}\cdot\vec{v}}{\|\vec{q}\|\|\vec{v}\|}$ where "$\cdot$" denotes the dot-product of the two vectors and $\vec{v} \in \{d_i, c_j\}$. We will discuss an alternative similarity measure in section 6.4.4, and show that it performs better than cosine similarity for the expert search task.

Now, that we have represented both candidates and documents in the same vector space, we can retrieve both candidates and documents relevant to a query, an advantage over pure document or expert search systems. We can even query for documents most representative of the domain of expertise of a candidate, using a candidate vector as query. It is useful, though, to rank relevant candidates and documents in separate sets, to avoid having to compare similarity of documents and candidates with a given query.

6.4.3 Extensions of the Model

Different types of refinement can be applied in order to include more evidences of candidate expertise. We will discuss three types of extensions: document dependent ones, candidate and topic dependent ones, and candidate dependent extensions.

For the document dependent extensions, we add weights for each document and use a diagonal matrix to add these these in our model: $C = D \times (diag(\vec{e}) \times R)$ where $diag(\vec{e})$ is a $m \times m$ diagonal matrix with the values e_i. Each value e_i represents the weight assigned to the document d_i.

This weight can, for example, represent time aspects, assigning for each document a weight proportional to the creation date of the document, valuing newer documents higher than old ones.

The second kind of refinement takes candidate and topic dependent aspects into account, relating the weight to each candidate and to each (topic) dimension of the space. This can be used to model the fact that a person working the first day on a topic is likely to be less expert on this topic than another person having worked on it all her life.

To represent this kind of refinement we can compute $C' = C \circ W$ where W is a matrix of the same dimension of C and $\circ$ indicates the Hadamard product between matrices. Each element $w_{j,k}$ of W indicates the weight for the dimension k of the candidate c_j.

Last, we can refine the results using evidence weights depending only on the candidate. We can for example use a cost function, to prefer older (more experienced) experts, or to prefer experts with a lower salary [73].

This can be represented as a transformation $C'' = C' \times diag(\vec{cf})$ where $diag(\vec{cf})$ is a $l \times l$ diagonal matrix with values cf_k, each value representing the cost associated with the candidate c_j.

6.4.4 Projection similarity

Long documents are usually not more relevant to a query than small ones. For this reason, in document retrieval to avoid favouring long over small documents measures like cosine similarity are used which normalize for document length. In expert retrieval, though, we do prefer to retrieve a person having more expertise in the topic expressed by the query. We will therefore define a measure we call projection similarity, which models the amount of expertise in the query topic by an orthogonal projection of the candidate vector onto the query vector, defined as follows: $projSim(\vec{q}, \vec{c_j}) = cos\theta \, \|\vec{c_j}\| = \frac{\vec{q} \cdot \vec{c_j}}{\|\vec{q}\|}$ where "$\cdot$", again, denotes the dot-product of the two vectors. Projection similarity favours long vectors over shorter ones, as we multiply the length of the vector $\vec{c_j}$ with the cosine of the angle between $\vec{c_j}$ and $\vec{q}$. So the longer the candidate vector the higher the similarity score.

6.4.5 Vector Space Dimensions (T)

Different ways of representing topics of expertise are possible. The following paragraphs illustrate three different ways to build and select the reference topics used as basis of the vector space.

Terms(TFxIDF) The classical way of building a vector space basis given a set of documents is to extract terms present in the document collection after eliminating the stop words and applying a stemming algorithm. The coordinates of a document are then the TF$\times$IDF values related to the corresponding term basis. Using this approach each candidate is represented as a vector containing a value for each term in the document collection representing her expertise on the topic represented by that term. We will use this approach as baseline for our experiments.

Terms & LSA Another relevant technique is to use latent semantic analysis [31] or latent semantic indexing (LSI), which aims at solving the problem of synonymy and polysemy. LSI is able to manipulate and significantly compress the dimensions for document collections and can improve performance compared to TF$\times$IDF without dimension reduction by 30% in the case of document retrieval [35].

Lexical compounds As last alternative we investigated the use of lexical compounds for keyword extraction. As [6] has shown we can extract from a set of documents a list of

"themes", as key concepts of that set. For example "business process model" is thus recognized as a topic and can be used to represent candidate expertise on that topic. [21, 6] have presented very good results for this technique for query expansion tasks.

In our context we use this method to extract the key expertises of a candidate from the set of documents related to her, and use this key expertises to place the expert in the vector space as described earlier.

6.4.6 Evaluation

This section discusses our experiments investigating 1) the benefit of projection similarity for retrieving candidates introduced in section 6.4.4, and 2) the benefit of various information retrieval optimizations for the vector space model to improve expert search.

We will focus on document dependent extensions as appropriate relevance assessments. Data are still missing for candidate and topic dependent extensions.

The TRECent Collection

The Enterprise Search Track (TRECent) [25] started in TREC 2005 as a pilot track and in 2006 took place with a complete set of relevance assessments. In TRECent 2006, participants used the W3C test collection [68], consisting of a document collection with about 330,000 HTML pages— mailing list entries, people home pages, CVS logs, W3C web pages, wiki pages—the relevance assessments from the human evaluators, a list of topics, and a list of expert candidates which contains the name and the e-mail address of persons to be considered as potential experts for the topics. We used the 2006 topics composed of title, description, and narrative parts, representing a query as concatenation of these constituents.

Experimental Setup

We implemented an expert search system which incorporates the model defined above. As output, our system produces a run file in TRECent format that allows us to use the official `trec_eval` utility program to compute the common evaluation measures in order to compare the performance of our system with results of other TRECent participants.

To compare performance for the different variants we tested, we ran the system varying only one variable at a time, keeping the others fixed. As reference run we use the run with the following properties: 1) use terms for indexing 2) use pruning method described below, considering only terms containing letters 3) use projection similarity measure 4) use weights (1/1) representing the occurrence of the candidates in either author or text fields 5) do not consider the PageRank values of the documents. All mean average precision values presented below are followed by the p-value of the t-test between the variant run and the reference run, noted between parentheses. We consider a difference to be statistically significant when $p \leq 0.05$.

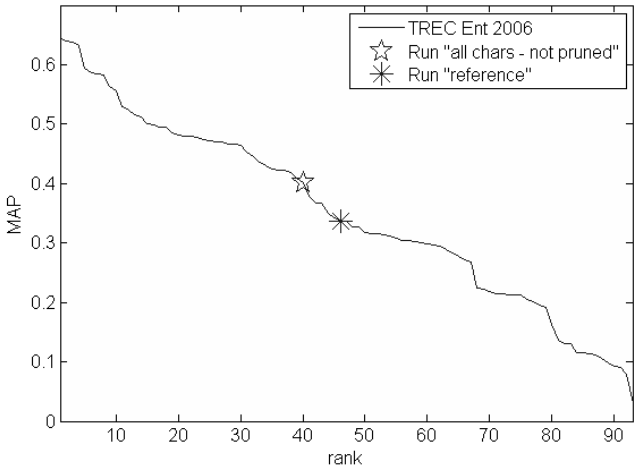
In figure 12(a) we show all runs submitted to TRECent 2006 ordered by MAP, as well as our system.

Vector space dimensions and pruning methods

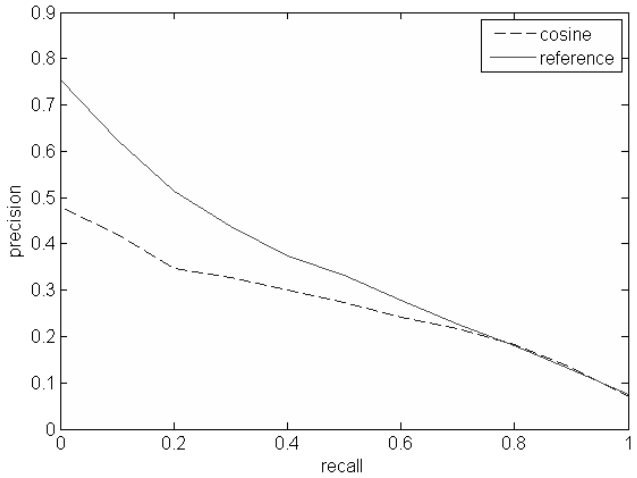
We compared three different indexing methods — term based, LSA based, and lexical compounds based — to investigate different possibilities of defining topics of expertise as described in section 6.4.5.

Furthermore, we tested different pruning schemes on the term basis. We used all the terms occurring in the document and then pruned the space dimensions considering only the first k bases ordered by document frequency where k is the rank where we reach 80% of the total document frequency in the document collection. We also tried to remove noise from the collection considering only terms consisting of letters.

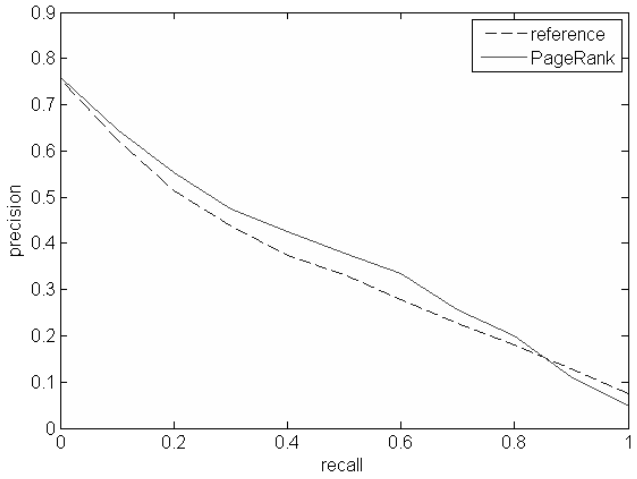
Performing the experiments on the term based indexing of the collection, we see that performance is maximized when we consider both digits and letters



(a)



(b)



(c)

Figure 12: (a) Retrieval effectiveness (MAP values) of the 2006 participant at TRECent (b) Precision/Recall curves varying the vector similarity function (c) Precision/Recall curves using PageRank weights for the documents

and when we do not prune the basis considering only the most frequent terms (see table 3). As expected from document retrieval, removing terms decreases retrieval effectiveness while increasing indexing efficiency. In the reference run we used the smallest basis to save time.

	Pruned	Not Pruned
Only Letters	0.3370	0.3854 (p =0.0091)
All Chars	0.3716 (p =0.0112)	0.4024 (p =0.0035)

Table 3: Retrieval effectiveness (MAP and p-value) varying the pruning techniques

Our experiments show that using lexical compounds as topics of expertise is not significantly better than using all terms in the documents (see table 4). We also see that applying LSA on the term space kills the effectiveness of our system. One reason for this is that we performed LSA on the already pruned version because of resource limitations. We also tried to prune the lexical compounds basis in the same way we pruned the term basis and we observed that, contrary to the term space, performance does not significantly change (p =0.8353) considering only the top k basis ordered by document frequency.

Dimension	Term	LSA	LexComp	LexComp Pruned
MAP (p-value)	0.3370	0.0894 (p =0.0)	0.3586 (p =0.5927)	0.3625 (p =0.5374)

Table 4: Retrieval effectiveness (MAP and p-value) using different vector space dimensions

Relationship weights

To compute the relationship matrix R (see section 6.4.2) we used a version of the collection which contains tags representing occurrences of candidate names and e-mails from [75].

To associate documents with candidates, we set the relationship weight to one when a candidate name, email or username as defined in [75] appears in a document. We intend to investigate further refinements of this scheme in the future. Table 5 shows how effectively we can identify the candidates in this tagged collection, where %cand is the percentage of candidates identified in the collection, %rel_cand is the percentage of identified candidates which are relevant, #avg is the average number of documents associated (that is, $r_{i,j} > 0$) with a candidate, and %docs is the percentage of document d_i with at least one relationship $r_{i,j} > 0$.

In these experiments we used different weights to represent the occurrence of a candidate's name or address in the author field. For these experiments we only consider the emails; this is sound as using only the mailing list part of the W3C collection on the 2006 queries performance was not significantly different. We observed that when not considering candidate occurrences in the text (i.e. text weight is zero) effectiveness decreases. Including text occurrences with 10% of the weight of an author occurence still yields performance lower than the reference run (which used 1/1 weights for author/text). Other combinations of author/text occurrence weights did not significantly change

collection	%cand	%rel_cand	#avg	%docs
2006	71.38%	97.89%	1246	40.89%

Table 5: Candidates extraction effectiveness

effectiveness compared to the reference run (see table 6). These results imply that candidate occurrences in the text are also very important and can not be ignored.

Author/Text weights	1/0	1/0.1	1/0.25	1/0.5	1/0.75	1/0.1
MAP	0.2246	0.3149	0.3306	0.3378	0.3365	0.3370
p-value	0.0	0.0183	0.1559	0.6803	0.5528	

Table 6: Retrieval effectiveness (MAP and p-value) using different text weights

Document dependent extensions

To experiment with and evaluate the document dependent extensions discussed in section 6.4.3 we used the PageRank values of the documents. These values are computed using the link structure of the HTML document collection [15]. We note that though PageRank has been shown not to work very well in the enterprise scenario the collection we use is based on a public web crawl so PageRank should be suited to identify authoritative documents.

We experimented with the use of a document dependent feature in order to refine the candidates position in the vector space. The MAP values show that using the PageRank values of the documents as weights for the candidate placement slightly improves performance, see table 7. The difference at high precision values is very small, though, see figure 12(c).

with PageRank weights	without PageRank weights
0.3435 (p =0.0515)	0.3370

Table 7: Retrieval effectiveness (MAP and p-value) using PageRank weights for the documents

Similarity functions

We investigated the performances of two different similarity measures comparing query and retrievable vectors: cosine similarity and projection similarity presented in section 6.4.4. Our hypothesis was that projection similarity can improve expert retrieval effectiveness.

While it is possible to use cosine similarity, this does not take information about the vector lengths into account. Using projection similarity retrieval effectiveness improves substantially: growing from a MAP of 0.2502 for cosine similarity to 0.3370 for projection similarity, with a statistically significant difference (p =0.0020) Figure 12(b) also shows, that especially for high precision values the improvement is substantial. These results confirm our intuition described in section 6.4.4.

6.5 Recommendation of Tags

Recommending tags can serve various purposes, such as: increasing the chances of getting a resource annotated, reminding a user what a resource is about and consolidating the vocabulary across the users. We describe here how the folksonomy structure of the Nepomuk network of peers can be used to find suitable tag recommendations for resources the user wants to annotate. We start by introducing social resource sharing systems, formalize the underlying datastructure called folksonomy, and formulate the tag recommendation problem. Then we introduce two algorithms to recommend tags to users and describe our evaluation methodology. Finally, we present results of the application of the proposed methods on three real-world datasets.[16]

[15] `http://apex.sjtu.edu.cn/apex_wiki/Shenghua_Bao/TREC_2006_PageRanks/` [16] The results presented here have been (partially) presented at the ECML/PKDD 2007 [47].

6.5.1 A Formal Model for Folksonomies.

A folksonomy describes the users, resources, and tags, and the user-based assignment of tags to resources. Formally, a folksonomy is a tuple $\mathbb{F} := (U, T, R, Y)$ where U, T, and R are finite sets, whose elements are called users, tags and resources, resp., and Y is a ternary relation between them, i. e., $Y \subseteq U \times T \times R$, whose elements are called tag assignments (tas for short).

In this deliverable, we will use an equivalent view on the folksonomy structure. We will consider it as 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 $r \in R$, $\text{tags}(()u, r) := \{t \in T \mid (u, t, r) \in Y\}$, i. e., $\text{tags}(()u, r)$ is the set of all tags that user u has assigned to resource r. The set of all posts of the folksonomy is then $P := \{(u, S, r) \mid u \in U, r \in R, S = \text{tags}(()u, r)\}$. Thus, each post consists of a user, a resource and all tags that this user has assigned to that resource.

Users are typically described by their user ID, and tags may be arbitrary strings. What is considered a resource depends on the type of system. In web based systems like del.icio.us, the resources are typically URLs, in BibSonomy URLs or publication references, and in last.fm, the resources are artists. On the social semantic desktop of Nepomuk resources annotated with tags are identified by their URI and are of different type. They might include wiki pages, PDF documents, emails, URLs, or local files.

6.5.2 Tag Recommender Systems

Recommender systems (RS) in general recommend interesting or personalized information objects to users based on explicit or implicit ratings. Usually RS predict ratings of objects or suggest a list of new objects that are relevant for the user most. In tag recommender systems the recommendations are, for a given user $u \in U$ and a given resource $r \in R$, a set $\tilde{T}(u, r) \subseteq T$ of tags. In many cases, $\tilde{T}(u, r)$ is computed by first generating a ranking on the set of tags according to some quality or relevance criterion, from which then the top n elements are selected.

6.5.3 Collaborative Filtering

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

Basic CF principle

The idea is to suggest new objects or to predict the utility of a certain object based on the opinion of like-minded users [64]. In CF, for m users and n objects, the user profiles are represented in a user-object matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$. The matrix can be decomposed into row vectors:

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

where $x_{u,o}$ indicates that user u rated object o by $x_{u,o} \in \mathbb{R}$. Each row vector $\vec{x}_u$ corresponds thus to a user profile representing the object ratings of a particular user. This decomposition leads to user-based CF. (The matrix can alternatively be represented by its column vectors leading to item-based recommendation algorithms.)

Now, one can compute, for a given user u, the recommendation as follows.

First, based on matrix $\mathbf{X}$ and for a given k, the set N_u^k of the k users that are most similar to user $u \in U$ are computed: $N_u^k := \arg\max_{v \in U}^k \text{sim}(\vec{x}_u, \vec{x}_v)$ where the superscript in the $\arg\max$ function indicates the number k of neighbors to be returned, and sim is regarded (in our setting) as the cosine similarity measure. Then, for a given $n \in \mathbb{N}$, the top n recommendations consist of a list of objects ranked by decreasing frequency of occurrence in the ratings of the neighbors (see Eq. 10 below for the folksonomy case).

This brief discussion refers only to the user-based CF case, moreover, we consider only the recommendation task since in collaborative tagging systems there are usually no ratings and therefore no prediction. For a detailed description about the item-based CF algorithm see [33].

CF for Tag Recommendations in Folksonomies

Because of the ternary relational nature of folksonomies, traditional CF cannot be applied directly, unless we reduce the ternary relation Y to a lower dimensional space. To this end we consider as matrix $\mathbf{X}$ alternatively the two 2-dimensional projections $\pi_{UR}Y \in \{0,1\}^{|U|\times|R|}$ with $(\pi_{UR}Y)_{u,r} := 1$ if there exists $t \in T$ s.t. $(u,t,r) \in Y$ and 0 else and $\pi_{UT}Y \in \{0,1\}^{|U|\times|T|}$ with $(\pi_{UT}Y)_{u,t} := 1$ if there exists $r \in R$ s.t. $(u,t,r) \in Y$ and 0 else. The projections preserve the user information, and lead to log-based like recommender systems based on occurrence or non-occurrence of resources or tags, resp., with the users. Notice that now we have two possible setups in which the k-neighborhood N_u^k of a user u can be formed, by considering either the resources or the tags as objects.

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

$$\tilde{T}(u,r) := \arg\max_{t \in T}^n \sum_{v \in N_u^k} \text{sim}(\vec{x}_u, \vec{x}_v)\delta(v,t,r) \tag{10}$$

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

6.5.4 A Graph Based approach

The seminal PageRank algorithm [16] reflects the idea that a web page is important if there are many pages linking to it, and if those pages are important themselves.[17] In [44], we employed the same underlying principle for Google-like search and ranking in folksonomies. The key idea of our FolkRank algorithm is that a resource which is tagged with important tags by important users becomes important itself. The same holds, symmetrically, for tags and users, thus we have a graph of vertices which are mutually reinforcing each other by spreading their weights. In this section we briefly recall the principles of the FolkRank algorithm, and explain how we use it for generating tag recommendations. More details can be found in [44].

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

Folksonomy-Adapted Pagerank

First we convert the folksonomy $\mathbb{F} = (U, T, R, Y)$ into an undirected tri-partite graph $G_{\mathbb{F}} = (V, E)$. The set V of nodes of the graph consists of the disjoint

[17] This idea was extended in a similar fashion to bipartite subgraphs of the web in HITS [49] and to n-ary directed graphs in [70].

union of the sets of tags, users and resources. All co-occurrences of tags and users, users and resources, tags and resources become edges between the respective nodes (more details in [44]).

The rank of the vertices of the graph are the entries in the fixed point $\vec{w}$ of the weight spreading computation

$$\vec{w} \leftarrow dA\vec{w} + (1-d)\vec{p} \ , \tag{11}$$

where $\vec{w}$ is a weight vector with one entry for each node, A is the row-stochastic version of the adjacency matrix of the graph $G_{\mathbb{F}}$ defined above, $\vec{p}$ is the preference vector, and $d \in [0,1]$ is determining the influence of $\vec{p}$.

For a global ranking, one will choose $\vec{p} = \mathbf{1}$, i. e., the vector composed by 1's. In order to generate recommendations, however, $\vec{p}$ can be tuned by giving a higher weight to the user and to the resource for which one currently wants to generate a recommendation. The recommendation $\tilde{T}(u,r)$ is then the set of the top n nodes in the ranking, restricted to tags. In the experiments presented in Section 6.5.5, we will see that this version performs reasonable, but not exceptional. This is in line with our observation in [44] which showed that the topic-specific rankings are biased by the global graph structure. As a consequence, we developed the following differential approach.

FolkRank—Topic-Specific Ranking

As the graph $G_{\mathbb{F}}$ that we created in the previous step is undirected, we face the problem that an application of the original PageRank would result in weights that flow in one direction of an edge and then 'swash back' along the same edge in the next iteration, so that one would basically rank the nodes in the folksonomy by their degree distribution. This makes it very difficult for other nodes than those with high edge degree to become highly ranked, no matter what the preference vector is.

This problem is solved by the differential approach in FolkRank, which computes a topic-specific ranking of the elements in a folksonomy. Let $\vec{w}_0$ be the fixed point from Equation (11) without preference vector and $\vec{w}_1$ be the fixed point with preference vector $\vec{p}$ and in this case $d = 0.7$. Then $\vec{w} := \vec{w}_1 - \vec{w}_0$ is the final weight vector. Thus, we compute the winners and losers of the mutual reinforcement of nodes when a user/resource pair is given, compared to the baseline without a preference vector. We call the resulting weight $\vec{w}[x]$ of an element x of the folksonomy the FolkRank of x.[18] For generating a tag recommendation for a given user/resource pair, we compute the ranking as described above, and then restrict the result set $\tilde{T}(u,r)$ to the top n tag nodes.

6.5.5 Evaluation

In this section we first describe the datasets we used, how we prepared the data, the methodology deployed to measure the performance, and which algorithms we used, together with their specific settings.

Datasets

To evaluate the proposed recommendation techniques we have chosen datasets from three different folksonomy systems: del.icio.us, Last.fm and BibSonomy. They have different sizes, different resources to annotate and are probably used by different people. Therefore they form a good basis to test our tag recommendation scenario in a general setting. Table 8 gives an overview on the datasets. For all datasets we disregarded if the tags had lower or upper case since this is the behaviour of most systems when querying them for posts tagged with a certain tag (although often they store the tags as entered by the user).

[18] In [44] we showed that $\vec{w}$ provides indeed valuable results on a large-scale real-world dataset while $\vec{w}_1$ provides an unstructured mix of topic-relevant elements with elements having high edge degree. In [45], we applied this approach for detecting trends over time in folksonomies.

Del.icio.us.: We used a dataset from del.icio.us[19] we obtained from July 27 to 30, 2005 [44]. Since del.icio.us allows its users to not tag resources at all (they can be accessed by the tag "system:unfiled") we added those posts with the tag "system:unfiled" to the dataset.

Last.fm.: The data was gathered during July 2006, partly through the web services API (collecting user nicknames), partly crawling the Last.fm[20] site. Here the resources are artist names, which are already normalized by the system.

BibSonomy: For BibSonomy,[21] we were able to create a complete snapshot of all users, resources (both publication references and bookmarks) and tags publicly available at April 30, 2007, 23:59:59 CEST. From the snapshot we excluded the posts from the DBLP computer science bibliography[22] since they are automatically inserted and all owned by one user and all tagged with the same tag (dblp). Therefore they do not provide meaningful information for the analysis.

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

Table 8: Characteristics of the used datasets.

Core computation

Many recommendation algorithms suffer from sparse data or the "long tail" of items which were used by only few users. Hence, to increase the chances of good results for all algorithms (with exception of the most popular tags recommender) we will restrict the evaluation to the "dense" part of the folksonomy, for which we adapt the notion of a p-core [13] to tri-partite hypergraphs. The p-core of level k has the property, that each user, tag and resource has/occurs in at least k posts.

To construct the p-core, recall that a folksonomy (U, T, R, Y) can be formalized equivalently as tri-partite hypergraph $G = (V, E)$ with $V = U \dot\cup T \dot\cup R$. First we define, for a subset $\tilde V$ of V (with $\tilde V = \tilde U \dot\cup \tilde T \dot\cup \tilde R$ and $\tilde U \subseteq U, \tilde T \subseteq T, \tilde R \subseteq R$), the function

$$P(v, \tilde V) = \begin{cases} \{(v, S, r) \mid r \in \tilde R, S = \text{tags}_{()}\tilde V(v, r)\} \\ \qquad\qquad \text{if} \quad v \in \tilde U \\ \{(u, v, r) \mid u \in \tilde U, r \in \tilde R\} \\ \qquad\qquad \text{if} \quad v \in \tilde T \\ \{(u, S, v) \mid u \in \tilde U, S = \text{tags}_{()}\tilde V(u, v)\} \\ \qquad\qquad \text{if} \quad v \in \tilde R \end{cases} \tag{12}$$

which assigns to each $v \in \tilde V$ the set of all posts in which v occurs. Here, $\text{tags}_{()}\tilde V$ is defined as in Section 6.5.1, but restricted to the subgraph $(\tilde V, E_{|\tilde V})$. Let $p(v, \tilde V) := |P(v, \tilde V)|$. The p-core at level $k \in \mathbb{N}$ is then the subgraph of (V, E) induced by $\tilde V$, where $\tilde V$ is a maximal subset of V such that, for all $v \in \tilde V$, $p(v, \tilde V) \geq k$ holds.

Since $p(v, \tilde V)$ is, for all v, a monotone function in $\tilde V$, the p-core at any level k is unique [13], and we can use the algorithm presented in [13] for its computation. An overview on the p-cores we used for our datasets is given in Table 9. For BibSonomy, we used $k = 5$ instead of 10 because of its smaller size. The largest k for which a p-core exists is listed, for each dataset, in the last column of Table 8.

[19] http://del.icio.us [20] http://www.last.fm [21] http://www.bibsonomy.org
[22] http://www.informatik.uni-trier.de/~ley/db/

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

Table 9: Characteristics of the p-cores at level k.

Evaluation methodology

To evaluate the recommenders we used a variant of the leave-one-out hold-out estimation [43] which we call LeavePostOut. In all datasets, we picked, for each user, one of his posts p randomly. The task of the different recommenders was then to predict the tags of this post, based on the folksonomy $\mathbb{F} \setminus \{p\}$.

As performance measures we use precision and recall which are standard in such scenarios [43]. With r being the resource from the randomly picked post of user u and $\tilde{T}(u,r)$ the set of recommended tags, recall and precision are defined as

$$\text{recall}(\tilde{T}(u,r)) = \frac{1}{|U|} \sum_{u \in U} \frac{|\text{tags}(()u,r) \cap \tilde{T}(u,r)|}{|\text{tags}(()u,r)|} \tag{13}$$

$$\text{precision}(\tilde{T}(u,r)) = \frac{1}{|U|} \sum_{u \in U} \frac{|\text{tags}(()u,r) \cap \tilde{T}(u,r)|}{|\tilde{T}(u,r)|}. \tag{14}$$

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

Most popular tags: For each tag we counted in how many posts it occurs and used the top tags (ranked by occurence count) as recommendations.

Most popular tags by resource: For a given resource we counted for all tags in how many posts they occur together with that resource. We then used the tags that occured most often together with that resource as recommendation.

Adapted PageRank: With the parameter $d = 0.7$ we stopped computation after 10 iterations or when the distance between two consecutive weight vectors was less than 10^{-6}. In $\vec{p}$, we gave higher weights to the user and the resource from the post which was chosen. While each user, tag and resource got a preference weight of 1, the user and resource from that particular post got a preference weight of $1 + |U|$ and $1 + |R|$, resp.

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

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

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

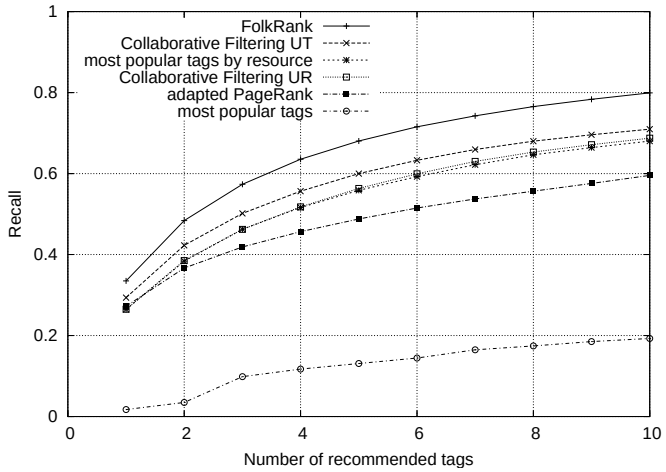


Figure 13: Recall for del.icio.us p-core at level 10

6.5.6 Results

In this section we present and describe the results of the evaluation. We will see that all three datasets show the same overall behavior: 'most popular tags' is outperformed by all other approaches; the CF-UT algorithm performs slightly better than and the CF-UR approach approx. as good as the 'most popular tag by resource', and FolkRank uniformly provides significantly better results.

We will now study the results in detail. There are two types of diagrams. The first type of diagram (Figure 13) shows in a straightforward manner how the recall depends on the number of recommended tags. In the other diagrams with usual precision-recall plots (Figures 14 and 15) a datapoint on a curve stands for the number of tags used for recommendation (starting with the highest ranked tag on the left of the curve and ending with ten tags on the right). Hence, the steady decay of all curves in those plots means that the more tags of the recommendation are regarded, the better the recall and the worse the precision will be.

Del.icio.us.

Figure 13 shows how the recall increases, when more tags of the recommendation are used. All algorithms perform significantly better than the baseline based on the most popular tags—whereas it is much harder to beat the resource specific most popular tags. The surprising result is that the graph based recommendations of FolkRank have superior recall—independent of the number of regarded tags. The second best results come from the collaborative filtering approach based on user tag similiarities. For ten recommended tags it reaches 89% of the recall of FolkRank (0.71 of 0.80)—a significant difference. The idea to suggest the top most popular tags of the resource gives a recall which is very similiar to using the CF recommender based on users resource similiarities—both perform worse than the aforementioned approaches. Between most popular tags by resource and most popular tags is the adapted PageRank which is influenced by the most popular tags, as discussed earlier.

The precision-recall plot in Figure 14 again reveals clearly the quality of the recommendations given by FolkRank compared to the other approaches. The top 10 tags given by FolkRank contained in average 80 % of the tags the users decided to attach to the selected resource. Nevertheless, the precision is rather poor with values below 0.2. So why do we call this a good result anyhow?

A post in del.icio.us contains only 2.45 tags in average. A precision of 100 % can therefore not be reached when recommending ten tags. However, from a subjective point of view, the additional 'wrong' tags may even be consid-

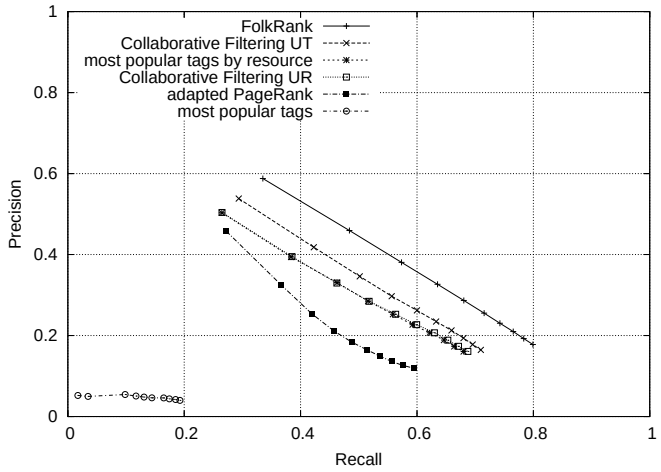


Figure 14: Recall and Precision for del.icio.us p-core at level 10

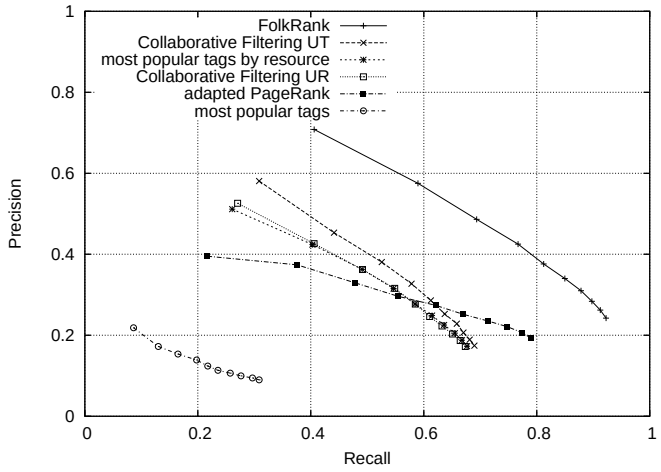


Figure 15: Recall and Precision for Last.fm p-core at level 10

ered as highly relevant, as the following example shows, where the user tnash has tagged the page `http://www.ariadne.ac.uk/issue43/chudnov/` with the tags semantic, web, and webdesign. Since that page discusses the interaction of publication reference management systems in the web by the OpenURL standard, the tags recommended by FolkRank (openurl, web, webdesign, libraries, search, semantic, metadata, social-software, sfx, seo) are adequate and capture not only the user's point of view that this is a webdesign related issue in the semantic web, but also provide him with more specific tags like libraries or metadata which the users nevertheless did not use. The CF based on user/tag similiarities recommends very similiar tags (openurl, libraries, social-software, sfx, metadata, me/toread, software, myndsi, work, 2read). The additional tags may thus animate users to use more tags and/or tags from a different viewpoint for describing resources, and thus lead to converging vocabularies.

The essential point in this example is, however, that FolkRank is able to predict—additionally to globally relevant tags—the exact tags of the user which CF could not. This is due to the fact that FolkRank considers, via the hypergraph structure, also the vocabulary of the user himself, which CF by definition doesn't do.

Last.fm. For this dataset, recall and precision for FolkRank are considerably higher than for the del.icio.us dataset, see Table 15. Even when just two tags are recommended, the recall is close to 60 %. Though the precision of the user-resource

Number of recommended tags	1	2	3	4	5	6	7	8	9	10
FolkRank	0.724	0.586	0.474	0.412	0.364	0.319	0.289	0.263	0.243	0.225
Collaborative Filtering UT	0.569	0.483	0.411	0.343	0.311	0.276	0.265	0.257	0.243	0.235
most popular tags by resource	0.534	0.440	0.382	0.350	0.311	0.288	0.267	0.250	0.241	0.234
Collaborative Filtering UR	0.509	0.478	0.408	0.341	0.311	0.285	0.267	0.252	0.241	0.234

Table 10: Precision for BibSonomy p-core at level 5

collaborative filtering approach is always slightly better than on the del.icio.us dataset, the recall is only better until the 7th tag where it falls below the recall reached on the del.icio.us dataset. Again, the graph based approach outperforms all other methods (CF-UT reaches at most 76 % of the recall of FolkRank). An interesting observation can be made about the adapted PageRank: its recall now is the second best after FolkRank for larger numbers of recommended tags. This shows the overall importance of general terms in this dataset—which have a high influence on the adapted PageRank (cf. Section 6.5.4).

BibSonomy. For the BibSonomy dataset the precision for FolkRank is similiar to the Last.fm dataset (see Table 10), but the recall (omitted here because of space restrictions) reaches only values comparable to the del.icio.us dataset. We will focus here on a phenomenon which is unique for that dataset. With an increasing number of suggested tags, the precision decrease is steeper for FolkRank than for the collaborative filtering and the 'most popular tags by resource' algorithm such that the latter two approaches for ten suggested tags finally overtake FolkRank. The reason is that the average number of tags in a post is around 4 for this dataset and while FolkRank can always recommend the maximum number of tags, for the other approaches there are often not enough tags available for recommendation. This is because in the p-core of order 5, for each post, often tags from only four other posts can be used for recommendation with these approaches. Consequently this behaviour is even more noticeable in the p-core of order 3 (which is not shown here).

7 Conclusion

In this deliverable we have provided the main infrastructure for sharing and recommending metadata, as part of WP5000's main contributions to the Social Semantic Desktop Nepomuk. We have investigated, implemented and analyzed methods for sharing metadata in large as well as recommending it.

The Metadata Sharing component, provides the central infrastructure for sharing structured data over a scalable P2P network. This component fosters semantic interoperability through iterative query reformulations, enabled by pairwise schema mappings which are imported in the system as part of the structured data shared by users. Metadata in RDF format can be efficiently searched through this component. However only simple atomic queries are supported so far. Some future work can comprise:

- Investigating approaches for efficient conjunctive queries;
- Analyzing different indexing methods which enable efficient similarity search.

With the Metadata Recommendation and the Expert Recommender components, users of the Social Semantic Desktop now are supported in several ways by the spread knowledge of their surrounding community. Several types of recommendations are generated from the available metadata of the P2P network of Nepomuk peers, and from the Desktop content, which can support the user in different situations.

The Metadata Recommendation component provide the user with recommendation of resources based on the shared metadata in the community. It also provide recommendations of additional metadata for the user's resources.

Future work might include:

- investigation on how to improve the effectiveness of the recommendation provided;
- extension of the approach using the metadata alignment services from WP5000 and WP2000;
- usage of the semantic search in the community provided by GridVine.

The Expert Recommender component uses a formal model for expert search adapting and extending the classical vector space model for documents. The proposed vector space model for expert search is flexible and keeps all benefits of the traditional model, allowing the expert search system designer to build upon a variety of techniques developed for document retrieval to improve expert search. Furthermore, we can query for both documents and expert with the same query.

Future work might include:

- investigation on how to combine different expert evidences and which of these combinations improve expert search effectiveness;
- exploration and experimentation of different possibilities to extract document-candidate relationship weights to improve retrieval effectiveness;
- evaluation of the effectiveness of retrieving both persons and documents together.

The Recommendation of Files and Metadata for enrichment uses a complex overlap computations, based on some metadata statistics and on string comparison measures. In this deliverable we present a fully operative prototype implementing the overlap computations integrated in the Nepomuk architecture, and a first evaluation of its functionalities. The prototype recommends

Files and additional Metadata for enrichment of existing metadata records based on the specification of a local file present in the users desktop.

Future work might include:

- the investigation of more advanced recommendation methods, by combining different techniques or evidences for the recommendation process;
- development of new approaches for ranking the obtained results by considering more closely the users information available.

By including tag recommendations in situations where a user wants to tag a resource — like a wiki page, an e-mail, or a file — we push the usage of a common vocabulary of keywords among peers and speed up the tagging process. Instead of entering each tag by hand, the user just needs to choose some tags from the suggestions computed by the recommender component. Furthermore, a simplified tagging process helps to increase the chance of getting a resource annotated. Similarly, tag recommendations can help the user to refine or extend a search when navigating through wiki pages or emails.

Those examples show the benefits of an integration of the Metadata Recommendation component in the end-user applications of the Social Semantic Desktop.

Future work might include:

- Investigating the behavior of the user and how it changes when recommendations are used.
- Dealing with sparse data or huge networks of peers.

In the third deliverable of WP5000 the focus will be put on the integration and evaluation of our components. Effort will be also put in the usage of a common evaluation dataset, in testing the efficiency of the components, and using advanced techniques in order to improve the effectiveness.

References

[1] K. Aberer. Efficient Search in Unbalanced, Randomized Peer-To-Peer Search Trees. Technical Report IC/2002/79, Swiss Federal Institute of Technology, Lausanne (EPFL), 2002. http://www.p-grid.org/Papers/TR-IC-2002-79.pdf.

[2] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In International Semantic Web Conference (ISWC), 2004.

[3] K. Aberer, A. Datta, M. Hauswirth, and R. Schmidt. Indexing data-oriented overlay networks. In International Conference on Very Large Databases (VLDB), 2005.

[4] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web. Journal of Artificial Intelligence Research, 25, 2006.

[5] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002), 2002.

[6] P. Anick and S. Tipirneni. The paraphrase search assistant: terminological feedback for iterative information seeking. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 153–159, 1999.

[7] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. Miller, and J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. SIGMOD Record, Special Issue on Peer-to-Peer Data Management, 32(3), 2003.

[8] L. Azzopardi, K. Balog, and M. de Rijke. Language modeling approaches for enterprise tasks. The Fourteenth Text REtrieval Conference (TREC 2005).

[9] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 43–50, 2006.

[10] K. Balog and M. de Rijke. Finding experts and their Details in e-mail corpora. Proceedings of the 15th international conference on World Wide Web, pages 1035–1036, 2006.

[11] K. Balog and M. de Rijke. Searching for people in the personal work space. International Workshop on Intelligent Information Access (IIIA-2006), 2006.

[12] K. Balog and M. de Rijke. Determining Expert Profiles (With an Application to Expert Finding). Proceedings of IJCAI-2007, pages 2657–2662, 2007.

[13] V. Batagelj and M. Zaversnik. Generalized cores, 2002. cs.DS/0202039, http://arxiv.org/abs/cs/0202039.

[14] D. Benz, K. Tso, and L. Schmidt-Thieme. Automatic bookmark classification: A collaborative approach. In Proceedings of the Second Workshop on Innovations in Web Infrastructure (IWI 2006), Edinburgh, Scotland, 2006.

[15] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data Management for Peer-to-Peer Computing : A Vision. In International Workshop on the Web and Databases (WebDB), 2002.

[16] 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.

[17] C. Campbell, P. Maglio, A. Cozzi, and B. Dom. Expertise identification using email communications. Proceedings of the 12th ACM Conference on Information and Knowledge Management (CIKM'03), pages 528–531, 2003.

[18] V. Carvalho and W. Cohen. Learning to Extract Signature and Reply Lines from Email. Proceedings of the Conference on Email and Anti-Spam, 2004.

[19] C. Cattuto, V. Loreto, and L. Pietronero. Collaborative tagging and semiotic dynamics, May 2006. http://arxiv.org/abs/cs/0605015.

[20] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 313–324, 2003.

[21] P. Chirita, C. Firan, and W. Nejdl. Summarizing Local Context to Personalize Global Web Search. Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM'06), pages 287–296, 2006.

[22] W. Cohen, P. Ravikumar, and S. Fienberg. A comparison of string distance metrics for name-matching tasks. Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03), 2003.

[23] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In IIWeb, pages 73–78, 2003.

[24] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In Proceedings of IJCAI-03 Workshop on Information Integration, pages 73–78, August 2003.

[25] N. Craswell and D. Hawking. Overview of the TREC-2004 Web Track. The Thirteenth Text REtrieval Conference (TREC 2004).

[26] N. Craswell, D. Hawking, A. Vercoustre, and P. Wilkins. P@noptic Expert: Searching for Experts not just for Documents. Ausweb, 2001.

[27] P. Cudré-Mauroux and K. Aberer. A Necessary Condition For Semantic Interoperability in the Large. In Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE), 2004.

[28] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic Message Passing in Peer Data Management Systems. In International Conference on Data Engineering (ICDE), 2006.

[29] V. Darlagiannis, R. Schmidt, N. Bonvin, and V. Agneeswaran. Nepomuk Deliverable D4.2 - advanced search and basic distributed storage, 2007.

[30] V. Darlagiannis, R. Schmidt, R. John, and E. Ioannou. Nepomuk Deliverable D4.1 - distributed search system - basic infrastructure, 2006.

[31] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6):391–407, 1990.

[32] G. Demartini, P. Haghani, R. Jäschke, A. Johnston, M. Kiesel, and R. Paiu. Nepomuk Deliverable D5.1 - Community Support Software First Version. http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D5-1, 2006.

[33] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst., 22(1):143–177, 2004.

[34] 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, Edinburgh, Scotland, 2006.

[35] S. Dumais. Improving the retrieval of information from external sources. Behavior Research Methods, Instruments and Computers, 23(2):229–236, 1991.

[36] H. Fang and C. Zhai. Probabilistic Models for Expert Finding. Proceedings of 29th European Conference on Information Retrieval (ECIR'07), pages 418–430, 2007.

[37] S. Ghita, W. Nejdl, and R. Paiu. Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context. Proceedings of ISWC, Galway, Ireland, November, 2005.

[38] O. Grebner, E. Ong, U. Riss, R. Gudjonsdottir, and H. Edlund. Nepomuk Deliverable D10.1 - SAP Scenario Report, 2006.

[39] S. Guha, N. Koudas, A. Marathe, and D. Srivastava. Merging the results of approximate match operations. Proceedings of the 30th International Conference on Very Large Databases (VLDB 2004), pages 636–647, 2004.

[40] A. Halevy, Z. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In International World Wide Web Conference (WWW), 2003.

[41] 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.

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

[43] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst., 22(1):5–53, 2004.

[44] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, The Semantic Web: Research and Applications, volume 4011 of Lecture Notes in Computer Science, pages 411–426, Heidelberg, June 2006. Springer.

[45] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Trend detection in folksonomies. In Y. S. Avrithis, Y. Kompatsiaris, S. Staab, and N. E. O'Connor, editors, Proc. First International Conference on Semantics And Digital Media Technology (SAMT), volume 4306 of LNCS, pages 56–70, Heidelberg, Dec 2006. Springer.

[46] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 538–543, 2002.

[47] R. Jäschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron, editors, Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, volume 4702 of Lecture Notes in Computer Science, pages 506–514, Berlin, Heidelberg, 2007. Springer.

[48] D. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. ACM Transactions on Database Systems (TODS), 31(2):716–767, 2006.

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

[50] G. Kokkinidis and V. Christophides. Semantic Query Routing and Processing in P2P Database Systems: The ICS-FORTH SQPeer Middleware. In EDBT Workshops, 2004.

[51] M. Kotelnikov, A. Polonsky, M. Kiesel, M. Völkel, H. Haller, M. Sogrin, P. Lannerö, and B. Davis. Nepomuk Deliverable D1.1 - Interactive Semantic Wikis. `http://nepomuk.semanticdesktop.org/xwiki/bin/Main1/D1-1`, 2006.

[52] S. Lauriere, A. Solleiro, S. Trug, C. Bogdan, K. Groth, and P. Lannero. Nepomuk Deliverable D11.1 - mandriva community case study - scenario report, 2007.

[53] J. Li, H. Boley, V. C. Bhavsar, and J. Mei. Expert finding for eCollaboration using FOAF with RuleML rules. Montreal Conference on eTechnologies (MCTECH), 2006.

[54] E. Liarou, S. Idreos, and M. Koubarakis. Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. In International Semantic Web Conference (ISWC), 2006.

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

[56] C. Macdonald and I. Ounis. Voting for Candidates: Adapting Data Fusion Techniques for an Expert Search Task. Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM'06), pages 387–396, 2006.

[57] C. Macdonald and I. Ounis. Using Relevance Feedback in Expert Search. Proceedings of 29th European Conference on Information Retrieval (ECIR'07), pages 431–443, 2007.

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

[59] A. McLean, A. Vercoustre, and M. Wu. Enterprise PeopleFinder: Combining Evidence from Web Pages and Corporate Data. Proceedings of Australian Document Computing Symposium, 2003.

[60] A. McLean, M. Wu, and A. Vercoustre. Combining Structured Corporate Data and Document Content to Improve Expertise Finding.

[61] W. Meng, C. Yu, and K. Liu. Building efficient and effective metasearch engines. ACM Computing Surveys (CSUR), 34(1):48–89, 2002.

[62] 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.

[63] G. Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In WWW '06: Proceedings of the 15th international conference on World Wide Web, pages 953–954, New York, NY, USA, 2006. ACM Press.

[64] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In World Wide Web, pages 285–295, 2001.

[65] 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.

[66] A. Seaborne. RDQL - A Query Language for RDF". W3C Member Submission, 2004. `http://www.w3.org/Submission/RDQL/`.

[67] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3), 1990.

[68] W3C Text Collection, 2005. http://research.microsoft.com/users/nickcr/w3c-summary (Last visit: September 2007).

[69] G. Wiederhold. Mediators in the Architecture of Future Information Systems. IEEE Computer, 25(3), 1992.

[70] 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.

[71] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In Proceedings of the Collaborative Web Tagging Workshop at the WWW 2006, Edinburgh, Scotland, 2006.

[72] J. Zhang, M. Ackerman, and L. Adamic. Expertise Networks in Online Communities: Structure and Algorithms. Proceedings of the 16th international conference on World Wide Web, pages 221–230, 2007.

[73] Y. Zhang. Personal communication, 2006.

[74] A. V. Zhdanova, L. J. B. Nixon, M. Mochol, and J. G. Breslin, editors. Proceedings of the 2nd International ISWC+ASWC Workshop on Finding Experts on the Web with Semantics, Busan, Korea, November 12, 2007, volume 290 of CEUR Workshop Proceedings. CEUR-WS.org, 2007.

[75] J. Zhu. W3C corpus annotated with W3C people identity. `http://ir.nist.gov/w3c/contrib/W3Ctagged.html`, September 2006.

[76] J. Zhu, A. Gonçalves, V. Uren, E. Motta, and R. Pacheco. Mining Web Data for Competency Management. Web Intelligence '05, pages 94–100, 2005.

A Abbreviations

API	Application Programming Interface
GAV	Global As View
NOA	NEPOMUK Annotation Ontology
NRL	NEPOMUK Representation Language
OSGi	Open Services Gateway Initiative
OWL	Web Ontology Language
P2P	Peer-to-Peer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VSM	Vector Space Model
WSDL	Web Service Description Language
XML	Extensible Markup Language

B ST5210 — Specification of Exchange Formats & Workflows for Metadata Sharing and Recommendations

Table of Contents

Overview

Current social networking research is mostly analyzing the interactions between users and the communities they generate. Although this is a necessary step in building social software, it is just its beginning. In order to interact and find relevant material in a community, there must be a concrete mechanism for metadata exchange between users, possibly subject to several access models. More, these shared metadata should also incorporate the PC desktop context under each it was created (e.g. a file was saved from an email attachment).

ST5210 will specify a syntactical framework for metadata sharing and user profile definition. It will investigate the resources, data structures, and especially user workflows (series of actions users make in order to determinate the context of their resource exchange) suitable for metadata based description. Once these items have been identified, sharing formats will be defined, i.e., which metadata should be included when exchanging information. To summarize, the current document will be a specification of:

1. Resources and workflows suitable to be semantically annotated
2. Metadata exchange formats for socially exchanging the above mentioned items
3. User profile definition for metadata based recommender tools

Specifications

Peers

Definition In WP5000, a "peer" is a generic term associated to a person. A "peer" in this sense is characterized by a unique URI, a name, has a certain number of resources and a profile. In the context of WP5000, a peer:

1. Can exchange resources and metadata with other peers

2. May have assigned trust values to the other peers with whom he exchanges information
3. May use the trust values to bias the ranking computation such that resources coming from highly trusted peers are also highly ranked
4. Can be member of one or more communities and share his resources with the other members

Communities of Peers

Definition In the context of WP5000 communities represent groups of peers, sharing some common characteristics (e.g., same topic of interest). Communities can be manually defined (e.g., FOAF, in which people manually specify their friends) – based on static information, or can be automatically inferred based on the information flow identified among peers.

Architecture The communities of peers we specify/identify are not isolated and independent from each other (Fig. 1). Same peers may be members of several communities and thus creating virtual links among communities. Additionally peers communicate not only with the other peers, members of the same community, but also with members from neighboring communities (Fig. 2).

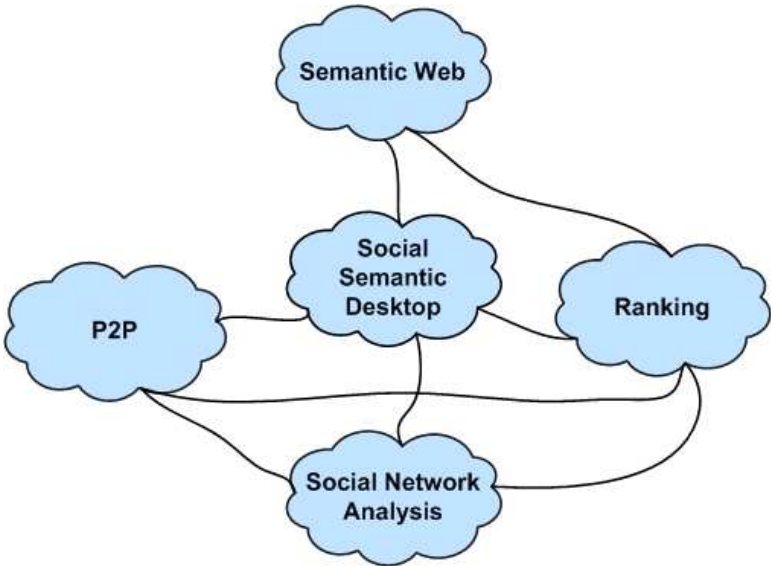


Fig. 1. Interaction among communities of peers

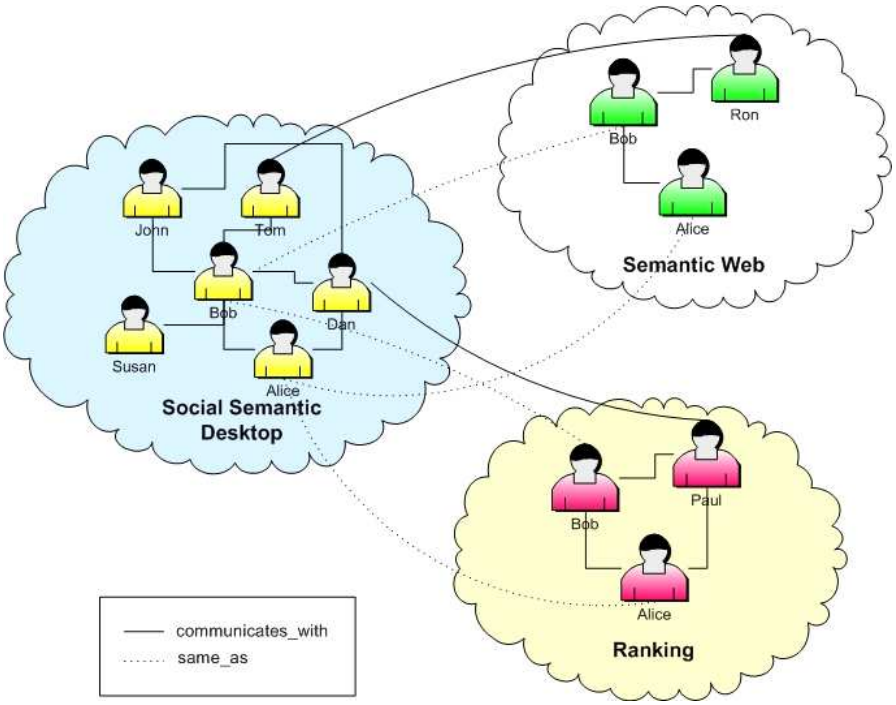


Fig. 2. Interactions among communities – detailed view

Profiles of Peers and Communities

Profiles of peers will be stored in FOAF/RDF format [FOAF, FOAF specification]. There are five broad categories in which the FOAF terms are grouped. However, for our purposes, of special importance are the foaf:interest and foaf:knows properties, as they allow to describe on one hand the user's competencies and interests, and on the other hand his social network. For community detection to work, it is necessary that each peer makes at least his foaf:interest information public. Basic information, which will be present in every peer profile, refers to:

1. peer's URI
2. foaf:interest – describing the peer's research interest
3. foaf:mbox – representing the peer's personal mailbox
4. foaf:homepage – describing the peer's home page

Additional and optional information specified in the peer profile will include:

1. foaf:surname – the surname of the peer
2. foaf:givenname – the first name of the peer
3. foaf:nick – nick name of the peer
4. foaf:title – Mr., Mrs., Ms., Dr., etc
5. foaf:img – peer's photo
6. foaf:knows – a person known by the peer
7. foaf:currentProject – a project this peer is currently working on
8. foaf:workplaceHomepage – the homepage of the organization the peer works for

9. foaf:publications – link to the publications of this peer

Profiles are defined manually by the user and may be extended automatically. We have developed in T4100 a technique for extracting the profile of the ontology of a peer, i.e. a compact representation of his knowledge [1]. For knowledge stored in a folksonomy system ("community peer"), we are currently working on extraction methods. The results will be presented in deliverables D5.1-3.

These profiles either manually specified or (semi)automatically inferred from the peers' interactions, will be stored locally at each peer. There are some reasons for storing them on the peers themselves: up-to-date information, peers can decide who sees what, etc. However, the basic information specified in the profile has to be available to everybody. Peers can only decide which of the additional/optional properties are publicly available.

The profiles will be exchanged between peers via the P2P network with the mechanisms provided for exchanging RDF data, and/or stored on a community peer, where they are the basis for the community detection and analysis.

Here is an example of a specification of a peer profile:

```
<rdf:RDF
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <foaf:Project rdf:about="http://nepomuk.semanticdesktop.org/">
    <foaf:homepage rdf:resource="http://nepomuk.semanticdesktop.org/"/>
  </foaf:Project>
  <foaf:Person rdf:about="http://www.kde.cs.uni-kassel.de/jaeschke">
    <foaf:workInfoHomepage
      rdf:resource="http://www.kde.cs.uni-kassel.de/jaeschke"/>
    <foaf:img>
      <foaf:Image rdf:about="http://www.kde.cs.uni-kassel.de/jaeschke/me.jpg"/>
    </foaf:img>
    <foaf:phone rdf:resource="tel:+495618046253"/>
    <foaf:knows>
      <foaf:Person rdf:about="http://www.bundestag.de/mdb/bio/M/miersja0.html">
        <foaf:homepage
          rdf:resource="http://www.bundestag.de/mdb/bio/M/miersja0.html"/>
        <foaf:name>Jakob Mierscheid</foaf:name>
        <foaf:mbox rdf:resource="mailto:wollner@spdfrak.de"/>
        <rdfs:seeAlso
          rdf:resource="http://www.bundestag.de/mdb/bio/M/miersja0.html"/>
      </foaf:Person>
    </foaf:knows>
    <foaf:title>Mr</foaf:title>
    <foaf:interest>
      <foaf:Document rdf:about="http://www.semanticweb.org/">
        <foaf:topic>Ontology</foaf:topic>
        <foaf:topic>Semantic Web</foaf:topic>
        <foaf:topic>Internet</foaf:topic>
      </foaf:Document>
    </foaf:interest>
    <foaf:name>Robert Jaeschke</foaf:name>
    <foaf:nick>Oberon</foaf:nick>
    <foaf:publications>
      <foaf:Document
rdf:about="http://www.bibsonomy.org/publ/search/J%C3%A4schke+user%3Ajaeschke"/>
    </foaf:publications>
    <foaf:givenname>Robert</foaf:givenname>
    <foaf:surname>Jaeschke</foaf:surname>
    <foaf:workplaceHomepage rdf:resource="http://www.kde.cs.uni-kassel.de/"/>
    <foaf:currentProject rdf:resource="http://nepomuk.semanticdesktop.org/"/>
```

```
    <foaf:mbox rdf:resource="mailto:jaeschke@kde.cs.uni-kassel.de"/>
  </foaf:Person>
</rdf:RDF>
```

Specification of the profiles of the communities relies mainly on two categories: Projects and Groups, which allow us to talk about groups and group membership among others. Groups are represented with the aid of the **foaf:Group** class, which represents a collection of individual agents. The **foaf:member** property allows us to explicitly express the membership of agents to a group. Since the **foaf:Person** class is a subclass of the **foaf:Agent** class, persons can also be members of a group. The profile of a community has to be publicly available.

Here is an example of a definition of a community of peers:

```
<rdf:RDF
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <foaf:Document rdf:about="http://www.semanticweb.org/">
    <foaf:topic>Internet</foaf:topic>
    <foaf:topic>Ontology</foaf:topic>
    <foaf:topic>Semantic Web</foaf:topic>
  </foaf:Document>
  <foaf:Group rdf:about="http://nepomuk.semanticdesktop.org/foaf/com/Nepomuk">
    <foaf:name>Nepomuk</foaf:name>
    <foaf:interest rdf:resource="http://www.semanticweb.org/"/>
    <foaf:member rdf:resource="http://lsirpeople.epfl.ch/hauswirth/"/>
    <foaf:member
rdf:resource="http://www.aifb.uni-karlsruhe.de/Personen/viewPerson?id_db=2076"/>
    <foaf:member rdf:resource="http://www.l3s.de/~paiu/"/>
    <foaf:member rdf:resource="http://www.kde.cs.uni-kassel.de/jaeschke"/>
  </foaf:Group>
</rdf:RDF>
```

Resources

Definition We define as a "resource" any information item that can be found on the users' desktop. This basically means that both desktop objects and their associated contextual information generated by the corresponding extractors and adapters are denoted as "resources".

This following list represents the desktop objects types for which Beagle++ has filters/extractors:

1. Office Documents
   (a) OpenOffice.org (http://www.openoffice.org/) (sxw, sxc, sxi and more)
   (b) OpenDocument (odt, ods, odp)
   (c) Microsoft Office (doc, xls, ppt)
   (d) AbiWord (http://www.abisource.com) (abw)
   (e) Rich Text Format (rtf)
   (f) PDF
2. Text Documents
   (a) HTML (xhtml, html, htm)
   (b) Source code (C, C++, C#, Fortran, Java, JavaScript, Lisp, Matlab, Pascal, Perl, PHP, Python, Ruby, Scilab and Shell scripts)

   (c) Plain text (txt, any plain text file that isn't filed under any other category)

3. Documentation/Help Documents
   (a) Texinfo
   (b) Man pages
   (c) Docbook
   (d) Monodoc
   (e) Windows help files (chm)
4. Images (jpeg, png, bmp, tiff, gif)
   (a) F-Spot (`http://f-spot.org/Main_Page`) and Digikam (`http://www.digikam.org/`) tags in the images are also indexed
5. Audio (mp3, ogg, flac)
6. Video (mpeg, asf, wmv, mng, mp4, quicktime and other formats supported by mplayer)
7. Application launchers
8. Linux packages (ebuild, rpm)
9. Generic XSLT files
10. RDF/RDFS, XML
11. Bibtex files

Desktop documents having one of the above mentioned MIME types are indexable and searchable within Beagle++. The list is easily extendable, since for adding new MIME types to be indexed and searched by Beagle++ one needs to create a corresponding extractor/adapter and register it to the currently available list of filters.

Metadata Creation and Storage As described above, each workspace has a number of extractors/adapters that receive desktop objects and generate contextual information (in RDF/XML format) describing these objects. The generated contextual information is stored in an RDF Store.

Example

Suppose you create a new file in your workspace named "books.bib", and you place the following content:

File: books.bib

```
@book{DBLP:books/aw/AbiteboulHV95,
  author = {Serge Abiteboul and Richard Hull and Victor Vianu},
  title  = {Foundations of Databases}
}
```

The extractor or adapter responsible for parsing "bib" files will be alerted and after processing the file, it will generate the appropriate contextual information (RDF/XML) describing the file. The corresponding RDF/XML created by the Beagle++ extractor for the "books.bib" file looks as follows:

Contextual information describing file "books.bib"

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:j.0="http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:j.1="http://www.kbs.uni-hannover.de/beagle++/ontology/domain_l3s#">
<rdf:Description rdf:about="file:/home/ioannou/books.bib#AbiteboulHV95">
  <rdf:type rdf:resource="http://www.kbs.uni-hannover.de/beagle++/ontology/domain_l3s#Publication"/>
  <j.0:desktop_document_author rdf:resource="file:/home/ioannou/books.bib#Abiteboul_"/>
  <j.0:desktop_document_author rdf:resource="file:/home/ioannou/books.bib#Hull_"/>
  <j.0:desktop_document_author rdf:resource="file:/home/ioannou/books.bib#Vianu_"/>
  <j.0:file_stored_from>books.bib</j.0:file_stored_from>
  <j.0:text_title>Foundations of Databases</j.0:text_title>
</rdf:Description>
<rdf:Description rdf:about="file:/home/ioannou/books.bib#Abiteboul_">
  <rdf:type rdf:resource="http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person"/>
  <j.0:person_name>Serge Abiteboul</j.0:person_name>
</rdf:Description>
<rdf:Description rdf:about="file:/home/ioannou/books.bib#Vianu_">
  <rdf:type rdf:resource="http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person"/>
  <j.0:person_name>Victor Vianu</j.0:person_name>
</rdf:Description>
<rdf:Description rdf:about="file:/home/ioannou/books.bib#Hull_">
  <rdf:type rdf:resource="http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person"/>
  <j.0:person_name>Richard Hull</j.0:person_name>
</rdf:Description>
</rdf:RDF>
```

Finally, the generated information will be added to the RDF Store. Its important to note that each resource is uniquely identified using a unique URI. In our example this URI is "file:/home/ioannou/books.bib#AbiteboulHV95".
The N-Triples found in the RDF Store corresponding to file "books.bib"

```
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/domain_l3s#Publication>
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#desktop_document_author>
   <file:/home/ioannou/books.bib#Abiteboul_
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#desktop_document_author>
   <file:/home/ioannou/books.bib#Hull_>
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#desktop_document_author>
   <file:/home/ioannou/books.bib#Vianu_>
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#file_stored_from>
   "books.bib"
<file:/home/ioannou/books.bib#AbiteboulHV95>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#text_title>
   "Foundations of Databases"
<file:/home/ioannou/books.bib#Abiteboul_>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person>
<file:/home/ioannou/books.bib#Abiteboul_>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#person_name>
   "Serge Abiteboul"
<file:/home/ioannou/books.bib#Vianu_>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person>
<file:/home/ioannou/books.bib#Vianu_>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#person_name>
   "Victor Vianu"
<file:/home/ioannou/books.bib#Hull_>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#Person>
<file:/home/ioannou/books.bib#Hull_>
   <http://www.kbs.uni-hannover.de/beagle++/ontology/desktop#person_name>
   "Richard Hull"
```

Resources and Workflows

Resources and Structures

For a description of the resources and structures that we annotate please refer to: https://www.l3s.de/web/upload/documents/1/pimo-report.pdf.

Required Workflows

Required workflows can be seen in PIMO description available at: https://www.l3s.de/web/upload/documents/1/pimo-report.pdf.

Literature

[1] Christoph Schmitz, Andreas Hotho, Robert Jaeschke, Gerd Stumme: Content Aggregation on Knowledge Bases using Graph Clustering. In Proceedings of the 3rd European Semantic Web Conference, Budva, Montenegro, 2006.

[2] FOAF. http://www.foaf-project.org

[3] FOAF specification. http://xmlns.com/foaf/0.1/