D3.2.1 Usage Tracking for Ontology Evolution

Peter Haase and York Sure
(Institute AIFB, University of Karlsruhe)

Abstract.
EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.2.1 (WP3.2)
This deliverable presents work performed in the task 'T3.2 Usage Tracking for Ontologies and Meta Data'. We address the problem of ontology evolution based on the usage context. We introduce the notion of an evaluation function that allows to measure the quality of an ontology with respect to given criteria. Within this framework, ontology evolution and discovering potentially useful changes can be formalized as an optimization problem. We instantiate the framework for the task of ontology pruning based on usage data, and for the task of collaborative evolution in a multi-user scenario, in which users maintain personalized ontologies.
Keyword list: ontology evolution, change discovery, ontology usage, ontology evaluation

Document Id.	SEKT/2005/D3.2.1/v1.1
Project	SEKT EU-IST-2003-506826
Date	July 24th, 2005
Distribution	public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006 Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma AI EAD, Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

The world is constantly changing, and so does required and available knowledge, e.g. stored in Digital Libraries. Knowledge workers heavily rely on the availability and accessibility of knowledge contained in such libraries. The sheer mass of knowledge available today, however, requires sophisticated support for searching and, often considered as equally important, personalization.

In SEKT we address these challenges by using ontologies. Ontologies by nature make implicit knowledge explicit, they describe relevant parts of the world and make them machine understandable and processable. To be effective, ontologies need to change possibly as fast as the parts of the world they describe

Change discovery aims at generating implicit requirements by inducing ontology changes from existing data. In this deliverable we focus on *usage-driven* change discovery. Usage data is a very valuable source of contextual information, based on which the ontology can be modified in order to reflect changes in the real world,

We describe a framework, in which ontology evolution and discovering potentially useful changes can be formalized as an optimization problem. We introduce the notion of an evaluation function that allows to measure the quality of an ontology with respect to given criteria. We instantiate the framework for the task of ontology pruning based on usage data, and for the task of collaborative evolution in a multi-user scenario, in which users maintain personalized ontologies. A first case study based on the Bibster system shows very promising results.

Contents

Chapter 1

Introduction

1.1 Motivation

The world is constantly changing, and so does required and available knowledge, e.g. stored in Digital Libraries. Knowledge workers heavily rely on the availability and accessability of knowledge contained in such libraries. The sheer mass of knowledge available today, however, requires sophisticated support for searching and, often considered as equally important, personalization.

In SEKT we address these challenges by using ontologies. Ontologies by nature make implicit knowledge explicit, they describe relevant parts of the world and make them machine understandable and processable. To be effective, ontologies need to change possibly as fast as the parts of the world they describe.

For the understanding of this deliverable it is important to distinguish between *change capturing* and *change discovery*.

The task of *change capturing* can be defined as the generation of ontology changes from explicit and implicit requirements. Explicit requirements are generated, for example, by ontology engineers who want to adapt the ontology to new requirements or by the end-users who provide the explicit feedback about the usability of ontology entities. The changes resulting from this kind of requirements are called top-down changes. Implicit requirements leading to so-called bottom-up changes are reflected in the behavior of the system and can be induced by applying change discovery methods.

Change discovery aims at generating implicit requirements by inducing ontology changes from existing data. [Sto04] defines three types of change discovery: (i) structure-driven, (ii) usage-driven and (iii) data-driven. Whereas structure-driven changes can be deduced from the ontology structure itself, usage-driven changes result from the usage patterns created over a period time. Data-driven changes are generated by modifications to the underlying data, such as text documents or a database, representing the knowledge modelled by an ontology.

In this deliverable we focus on the *usage-driven* change discovery. Usage data is a very valuable source of contextual information, based on which the ontology can be modified in order to reflect changes in the real world, changes in user's requirements, drawbacks in the initial design, to incorporate additional functionality or to allow for incremental improvement.

1.2 The SEKT Big Picture

This report is part of the work performed in workpackage (WP) 3 on "Ontology and Metadata Management", more specifically work performed in the task 'T3.2 Usage Tracking for Ontologies and Meta Data'. As shown in Figure 1.1 this work is closely related with other technical workpackages in SEKT. The main goal of this workpackage is to enable and to facilitate the setting up and maintenance of semantic knowledge management applications by supporting the complex tasks of managing ontologies and corresponding metadata.

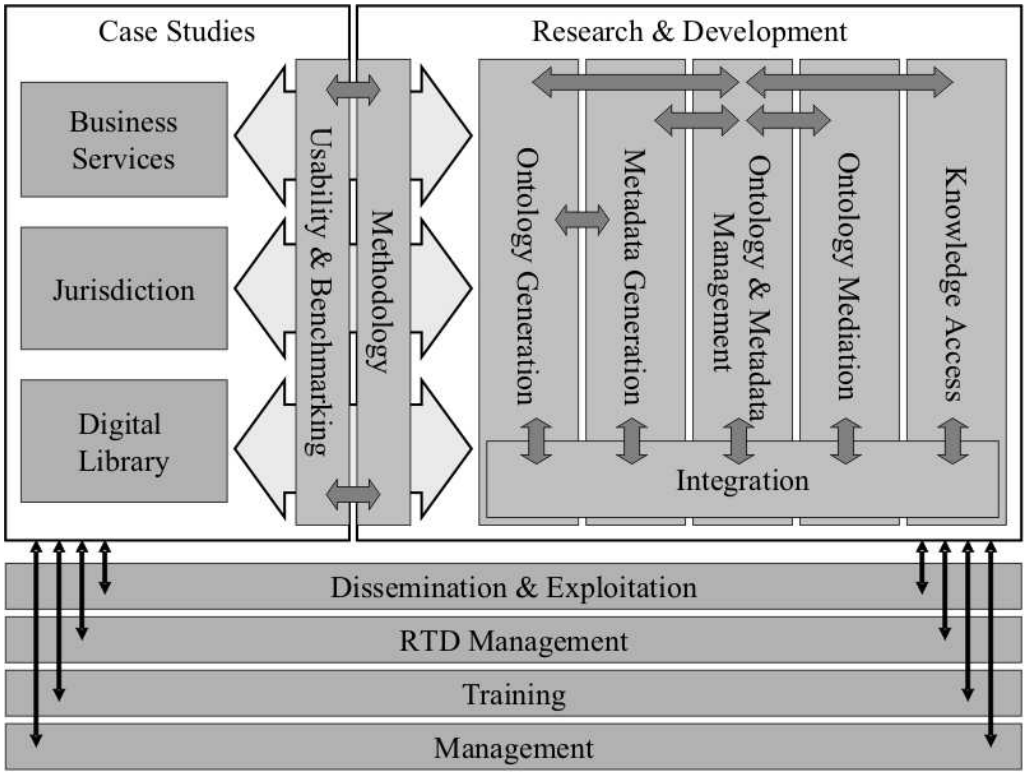

Figure 1.1: The SEKT Big Picture

1.3 Overview of the deliverable

In Chapter 2 we present a formal framework for the problem of ontology evolution and change discovery. We then present two instantiations of this framework: In Chapter 3 we show how the framework can be used for the task of ontology pruning based on an evaluation function that captures the cost incurred for accessing information structured using the ontology. In a second instantiation in Chapter 4, we present an approach to recommend ontology change operations to a personalized ontology based on the usage information of the individual ontologies in a user community. We conclude in Chapter 5.

In the rest of this chapter we explain how the deliverable is relevant to the SEKT case studies and present a logical architecture that integrates with the results of tasks T3.1 Incremental Ontology Evolution and T3.3 Data-driven Change Discovery.

1.4 Relation with the SEKT Case Studies

Ontologies play a crucial role in structuring information and enhancing information access in all case studies of the SEKT project. In the following we exemplarily illustrate the role of this work for the BT Digital Library case study.

In the BT Digital Library so-called information spaces have been created on topics known to be of interest to people in the company or through the contents of journals in the library. One of the key elements of the case study is to use information space *ontologies* to enhance the search in information spaces. Two main challenges for adapting such information space ontologies arise. First, the evolution of ontologies to reflect changing data, i.e. the documents stored in the Digital Library (addressed in task 3.3 “Data-driven Change Discovery”). Second, the evolution of ontologies to reflect changing interests of people.

For example, if none of the users of the Digital Library was interested in documents about a particular topic of the topic hierarchy, then, probably, this topic should be excluded from the list of concepts offered by that application. These ”discovered” changes are very important for optimizing performance of an application, e.g. by reducing the hierarchies of topics that have to be browsed. Moreover, they enable a continual adaptation of the application to the implicit changes in the business environment. Chapter 3 describes methods to discover such useful changes based on the usage data.

While the evolution support for a single ontology structuring an information space addresses the problem of *changing* interests, it does not consider the problem of *diverging* interests, i.e. that different people have different preferences with respect to how the information should be structured. Here, a personalization of the information spaces allows an organization of the documents according to the interests of the individual users, expressed in a personalized ontology. In such a scenario it makes sense to exploit changes in the ontologies of other, similar users’ ontologies to recommend potentially useful changes.

Such a collaborative scenario, in which changes are recommended based on methods from Collaborative Filtering, is described in Chapter 4.

1.5 Logical Architecture

In this section we will present a logical architecture to support the evolution of ontologies in a Digital Library, as shown in Figure 1.2.

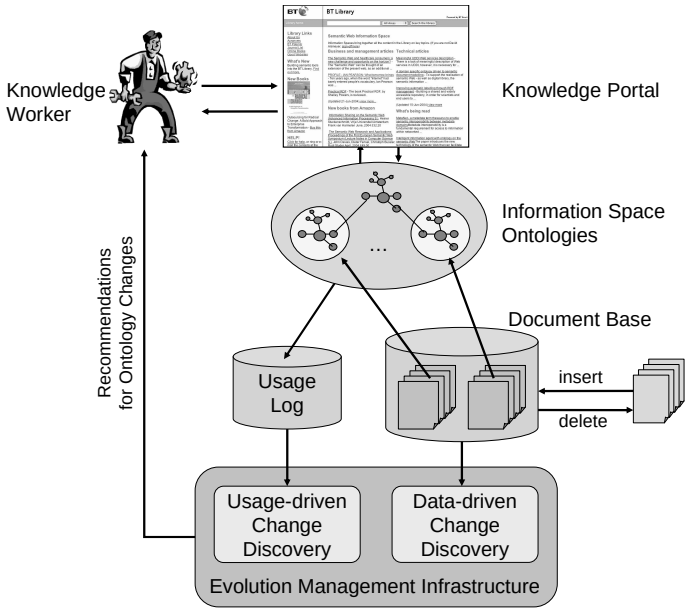

Figure 1.2: Logical Architecture

In this architecture, a knowledge worker interacts with a knowledge portal to access the content of the Digital Library, which is maintained in a document base and organized using ontologies in information spaces. The interaction is recorded in a usage log. This usage information and the information about changes in the document base are exploited to recommend changes to the ontologies, thus closing the loop with the knowledge worker.

Knowledge Worker The knowledge worker primarily *consumes* knowledge from the Digital Library. He uses the Digital Library to fulfil a particular information need. However, an advanced knowledge worker may also *contribute* to the Digital Library, either by contributing content or by organizing the existing content, providing metadata, etc. In particular, an advanced knowledge worker can take the role of an *ontology engineer*.

Knowledge Portal The knowledge worker interacts with the knowledge portal as the user interface. It allows the user to search the library's contents, it presents the content

in an organized way. The knowledge portal may also provide the knowledge worker with information in a proactive manner, e.g. by alerting, notification, etc.

Document Base The document base comprises a corpus of documents. In the context of the Digital Library, these documents are typically text documents, but may also include multimedia content such as audio, video, and images. While we treat the document as one logical unit, it may actually consist of a number of distributed sources. For example, in the case of the BT Digital Library, the document base includes two databases, Inspec and ABI / Inform.

The content of the document base typically is not static, but changes over time: New documents come in, but also documents may be removed from the the document base.

Information Spaces Information Spaces are logical units to organize a collection of documents according to a certain criterion. Information Spaces thus bring together content from the library's databases into a single a single place in the library.

One possible organization could be according to topics, e.g. there might be an information space to cover the topic *Semantic Web*. In the simplest case, an information space can be described with a search string. In general however, the description of the information space can be any formal specification.

It is also possible to support *personal information spaces*, i.e. an organization of the documents according to the interests of the individual knowledge workers. Such a personal information space can be specified with a semantic user profile.

Ontologies Ontologies are the basis for rich, semantic descriptions of the content in the Digital Library. Here we can identify two main modules of the ontology: The *application ontology* describes different generic aspects of bibliographic metadata (such as author, creation data) and are valid across various bibliographic sources.

Domain ontologies describe aspects that are specific to particular domains and is used as a conceptual backbone for structuring the domain information provided in the information spaces. Such a domain ontology typically comprises conceptual relations, such as a topic hierarchy, but also richer taxonomic and non-taxonomic relations.

While the application ontology can be assumed to be fairly static and valid across information spaces, the domain ontologies must be continuously adapted to the changing needs.

The ontologies are used for various purposes: First of all, the documents in the document base are annotated and classified according to the ontology. This ontological metadata can then be exploited for advanced knowledge access, including navigation, browsing, and semantic searches. Finally, the ontology can be used for the visualization of results, e.g. for displaying the relationships between information objects.

Usage Log The interaction of the knowledge worker with the knowledge portal is recorded in a usage log. It is of particular interest, how the ontology has been used in the interaction, i.e. which elements have been queried, which paths have been navigated, etc.

By tracking users' interactions with the application in a log file, it is possible to collect useful information that can be used to assess what the main interests of the users are. In this way, we are able to obtain implicit feedback and to extract needs for changes to the ontology to improve the interaction with the application.

Evolution Management Ontology evolution is timely adaptation of the ontology to changes and the consistent management of these changes. It is not a trivial process, due to the variety of sources and consequences of changes, cannot thus cannot be performed manually by the knowledge worker. This process is supported by the evolution management infrastructure. The first important aspect is the discovery of changes. While in some cases changes to the ontology may be requested explicitly, the actual challenge is to obtain and to examine the non-explicit but available knowledge about the needs of the end-users. This can be done by analysing various data sources related to the content that is described using the ontology and also the end-users' behaviour which include the information about her likes, dislikes, preferences or the way she behaves. Based on the analysis of this information, the knowledge worker can be suggested to make changes in the ontology resulting in an ontology better suited for the needs of end-users. In the following sections we will discuss the possibility of continuous ontology improvement by semi-automatic discovery of such changes, i.e. usage-driven and data-driven ontology evolution.

The second important aspect in the evolution process is to guarantee the consistency of the ontology when changes occur, considering the semantics of the ontology change. Here we refer the reader to [SMMS02] for further reading.

Chapter 2

General Framework

2.1 Ontology Model and Ontology Change Operations

2.1.1 Ontology Model

As the OWL ontology language has been standardized by the W3C consortium and the SEKT consortium has decided for OWL (more precisely OWL-DLP) as the default ontology language to use within the project, we will describe the framework in terms of the underlying OWL ontology model.

Because of their computational characteristics, the sublanguages OWL-DL, OWL-Lite, OWL-DLP are of particular importance. These languages are syntactic variants of the $\mathcal{SHOIN}(\mathbf{D})$, $\mathcal{SHIF}(\mathbf{D})$ and the Horn-fragment of $\mathcal{SHOIN}(\mathbf{D})$ description logics, respectively [HPS04]. In the following we will therefore use the more compact, traditional $\mathcal{SHOIN}(\mathbf{D})$ description logic syntax, which we review in the following:

We use a datatype theory $\mathbf{D}$, a set of concept names N_C, sets of abstract and concrete individual names N_{I_a} and N_{I_c}, respectively, and sets of abstract and concrete role names N_{R_a} and N_{R_c}, respectively.

The set of $\mathcal{SHOIN}(\mathbf{D})$ *concepts* is defined by the following syntactic rules, where A is an atomic concept, R is an abstract role, S is an abstract simple role (a role not having transitive subroles), $T_{(i)}$ are concrete roles, d is a concrete domain predicate, a_i and c_i are abstract and concrete individuals, respectively, and n is a non-negative integer:

$$
\begin{array}{lcl}
C & \rightarrow & A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \geq n\, S \mid \leq n\, S \mid \{a_1, \ldots, a_n\} \mid \\
 & & \mid \geq n\, T \mid \leq n\, T \mid \exists T_1, \ldots, T_n.D \mid \forall T_1, \ldots, T_n.D \\
D & \rightarrow & d \mid \{c_1, \ldots, c_n\}
\end{array}
$$

An ontology is a finite set of axioms of the form[1]:

- concept inclusion axioms $C \sqsubseteq D$, stating that the concept C is a subconcept of the concept D,
- transitivity axioms $\mathsf{Trans}(R)$, stating that the abstract role R is transitive,
- role inclusion axioms $R \sqsubseteq S$ ($T \sqsubseteq U$) stating that the abstract role R (or concrete role T) is a subrole of the abstract role S (or concrete role U).
- concept assertions $C(a)$ stating that the abstract individual a is in the extension of the concept C,
- abstract role assertions $R(a, b)$ and $T(a, c)$) stating that the abstract individuals a, b (or a, c) are in the extension of the role R (T),
- concrete role assertions $T(a, c)$) stating that the abstract individual a and the concrete individual c are in the extension of the concrete role T,
- individual (in)equalities $a \approx b$, and $a \not\approx b$, respectively, stating that a and b denote the same (different) individuals.

In the following, we denote the set of all possible ontologies with $\mathcal{O}$.

2.1.2 Ontology Change Operations

Definition 1 *An* ontology change operation *oco $\in$ OCO is a function oco : $\mathcal{O} \rightarrow \mathcal{O}$. Here OCO denotes the set of all possible ontology change operations.*

For the above defined ontology model, we allow the atomic change operations of adding and removing axioms, which we denote with α^+ and α^-, respectively. Complex ontology change operations can be expressed as a sequence of atomic ontology change operations. The semantics of the sequence is the chaining of the corresponding functions: For some atomic change operations $\text{oco}_1, ..., \text{oco}_n$ we can define $\text{oco}_{\text{complex}} = \text{oco}_n \circ ... \circ \text{oco}_1 = \text{oco}_n(...\text{oco}_1))$.

2.1.3 Ontology Rating Annotations

Our ontology model so far describes the actual state of an ontology as an isolated entity. Once we enter the more dynamic scenario of ontology evolution, it makes sense to consider contextual information about the ontology.

We model this contextual information by a rating annotation.

[1]For the direct model-theoretic semantics of $\mathcal{SHOIN}(\mathbf{D})$ we refer the reader to [HST00].

Definition 2 *Let $N := N_C \cup N_{I_a} \cup N_{I_c} \cup N_{R_a} \cup N_{R_c}$ denote the set of all possible names (symbols) and $\mathcal{A}$ the set of all possible axioms, then an ontology rating annotation is a partial function $r : N \cup \mathcal{A} \rightarrow \mathbb{R}$.*

The definition states that we allow ratings on both the axioms of the ontologies as well as the names over which the axioms are defined.

In the scope of this work, we use rating annotations to indicate the importance of particular elements. Here, high positive values denote high importance of a symbol or axiom, negative values that it is unwanted by the user.

Effectively we allow that a user (i) can express in a more fine-grained way how important a certain symbol (name) or axiom is for him and (ii) can express explicitly negative ratings for symbols (names) or axioms he does not want to be part of his ontology, In the context of software configuration management, the latter is known as specifying a ”taboo”.

In particular, we define the following two ontology rating annotations:

1. We use an explicit rating, called the membership-rating r^m with taboos, for which (i) all symbols and axioms actually part of the ontology have rating +1, (ii) all symbols and axioms not actually part of the ontology can be explicitly marked taboo by the user and then get a rating -1.

2. We use an implicit, usage-based rating called r^u, which indicates the relevance of the elements based on how they have been used, e.g. counts the percentage of queries issued by the user and instances in his knowledge base that reference a given symbol name.

We will consider rating annotations as an additional ontology component in the following.

2.2 Evaluation Function

In order to be able to define what a “good” ontology for a particular context is, we need to be able to measure the quality of the ontology with respect to given set of criteria, e.g. how well they reflect the underlying data they describe or how important they are based on the usage of the ontology. We therefore define the notion of an *ontology evaluation function*.

Definition 3 *Let V be a vocabulary and $\mathcal{O}$ the set of possible ontologies over V, then an ontology evaluation function is a function*

$$e : \mathcal{O} \times 2^{\mathbb{R}} \rightarrow \mathbb{R}$$

Effectively, the evaluation function provides a total order over the space of possible ontologies and thus allows to compare given ontologies. Here it is important to note that the evaluation function does not operate on the ontology itself, but takes a set of rating annotations into account and thus provides an evaluation measure for a given context.

2.3 Ontology Evolution

Based on the ontology evaluation function, we can now measure whether a particular change to an ontology leads to an "improvement" of the ontology for the given context. As this context changes over time, we can look at ontology evolution as adaptation to the changing context by discovering and applying changed to the ontology. Essentially, the goal is to discover changes that lead to a maximized evaluation function, i.e. the ideal ontology for the particular context:

Definition 4 *For a given ontology O and an evaluation function e, we can define the problem of change discovery as an optimization problem:*

$$max_{oco \in OCO}\ e(oco(O))$$

Of course, oco can be complex (and in fact will be complex in typical scenarios).

Having the problem stated as an optimization problem opens the door to applying established optimization techniques to find the "best" ontology with respect to the evaluation function. These can be either optimization techniques, heuristics-based techniques, but also *evolutionary elgorithms* (in particular genetic algorithms).

Speaking in terms of evolutionary algorithms (which have been inspired by biological evolution), the rating annotations define the environment in which the ontologies "live in", and the evaluation function determines the fitness of the ontology for this environment. The change operations serve as the "genetic operators".

Chapter 3

Usage Based Ontology Evolution

3.1 Background: Usage-driven Ontology Pruning

Our goal is to help an ontology engineer in the continual improvement of the ontology. This support can be split into two phases: (i) to help the ontology engineer find the changes that should be performed and (ii) to help him in performing such changes. The first phase is focused on discovering some anomalies in the ontology design, whose repairing improves the usability of the ontology. It results in a set of ontology changes. One important problem we face in developing an ontology is the creation of a hierarchy of concepts, since a hierarchy, depending on the users' needs, can be defined from various points of view and on the different levels of granularity. It is clear that the initial hierarchy has to be pruned, in order to fulfil the user's needs. Moreover, the users' needs can change over time, and the hierarchy should reflect such a migration. The usage of the hierarchy is the best way to estimate how a hierarchy corresponds to the needs of the users. Consider the example shown in Figure 3.1.

Let us assume that in the initial hierarchy (developed by using one of above-mentioned approaches), the concept X has ten subconcepts (c1, c2, ..., c10), i.e. an ontology engineer has found that these ten concepts correspond to the users' needs in the best way. However, the usage of this hierarchy in a longer period of time showed that about 95% of the users are interested in just three subconcepts (i.e. 95=40+32+23) out of these ten. It means that 95% of the users obtain 70% (i.e. 7 of 10 subconcepts) useless information via browsing this hierarchy, since they find seven subconcepts irrelevant. Consequently, this 95% of the users invest more time for performing a task than needed, since irrelevant information can get their attention. Moreover, there are more chances to make an accidental error (e.g. an accidental click on the wrong link), since the probability of selecting irrelevant information is greater.

In order to make this hierarchy more suitable to the users' needs, two ways of "restructuring" the initial hierarchy would be useful:

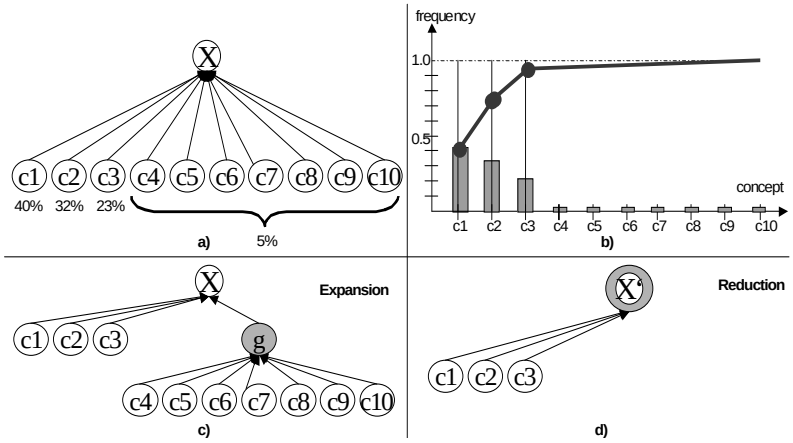

Figure 3.1: An example of the non-uniformity in the usage of the children. (a) the problem; (b) the Pareto diagram of the problem; (c) the resulting ontology after its extension and (d) the resulting ontology after its reduction.

1. **expansion** – to put down in the hierarchy all seven "irrelevant" subconcepts, while grouping them into a new subconcept g (see in Figure 3.1c);

2. **reduction** – to remove all seven "irrelevant" concepts, while redistributing their instances into remaining subconcepts or the parent concept (see in Figure 3.1d).

Through the expansion, the needs of the 5% of the users are preserved by the newly introduced concept and the remaining 95% of the users benefit from the more compact structure. By the reduction, the new structure corresponds completely to the needs of 95% of the users. The needs of 5% of the users are implicitly satisfied. Moreover, the usability of the ontology increased, since the instances which were hidden in the "irrelevant" subconcepts are now visible for additional 95% of the users. Consequently, these users might find them useful, although in the initial classification they are a priori considered as irrelevant (i.e. these instances were not considered at all). Note that the Pareto diagram shown in Figure 3.1b enables the automatic discovery of the minimal subset of the subconcepts which covers the needs of most of the users. For a formalization of this discovery process, including an evaluation study, we refer the interested reader to [SSGS03].

The problem of post-pruning a hierarchy in order to increase its usability is explored in the research related to modelling the user interface. The past work [BRS92] showed the importance of a balanced hierarchy for the efficient search through hierarchies of menus. Indeed, even though the generally accepted guidelines for the menu design favor breadth over depth [Kig84], the problem with the breadth hierarchy in large-scale systems is that

the number of items at each level may be overwhelming. Hence, a depth hierarchy that limits the number of items at each level may be more effective. This is the so-called breadth/depth trade-off.

Although there are some methods for an automatic hierarchy generation, such a hierarchy has to be manually pruned, in order to ensure the usability of the hierarchy. The main criterion is the *coherence* of the hierarchy (some kind of the uniform distribution of documents in all parts of the hierarchy), which ensures that the hierarchy is closely tailored to the needs of the intended user.

A weak point of the existing methods is that they are based on *heuristics* to discover potentially useful changes. These heuristics typically cannot guarantee that a change actually leads to an improvement, Also, as typically these heuristics are based only on local information about the neighborhood of a node in a graph, they do not necessarily lead to global optimization: A change may lead to resolving a local problem, but may introduce new ones in other places.

3.2 Usage rating model

In our model we consider a class hierarchy, which is used to classify a set of instances. With respect to the ontology model, it does not matter at this point, whether these classes are modelled as concepts and subconcept-axioms, or whether they are modelled as individuals and an additional relation between the individuals representing the class hierarchy. For a concept c in the concept hierarchy, $depth(c)$ denotes the depth of c in the hierarchy, whereas $breadth(c)$ denotes the breadth, i.e. the number of siblings of c.

We consider two usage rating annotations:

- $r_{queries}$ annotates the concepts with the number of queries that have referenced the particular concept,
- $r_{instances}$ annotates the concepts with the number of instances that are classified under the particular concept.

The two rating annotations capture two important and typical dimensions of usage, one with respect to the content (how the concepts are used to classify instances, and one with respect to the usage by the end users, i.e. which concepts were actually queried. This information is available in a wide range of application scenarios. Of course, in specific scenarios further information may be available and thus additional rating annotations can be defined.

3.3 Cost-based Evaluation Functions

With our evaluation function we try to capture that the quality of an ontology is determined by how effectively it allows the users to obtain the relevant instances. To measure the effectiveness, we introduce a cost model to allow to quantify the user effort to arrive at the desired information. For the case of navigating a hierarchy, this cost is determined by the complexity of the hierarchy in terms of the breadth and depth: The breadth here means the number of choices (child nodes) the user has to consider to decide for the right branch to follow: The broader the hierarchy, the longer it takes to make the correct choice. The depth means, how many links the user needs to follow to arrive at the correct concept, under which the desired instance is classified: The deeper the hierarchy, the more "clicks" need to be performed. To minimize the cost, both depth and breadth need to be minimized, i.e. the right balance between them needs to be found.

A very simple, but intuitive cost function, is presented in the following. For a given ontology O with the set of concepts C, we can calculate the cost as the weighted sum of the cost of the individual concepts, where the weight is the importance of the concept according to the rating annotations $r_{queries}$, i.e. the number of queries:

$$Cost(O) = \sum_{c \in C} r_{queries}(c) * cost(c)$$

The cost of the individual concepts is:

$$cost(c) = cost(parent(c)) + (k_d * depth(c) + k_b * breadth(c))$$

Essentially, the cost is determined by the cost of the parent-concept ($parent(c)$), plus the cost for following the link (weighted by a constant k_d) and the cost caused by the breadth of c (weighted by a constant k_b).

Example: ACM Topic Ontology In our example we use the ACM Topic Hierarchy. This topic hierarchy describes specific categories of literature for the Computer Science domain. It covers large areas of computer science, containing over 1287 topics ordered using taxonomic relations. The ACM classification is updated when necessary to reflect changes in the domain. As a change to the classification scheme is a mainly manual task with immense implications on existing physical and Digital Libraries, these changes are rare, costly, but unavoidable. Automated support, e.g. to proactively recommend useful changes, would be a great benefit. In personal information sharing systems, where users rely on a shared background ontology, but may their personal views or extensions (c.f. Chapter 4), such automated recommendations are of particular value.

Figure 3.2 shows a small fragment of the ACM Topic Hierarchy in the 1998 version. Table 3.1 exemplarily shows rating annotations for the corresponding concepts and a calculation for $Cost(O)$ according to the cost model described above. Please note that in the

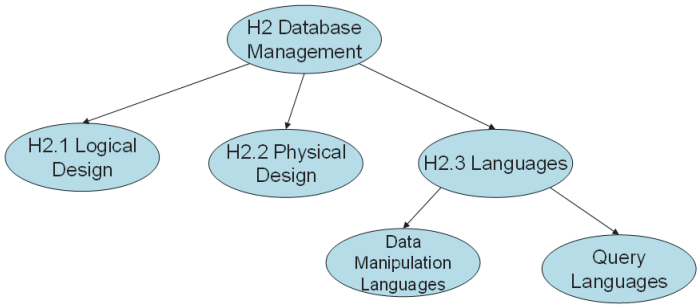

Figure 3.2: Fragment of the ACM Topic Ontology

Concept c	$r_{queries}(c)$	$cost(c)$
H2 Database Management	0	2
H2.1 Logical Design	1	6
H2.2 Physical Design	2	6
H2.3 Languages	0	6
Data Manipulation Languages	5	9
Query Languages	6	9
$Cost(O)$		117

Table 3.1: Rating Annotations and Costs (1)

example instances are classified only under leaf concepts, none-leaf concepts are therefore rated with 0. We see that the concepts Logical Design and Physical Design are fairly unimportant compared with Query Languages and Data Manipulation Languages according to their ratings. However, the more important concepts are deeper in the hierarchy, increasing the cost for accessing relevant instances. In the following we exemplarily show how to generate changes to alleviate this problem.

3.4 Generation of Changes

[SSGS03] presents useful heuristics for the generation of changes based on local properties. However, as mentioned before these heuristics do not guarantee that a change leads to a global improvement of the ontology. In the following we describe a selection of two relevant complex change operations [1] and illustrate based on the example from the previous section, how their application can improve the ontology according to the evaluation function.

[1] Of course, further complex change operations are possible. In this sense, the selection of these two change operations has exemplary character.

3.4.1 Grouping concepts

One possible change is to group concepts with low ratings under a newly introduced concept. This decreases the breadth at the level of the new concept and thus the cost of the sibling concepts. Figure 3.3 shows the previous example after a change that grouped the unimportant concepts Logical Design and Physical Design under a new concept Design. Table 3.2 shows the reduction in cost.

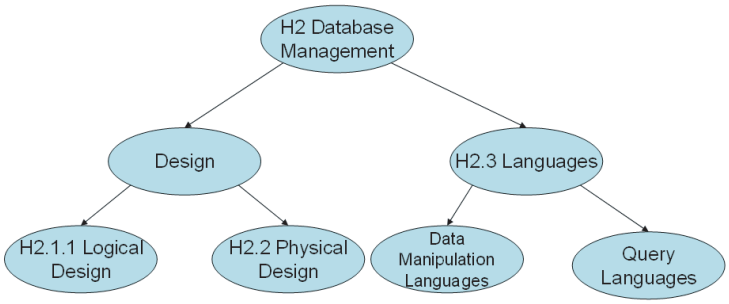

Figure 3.3: Fragment of the ACM Topic Ontology after Grouping Concepts

Concept c	$r_{queries}(c)$	$cost(c)$
H2 Database Management	0	2
H2.1+2 Design	1	5
H2.1 Logical Design	1	8
H2.2 Physical Design	2	8
H2.3 Languages	0	5
Data Manipulation Languages	5	8
Query Languages	6	8
$Cost(O)$		112

Table 3.2: Rating Annotations and Costs (2)

3.4.2 Pulling-up concepts

An alternative change is to “pull-up” important concepts that have previously been grouped under a common parent concept. The result is that the depth of these concepts is reduced, resulting in a decreased cost. Figure 3.4 shows the running example after a change that pulled-up the important concepts Data Manipulation Languages and Query Languages by removing the common parent concept Query Languages. Table 3.3 shows the reduction in cost.

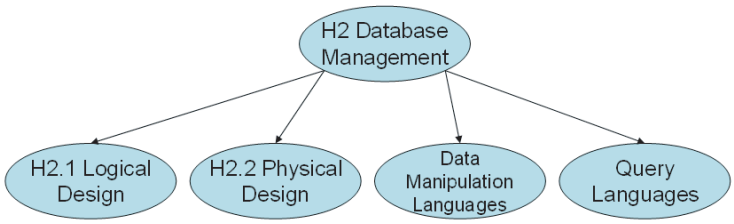

Figure 3.4: Fragment of the ACM Topic Ontology after Pulling-up Concepts

Concept c	$r_{queries}(c)$	$cost(c)$
H2 Database Management	0	2
H2.1 Logical Design	1	7
H2.2 Physical Design	2	7
Data Manipulation Languages	5	7
Query Languages	6	7
$Cost(O)$		98

Table 3.3: Rating Annotations and Costs (3)

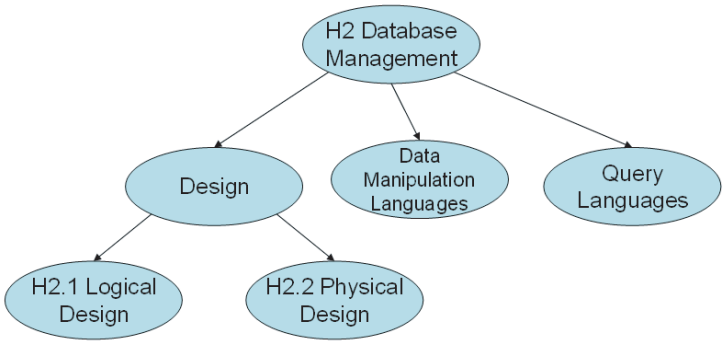

Figure 3.5: Fragment of the ACM Topic Ontology after Splitting and Merging Concepts

Concept c	$r_{queries}(c)$	$cost(c)$
H2 Database Management	0	2
H2.1+2 Design	1	6
H2.1 Logical Design	1	9
H2.2 Physical Design	2	9
Data Manipulation Languages	5	6
Query Languages	6	6
$Cost(O)$		93

Table 3.4: Rating Annotations and Costs (4)

Chapter 4

Collaborative and Usage-driven Evolution

4.1 Introduction

Large information repositories as digital libraries, online shops, etc. rely on a taxonomy of the objects under consideration to structure the vast contents and facilitate browsing and searching (e.g., ACM Topic Hierarchy for computer science literature, Amazon product taxonomy, etc.). As in heterogenous communities users typically will use different parts of such an ontology with varying intensity, customization and personalization of the ontologies is desirable.

Such personal ontologies reflect the interests of users at certain times. Interests might change as well as the available data, therefore the personalization requires quite naturally support for the evolution of personal ontologies. The sheer size of e.g. the ACM Topic Hierarchy makes it quite difficult for users to easily locate topics which are relevant for them. Often one can benefit from having a community of users which allows for recommending relevant topics according to similar interests. Of particular interest are therefore collaborative filtering systems which can produce personal recommendations by computing the similarity between own preferences and the one of other people.

We performed our evaluation within the Bibster community. Bibster is a semantics-based Peer-to-Peer application aiming at researchers who want to benefit from sharing bibliographic metadata. It enables the management of bibliographic metadata in a Peer-to-Peer fashion: it allows to import bibliographic metadata, e.g. from BIBTEX files, into a local knowledge repository, to share and search the knowledge in the Peer-to-Peer system, as well as to edit and export the bibliographic metadata.

As our main contribution we adapt a collaborative filtering recommender system to assist users in the management and evolution of their personal ontology by providing detailed suggestions of ontology changes. The approach is implemented as an extension

of the Bibster application and has been thoroughly evaluated with very promising results.

The chapter is structured as follows. In the next Section 4.2 we present related work in the areas of work in recommender systems, work in using taxonomies in recommender systems, and work in learning taxonomies and ontology evolution in general. In Section 2.1 we describe our underlying ontology model which is based on OWL, the change operations used during the evolution of ontologies, and the ontology rating annotations allowing each user to express more fine-grained the importance of certain ontology parts. The recommender method itself and its functionality is illustrated in Section 4.3. We will introduce the Bibster applications and its extensions with the recommender functionality in Section 4.4 followed by evaluation results in Section 4.5. The evaluation was performed as an experiment within the Bibster community and shows the performance improvements over non-personalized recommendations.

4.2 Related Work

Related work exists in three different aspects: work in recommender systems, especially collaborative filtering in general, work in using taxonomies in recommender systems, and work in learning taxonomies and ontology evolution in general.

Recommender systems have their roots in relevance feedback in information retrieval [Sal71], i.e., adding terms to (query expansion) or re-weighting terms of (term re-weighting) a query to a document repository based on terms in documents in the result set of the original query that have been marked relevant or non-relevant by the user, as well as adaptive hypertext and hypermedia [SF91], i.e., the automatic adaptation of the link structure of a document repository based on previous link usage by users.

Recommender systems broaden the domain from documents and link structure to arbitrary domains (e.g., movies, products), do not necessarily rely on attributes of the objects under consideration (i.e., terms in the case of documents and called *items* in the context of recommender systems), and typically combine knowledge about different users. They first have been formulated as filtering techniques generally grouped in three different types:

(1) *collaborative filtering* is basically a nearest-neighbor model based on user–item correlations; if correlations are computed between users, it is called *user-based*, if between items, it is called *item-based.* (2) *content-based* or *feature-based* recommender systems use similarities between rated items of a single user and items in the repository. User- and item-based collaborative filtering and content-based recommender systems have been introduced in [GNOT92, RIS+94], [SM95] and [BS97], respectively, and are exemplified by the three systems presented there, MovieLens, Ringo, and fab. (3) *Hybrid* recommender systems try to combine both approaches [BS97, Bur02]. Although most recommender systems research meanwhile focuses on ore complex models treating the task as a learning or classification problem, collaborative filtering models still are under active investigation [HKTR04, DK04] due to their simplicity and comparable fair

quality.

Taxonomies are used in recommender systems to improve recommendation quality for items, e.g., in [MSR04] and [ZSTL04]. But to our knowledge there is no former approach for the inverse task, to use recommender systems for the personalization of the taxonomy or more generally of an ontology.

Ontology evolution is a central task in ontology management that has been addressed for example in [KN03] and [SMMS02]. In [SMMS02] the authors identify a possible six-phase evolution process: (1) change capturing, (2) change representation, (3) semantics of change, (4) change implementation, (5) change propagation, and (6) change validation. Our work addresses the phase of change capturing, more specifically the process of capturing implicit requirements for ontology changes from usage information about the ontology. One approach for *usage-driven change discovery* in ontology management systems has been explored in [SS02], where the user's behavior during the knowledge providing and searching phase is analyzed. [SHG03] describes a tool for guiding ontology managers through the modification of an ontology based on the analysis of end-users' interactions with ontology-based applications, which are tracked in a usage-log.

However, the existing work only addressed the evolution of a single ontology in a centralized scenario. In our work we are extending the idea of applying usage-information to a multi-ontology model by using collaborative filtering to recommend ontology changes based on the usage of the personal ontologies.

4.3 Recommending Ontology Changes

A recommender system for ontology changes tries to suggest ontology changes to the user based on some information about him and potential other users. Formally, an *ontology recommender* is a function

$$\varrho : \mathcal{X} \rightarrow 2^{\mathrm{OCO}} \tag{4.1}$$

where $\mathcal{X}$ contains suitable descriptions of the target ontology and user.

For example, let recommendations depend only on the actual state of a user's ontology, i.e., $\mathcal{X} = \mathcal{O}$, where $\mathcal{O}$ denotes the set of possible ontologies. A simple ontology evolution recommender can be built by just evaluating some heuristics on the actual state of the ontology, e.g., if the number of instances of a concept exceeds a given threshold, it recommends to add subconcepts to this concept. But without any additional information, this is hardly very useful, as we would not be able to give any semantics to these subconcepts: we could recommend to further subdivde the concept, but not how, i.e., neither be able to suggest a suitable label for these subconcepts nor assertions of instances to them. We will call such an approach *content-based* to distinguish it from more complex ones.

Recommendation quality eventually can be improved by taking into account other

users' ontologies and thereby establishing some kind of collaborative ontology evolution scenario, where each user keeps his personal ontology but still profits from annotations of other users. The basic idea is as follows: assume that for a target ontology we know similar ontologies called *neighbors* for short, then we would like to spot patterns in similar ontologies that are absent in our target ontology and recommend them to the target ontology. Another wording of the same idea is that we would like to extract ontology change operations that applied to the target ontology increases the similarity with its neighbors Ω.

This criterion can directly be used to formulate an ontology evaluation function (c.f. equation 4.6 below).

Let

$$\mathrm{sim} : \mathcal{O} \times \mathcal{O} \to \mathbb{R} \tag{4.2}$$

be such a similarity measure where $\mathrm{sim}(O, P)$ is large for similar ontologies O and P and small for dissimilar ontologies. Typically, these measures are symmetric and maximal for two same arguments. For further properties and examples of similarity functions for ontologies, we refer the reader to [EHS04].

Recall that ontologies in our context may have additional rating annotations that are valuable information to consider in similarity measures suitable for recommendation tasks.

We can choose a simple unnormalized correlation measure (vector similarity) to compute similarities between ontologies of two users based on their ratings of the elements (i.e. symbol names and axioms) in the ontology:

$$\mathrm{sim}_r(O, P) := \frac{\sum_{s \in N \cup \mathcal{A}} r_O(s)\, r_P(s)}{\sqrt{\sum_{s \in N \cup \mathcal{A}} r_O(s)^2} \sqrt{\sum_{s \in N \cup \mathcal{A}} r_P(s)^2}} \tag{4.3}$$

Similarities for the two different rating annotations r^m and r^u are computed separately and then linearly combined with equal weights:

$$\mathrm{sim}(O, P) := \frac{1}{2}\,\mathrm{sim}_{r^m}(O, P) + \frac{1}{2}\,\mathrm{sim}_{r^u}(O, P) \tag{4.4}$$

Finally, as in standard user-based collaborative filtering, ratings of all neighbors Ω are aggregated using the similarity-weighted sum of their membership ratings r^m, allowing for a personalized recommender function:

$$r_{\textit{personalized}}(O, \Omega, c) := \frac{\sum_{P \in \Omega} \mathrm{sim}(O, P)\, r^m_P(c)}{\sum_{P \in \Omega} |\,\mathrm{sim}(O, P)|} \tag{4.5}$$

The recommendations are obtained directly from the rating: Elements with a positive

rating are recommended to be added to the ontology, elements with a negative rating are recommended to be removed.

Let us consider the recommended changes in terms of an ontology evaluation function. As stated above, the goal is to increase the weighted similarity of an ontology with those of the neighbors in Ω:

$$e_\Omega(O) := \sum_{P \in \Omega} \mathrm{sim}(O, P) * \mathrm{sim}_{r^m}(O, P) \tag{4.6}$$

The similarity of the states of the ontologies is determined by $\mathrm{sim}_{r^m}(O, P)$, whereas $\mathrm{sim}(O, P)$ serves as a weighting factor. Essentially, the recommendation function tries to exactly recommend those changes whose application results in an increase of the value of the ontology according to the ontology evaluation function.

Disregarding the similarity measure between the users' ontologies, we can build a naive recommender that does not provide personalized recommendations, but instead simply recommends "most popular" operations based on an unweighed average of the membership ratings:

$$r_{\textit{baseline}}(O, \Omega, c) = \frac{\sum_{P \in \Omega} r_P^m(c)}{|\Omega|} \tag{4.7}$$

4.4 Case Study: Bibster

In this section we will first introduce the Bibster system [HBE+04] and the role of personalized ontologies in its application scenario. We will then describe how the recommender functionality is applied in the system to support the users in evolving their personalized ontologies.

4.4.1 Application Scenario: Sharing Bibliographic Metadata with Bibster

Bibster[1] is an award-winning semantics-based Peer-to-Peer application aiming at researchers who want to benefit from sharing bibliographic metadata. Many researchers in computer science keep lists of bibliographic metadata, preferably in BibTeX format, that they must laboriously maintain manually.

At the same time, many researchers are willing to share these resources, assuming they do not have to invest work in doing so.

[1] `http://bibster.semanticweb.org/`

Bibster enables the management of bibliographic metadata in a Peer-to-Peer fashion: it allows to import bibliographic metadata, e.g. from BIBTEX files, into a local knowledge repository, to share and search the knowledge in the Peer-to-Peer system, as well as to edit and export the bibliographic metadata.

Two ontologies are used to describe properties of bibliographic entries in Bibster, an application ontology and a domain ontology [Gua98]. Bibster makes a rather strong commitment to the application ontology, but the domain ontology can be easily substituted to allow for the adaption to different domains.

Bibster uses the SWRC[2] ontology as application ontology, that describes different generic aspects of bibliographic metadata. The SWRC ontology has been used already in various projects, e.g. also in the semantic portal of the Institute AIFB[3].

In our scenario we use the ACM Topic Hierarchy[4] as the domain ontology. This topic hierarchy describes specific categories of literature for the Computer Science domain. It covers large areas of computer science, containing over 1287 topics ordered using taxonomic relations, e.g.:

$$SubTopic(Artificial\_Intelligence, Knowledge\_Representation\_Formalisms).$$

The $SubTopic$ relation is transitive, i.e. $\mathsf{Trans}(SubTopic)$.

The domain ontology is being used for classification of metadata entries, e.g.

$isAbout(someArticle, Artificial\_Intelligence)$, therefore enabling advanced querying and browsing. The classification can be done automatically by the application or manually (by drag and drop).

4.4.2 Extensions for Evolution and Recommendations

In Bibster we initially assumed both ontologies to be globally shared and static. This basically holds for the application ontology, but users want to adapt the domain ontology continuously to their needs. This is largely motivated by the sheer size of the ACM Topic Hierarchy which makes browsing, and therefore also querying and manual classification, difficult for users.

As part of this work we implemented extensions as described in the previous Section 4.3 to Bibster which support the evolution – i.e. the continuous adaptation – of the domain ontology by the users. A basic assumption here is that all users agree in general on the ACM Topic Hierarchy as domain ontology, but each user is only interested in seeing

[2] http://ontoware.org/projects/swrc/
[3] http://www.aifb.uni-karlsruhe.de/about.html
[4] http://www.acm.org/class/1998/

those parts of it which are relevant for him at a certain point of time.

In the application, we have separated the interaction with the ontology in two modes: a *usage mode* and an *evolution mode*. The usage mode is active for the management of the bibliographic metadata itself, i.e. creating and searching for the bibliographic metadata. This mode only shows the current view on the ontology consisting of the topics that the user has explicitly included in his ontology. The evolution mode allows for the adaptation of the ontology. In this mode also the possible extensions along with the corresponding recommendations are shown.

Ontology Change Operations To keep things simple and trying to separate effects from eventually different sources as much as possible, we allow as change operations the addition and removal of topics from the personal ontology. More specifically, this addition/removal corresponds to the addition/removal of the individual assertion axiom (e.g. $Topic(Knowledge\_Representation\_Formalisms)$ and the role assertion axiom that fixes the position in the topic hierarchy (e.g. $SubTopic(Artificial\_Intelligence, Knowledge\_Representation\_Formalisms)$). The addition of topics is restricted to those topics that are predefined in the ACM Topic Hierarchy. Also, the position of the topics is fixed globally by the background ontology.

Ontology Ratings To elicit as much information as possible from users' work with the application, we gather various ontology rating annotations in the different modes.

We obtain the membership-rating r^m in the evolution mode from the explicit user actions (c.f. Figure 4.2): The user can either add a topic in the taxonomy, which will assigning a rating +1 for the topic, or he can exclude (taboo) the topic from the taxonomy, which will assign -1 for the explicitly taboo-ed topic.

We obtain the usage-based rating r^u in the usage mode by counting the percentage of queries issued by the user and instances in his knowledge base that reference a given topic.

(For this, references to all topics are retained, especially also to topics not contained in the ontology of the user.)

The ontology ratings of the individual users are propagated together with peer profile descriptions as advertisements in the Peer-to-Peer network, such that every peers is informed about the usage of the ontology in the network. For the details of this process, we refer the reader to [HBE+04].

Recommending Ontology Changes For the recommendations of topics we rely on the rating function $r_{\text{personalized}}$ presented in the previous section. From the ratings of the topics, we can directly obtain the recommendations:

Topics with a positive rating are recommended to be added to the ontology, topics

with a negative rating are recommended to be removed. (Please note that adding a topic actually means adding the corresponding axioms, as described above.)

Topics in the topic hierarchy are visualized depending on the current rating r^m of the topic and on the recommendation for the topic using a the coding scheme shown in Figure 4.1.

Rating	Recommendation		
	Remove	Neutral	Add
Taboo-ed	X `topicname`	X `topicname`	`+ topicname`
Unrated	`- topicname`	`? topicname`	`+ topicname`
Accepted	`- topicname`	√ `topicname`	√ `topicname`

Figure 4.1: Visualization of topics in evolution mode

Figure 4.2 shows a screenshot of the ontology in the evolution mode.

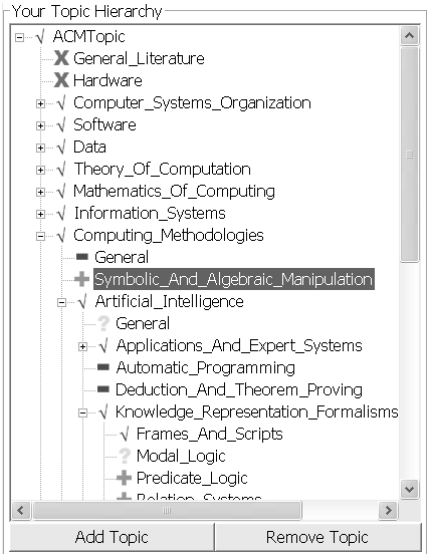

Figure 4.2: Screenshot

4.5 Evaluation

For our evaluation, we wanted to study two questions: (i) do users accept recommendations for ontology changes at all? (ii) is a personalized recommender better suited for the task than a naive, non-personalized recommender?

To answer these questions, we have performed a user experiment in an in-situ setting using the Bibster system, in which we compared the baseline (non-personalized) and the personalized recommender, as defined in the previous section. In the following we will describe the setup of the experiment, evaluation measures, and the results.

4.5.1 Design of the Experiment

The experiment was performed within three Computer Science departments at different locations. For a pre-arranged period of one hour, 23 users were actively using the system. The recommender strategy (baseline or personalized) was chosen randomly for each user at the first start of the Bibster application. The users were not aware of the existence of the different recommendation strategies.

During the experiment, the users performed the following activities (in no particular order), which are typical for the everyday use of the system:

- *Import data:* The users need to load their personal bibliography as initial dataset. This data should also reflect their research interest. As described before, the classification information of the bibliographic instances is part of the ontology rating and thus used to compute the similarity between the peers.
- *Perform queries:* The users were asked to search for bibliographic entries of their interest by performing queries in the Peer-to-Peer system. These queries may refer to specific topics in the ontology, and are thus again used as ontology ratings.
- *Adapt ontology:* Finally the users were asked to adapt their ontology to their personal needs and interests by adding or removing topics. This process was guided by the recommendations of the respective recommender function. The recommendations were updated (recalculated) after every ontology change operation.

The user actions were logged at every peer for later analysis. The logged information included: The type of the action (e.g. user query, ontology change operations), the provided recommendations, and a timestamp.

4.5.2 Evaluation Measures

We base our evaluation on the collected usage information in form of events consisting of the actual user action $e \in \text{OCO}$, i.e., the specific ontology change operation performed, and the set $\hat{E} \subseteq \text{OCO}$ of recommendations at that point in time, represented by a set $\mathcal{E} \subseteq \text{OCO} \times 2^{\text{OCO}}$.

We observe a successful recommendation or a *hit*, when $e \in \hat{E}$.

User Action	Recommendation Remove	None	Add
Remove	1	0	-1
None	0	0	0
Add	-1	0	1

Table 4.1: Evaluation Profit Matrix

For non-hits, we distinguish two situations: (i) If the actual recommendation was exactly the opposite action, e.g., we recommended to add a topic but the user taboo-ed it, then we call this an *error*. (ii) If there was no recommendation for this action neither for its opposite, we call this *restraint*. Based on these counts, we can compute the following performance measures.

$$\mathit{recall}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid e\in\hat{E}\}|}{|\mathcal{E}|} \tag{4.8}$$

$$\mathit{error}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid \mathrm{opp}(e)\in\hat{E}\}|}{|\mathcal{E}|} \tag{4.9}$$

$$\mathit{restraint}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid \mathrm{opp}(e)\notin\hat{E}\wedge e\notin\hat{E}\}|}{|\mathcal{E}|} \tag{4.10}$$

where opp denotes the respective opposite operation, e.g., $\mathrm{opp}(e^+) := e^-$ and $\mathrm{opp}(e^-) := e^+$.

Higher recall and lower error and restraint are better.

For a higher level of detail, we do so not only for all user actions, but also for some classes OCOC $\subseteq$ OCO of user actions, such as all *add*- and all *remove/taboo*-operations.

As each of the measures alone can be optimized by a trivial strategy, we also computed the profit of the recommenders w.r.t. the profit matrix in Table 4.1:

$$\mathit{profit}(\mathcal{E}) := \frac{\sum_{(e,\hat{E})\in\mathcal{E}} \sum_{\hat{e}\in\hat{E}} \mathit{profit}(e,\hat{e})}{|\mathcal{E}|} = \mathit{recall}(\mathcal{E}) - \mathit{error}(\mathcal{E}) \tag{4.11}$$

An intuitive reading of the profit is: The higher the profit, the better the performance of the recommender. In the best case ($profit = 1$), all user actions were correctly recommended by the system, in the worst case ($profit = -1$), all user actions were opposite of the recommendation.

4.5.3 Evaluation Results

For the 23 participating users in the experiment, the baseline recommender was active for 10 users, the personalized recommender was active for the other 13 users. The participants performed a total of 669 user actions (452 add topic and 217 remove topic), 335

ACM Topic	# Add Actions
Information_Systems	23
Computing_Methodologies	15
Data	14
Computing_Methodologies/Artificial_Intelligence	12
Information_Systems/Database_Management	12
Software	11
Mathematics_Of_Computing	10
Computer_Systems_Organization	10
Computer_Systems_Organization/Computer_Communication_Networks	10
Computing_Methodologies/Artificial_Intelligence/ Knowledge_Representation_Formalisms_And_Methods	10

Table 4.2: Most Popular Topics

of these action were performed by users with the baseline strategy, 334 by users with the personalized recommender. Table 4.2 shows the number of add-topic-actions for the most popular topics.

Figure 4.3 shows the cumulative results of the performance measures defined above for the baseline and the personalized recommender. The diagrams show the results for *Add* and *Remove* operations separately, as well as combined for all change operations.

As we can see in Figure 4.3 (upper right), overall the personalized recommender correctly recommended more than 55% of the user actions, while the baseline achieved less than 30%. The error rate of the baseline algorithm is considerably higher: We observed an *error* $= 17\%$ and 9% for the baseline and the personalized approach, respectively. Further we observed a very large amount of restraint operations with *restraint* $= 67\%$ for users with the baseline strategy. Probably this is the result of a large number of recommendations irrelevant to the user given by the system with the baseline strategy. In such a case the user would not like to follow the system and constructs the ontology mainly by themselves. Only from time to time he takes some of the recommendations into account.

By comparing add and remove operations we observe a considerably higher amount of *error* recommendations for remove operations in comparison to the really small amount for the add recommendations while the correct recommendations are comparable for both operations (cf. Figure 4.3, left side). We think that this observation is based on the fact that a user is more likely to follow an add operation without a "substantiated" reason or explanation than a remove operation.

While adding something to his "collection" and following the idea of having more the remove operation forces the feeling of "loosing" something, so typically users are more reluctant to remove topics.

Calculating the overall profit of the two recommender functions, we obtain

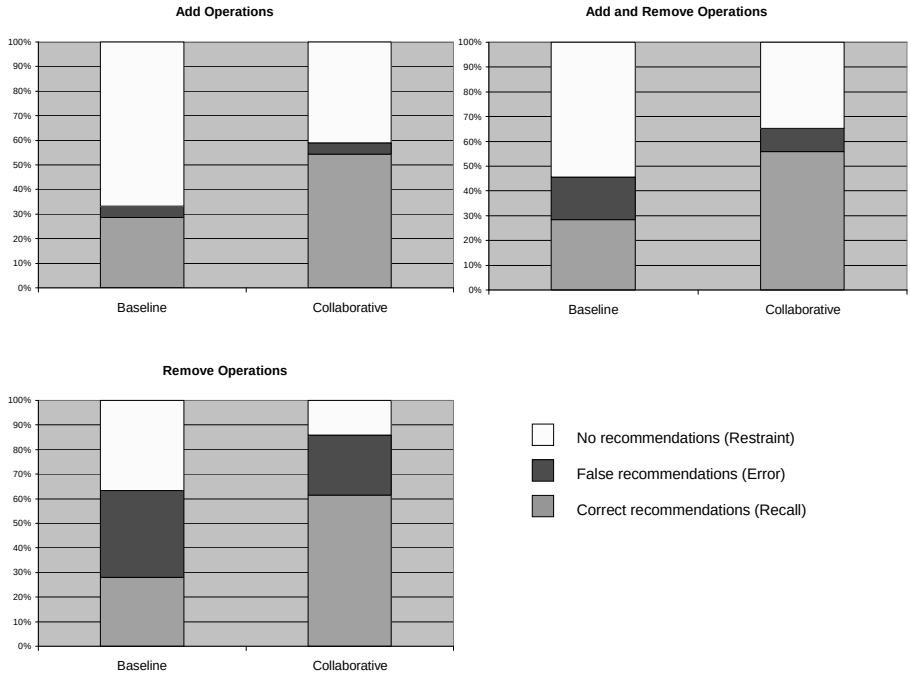

Figure 4.3: Performance measures of the recommender

profit($\mathcal{E}$) $= 0.11$ for the baseline recommender. For the collaborative recommender, we obtain a significantly better value of *profit*($\mathcal{E}$) $= 0.47$. Concluding we can state that the personalized recommender function provides substantially more useful recommendations.

Chapter 5

Conclusion

In this deliverable we have presented an approach to the problem of change discovery by describing a framework based on the notion of an ontology evaluation function that allows to measure the quality of on ontology with respect to given criteria. While this framework is general enough to define evaluation functions for arbitrary contexts, we have applied it for the problem of change discovery based on the usage of the ontology.

In a first instantiation we have shown how the framework can be used for the task of ontology pruning based on an evaluation function that captures the cost incurred for accessing information structured using the ontology.

In a second instantiation, we have presented an approach to recommend ontology change operations to a personalized ontology based on the usage information of the individual ontologies in a user community. In this approach we have adapted a collaborative filtering algorithm to determine the relevance of ontology change operations based on the similarity of the users' ontologies.

In our experimental evaluation with the Peer-to-Peer system Bibster we have seen that the users actually accept recommendations of the system for the evolution of their personal ontologies. The results further show the benefit of exploiting the similarity between the users' ontologies in a personalized recommender compared with a simple, non-personalized baseline recommender. In our experiment we have made various simplifying assumptions. Their relaxation will open fruitful directions for future work: We assumed a fixed background ontology which limits the space of change operations. Relaxing this assumption will introduce challenges related to aligning heterogeneous ontologies. Further, the recommendation of adding or removing concepts in a given concept hierarchy can only be a first step. Next steps will therefore also include recommendations of richer change operations.

In the scope of the SEKT project, we will apply and evaluate the presented framework and methods in the SEKT case studies.

Bibliography

[BRS92] Rodrigo A. Botafogo, Ehud Rivlin, and Ben Shneiderman. Structural analysis of hypertexts: identifying hierarchies and useful metrics. *ACM Trans. Inf. Syst.*, 10(2):142–180, 1992.

[BS97] M. Balabanović and Y. Shoham. Fab - content-based, collaborative recommendation. *CACM*, 40(3):66–72, 1997.

[Bur02] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User Adapted Interaction*, 12/4:331–370, 2002.

[DK04] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Sys.*, 22(1):143–177, 2004.

[EHS04] Marc Ehrig, Peter Haase, and Nenad Stojanovic. Similarity for ontologies - a comprehensive framework. In *Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004*, DEC 2004.

[GNOT92] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *CACM*, 35(12):61–70, 1992.

[Gua98] N. Guarino. Formal ontology and information systems. volume 46 of *Frontiers in Artificial Intelligence and Applications*, Trento, Italy, 1998. IOS-Press.

[HBE+04] Peter Haase, Jeen Broekstra, Marc Ehrig, Maarten Menken, Peter Mika, Michal Plechawski, Pawel Pyszlak, Björn Schnizler, Ronny Siebes, Steffen Staab, and Christoph Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004*, NOV 2004.

[HKTR04] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Sys.*, 22(1):5–53, 2004.

[HPS04] I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. *Journal of Web Semantics*, 1(4), 2004.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.

[Kig84] J. I. Kiger. The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Mach. Stud.*, 20(2):201–213, 1984.

[KN03] M. Klein and N. Noy. A component-based framework for ontology evolution. In *Proc. of the WS on Ont. and Distr. Sys., IJCAI '03*, Acapulco, Mexico, August9, 2003.

[MSR04] S. Middleton, N. Shadbolt, and D. De Roure. Ontological user profiling in recommender systems. *ACM Trans. on Inf. Systems*, 22:54–88, 2004.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of the Conf. on Comp. Sup. Coop. Work (CSCW'94)*, pages 175–186, Chapel Hill NC, 1994. Addison-Wesley.

[Sal71] G. Salton. Relevance feedback and the optimization of retrieval effectiveness. In G. Salton, editor, *The SMART system — experiments in automatic document processing*, pages 324–336. Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.

[SF91] P. D. Stotts and R. Furuta. Dynamic adaptation of hypertext structure. In *Hypertext'91 Proc., San Antonio, TX, USA*, pages 219–231. ACM, 1991.

[SHG03] N. Stojanovic, J. Hartmann, and J. Gonzalez. Ontomanager - a system for usage-based ontology management. In *In Proc. of FGML Workshop. SIG of German Information Society (FGML - Fachgruppe Maschinelles Lernen GI e.V.)*, 2003.

[SM95] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "'word of mouth"'. In *Proc. of the SIGCHI conf. on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[SMMS02] L. Stojanovic, A. Mädche, B. Motik, and N. Stojanovic. User-driven ontology evolution management. In *European Conf. Knowledge Eng. and Management (EKAW 2002)*, pages 285–300. Springer-Verlag, 2002.

[SS02] N. Stojanovic and L. Stojanovic. Usage-oriented evolution of ontology-based knowledge management systems. In *Int. Conf. on Ontologies, Databases and Applications of Semantics, (ODBASE 2002), Irvine, CA*, LNCS, pages 230–242, 2002.

[SSGS03] L. Stojanovic, N. Stojanovic, J. Gonzalez, and R. Studer. Ontomanager - a system for the usage-based ontology management. In *ODBASE 2003*, volume 2888, pages 858–875. Springer, DEC 2003.

[Sto04] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, University of Karlsruhe, 2004.

[ZSTL04] C. Ziegler, L. Schmidt-Thieme, and G. Lausen. Exploiting semantic product descriptions for recommender systems. In *Proc. 2nd ACM SIGIR Sem. Web and IR WS (SWIR '04), July 25-29, 2004, Sheff., UK*, 2004.