Collaborative and Usage-Driven Evolution of Personal Ontologies

Peter Haase[1], Andreas Hotho[2], Lars Schmidt-Thieme[3], and York Sure[1]

[1] Institute AIFB, U of Karlsruhe, Germany
{haase,sure}@aifb.uni-karlsruhe.de
[2] Knowledge Discovery Engineering Group, U of Kassel, Germany
hotho@cs.uni-kassel.de
[3] Computer-based New Media Group,
Institute for Computer Science, U of Freiburg, Germany
lst@informatik.uni-freiburg.de

Abstract. 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 classification 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. Of particular interest for supporting users during the personalization are collaborative filtering systems which can produce personal recommendations by computing the similarity between own preferences and the one of other people. In this paper 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. Such a system has been implemented in the context of Bibster, a peer-to-peer based personal bibliography management tool. Finally, we report on an experiment with the Bibster community that shows the performance improvements over non-personalized recommendations.

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

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 486–499, 2005.

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 in this paper 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 paper is structured as follows. In the next Section 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 3 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. We will introduce the Bibster applications and its extensions with the recommender functionality in Section 5 followed by evaluation results in Section 6. The evaluation was performed as an experiment within the Bibster community and shows the performance improvements over non-personalized recommendations. Finally, we conclude in Section 7.

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 [15], 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 [20], 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 [5, 14], [16] and [1], respectively, and are exemplified by the three systems presented there, MovieLens, Ringo, and fab. (3) *Hybrid* recommender systems try to combine both approaches [1, 2]. Although most recommender systems research meanwhile focuses on more complex models treating the task as a learning or classification problem, collaborative filtering models still are under active investigation [8, 3] due to their simplicity and comparable fair quality.

Taxonomies are used in recommender systems to improve recommendation quality for items, e.g., in [13] and [21]. 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 [12] and [17]. In [17] 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 [19], where the user's behavior during the knowledge providing and searching phase is analyzed. [18] 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.

3 Ontology Model and Ontology Change Operations

3.1 Ontology Model

As the OWL ontology language has been standardized by the W3C consortium, we will adhere to the underlying OWL ontology model. Because of their computational characteristics, the sublanguages OWL-DL and OWL-Lite are of particular importance. These languages are syntactic variants of the $\mathcal{SHOIN}(\mathbf{D})$ and $\mathcal{SHIF}(\mathbf{D})$ description logics, respectively [9]. 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:

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

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 $\mathcal{O}$.

3.2 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 $\text{oco}_{\text{complex}} = \text{oco}_n \circ ... \circ \text{oco}_1 = \text{oco}_n(...\text{oco}_1))$.

3.3 Ontology Rating Annotations

Our ontology model so far describes the actual state of an ontology for a user. Once we enter the more dynamic scenario of ontology evolution, it makes sense that a user (i) can express more 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 not part of his ontology. In the context of software configuration management, the latter is known as specifying a "taboo".

We model this importance information by a rating annotation.

[1] For the direct model-theoretic semantics of $\mathcal{SHOIN}(\mathbf{D})$ we refer the reader to [10].

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 $r : N \cup \mathcal{A} \to \mathbb{R}$.*

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. High values denote the relative importance of a symbol or axiom, negative values that it is unwanted by the user.

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 it has 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.

3.4 Ontology Aligmnent

An additional problem that we have to face is: If two ontologies talk about a name s, does this name refer to the same entity? Generally, this will not be the case and we will have to establish mappings between the symbol names of each pair of ontologies. This problem is known as *ontology alignment* or *ontology mapping* in the literature.

As in most applications, individuals eventually may have global IDs – e.g., URIs for web pages, ISBNs for books, etc. – concepts and relations typically have not. But although we think that the ontology alignment task is a crucial requirement for recommending ontology changes, for the sake of simplicity we will not pursue this problem here any further and rather refer the reader to e.g. [11]. In the following we assume that all symbols are global identifiers.

4 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 map

$$\varrho : \mathcal{X} \to \mathcal{P}(\mathrm{OCO}) \tag{1}$$

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.

Let

$$\mathrm{sim} : \mathcal{O} \times \mathcal{O} \to \mathbb{R} \tag{2}$$

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 [4].

Recall that ontologies in our context may have additional rating annotations that are valueable 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:

$$\mathrm{sim}_r(O, P) := \frac{\sum_{s \in N \cup \mathcal{A}} r_O(s)\, r_P(s)}{\sqrt{\sum_{s \in N \cup \mathcal{A}} r_O(s)^2} \sqrt{\sum_{s \in N \cup \mathcal{A}} r_P(s)^2}} \tag{3}$$

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

$$\mathrm{sim}(O, P) := \frac{1}{2}\,\mathrm{sim}_{r^m}(O, P) + \frac{1}{2}\,\mathrm{sim}_{r^u}(O, P) \tag{4}$$

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:

$$r_{\textit{personalized}}(O, \Omega, c) := \frac{\sum_{P \in \Omega} \mathrm{sim}(O, P)\, r^m_P(c)}{\sum_{P \in \Omega} |\,\mathrm{sim}(O, P)|} \tag{5}$$

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. 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 unweighted average of the membership ratings:

$$r_{baseline}(O, \Omega, c) = \frac{\sum_{P \in \Omega} r_P^m(c)}{|\Omega|} \tag{6}$$

5 Case Study: Bibster

In this section we will first introduce the Bibster system [7] 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.

5.1 Application Scenario: Sharing Bibliographic Metadata with Bibster

Bibster[2] 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.

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 [6]. 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[3] 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[4].

In our scenario we use the ACM Topic Hierarchy[5] 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 query-

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

ing and browsing. The classification can be done automatically by the application or manually (by drag and drop).

5.2 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 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 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.

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

Fig. 1. Visualization of topics in evolution mode

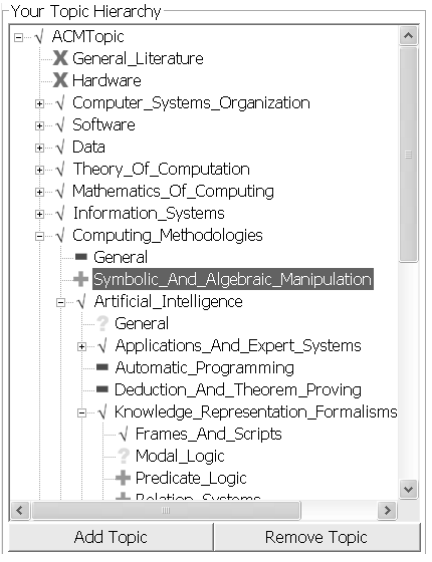

Fig. 2. Screenshot

We obtain the membership-rating r^m in the evolution mode from the explicit user actions (c.f. Figure 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 [7].

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 1. Figure 2 shows a screenshot of the ontology in the evolution mode.

6 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.

6.1 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.

6.2 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 $\mathcal{E} \subseteq \text{OCO} \times \mathcal{P}(\text{OCO})$.

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

$$\mathit{recall}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid e\in\hat{E}\}|}{|\mathcal{E}|} \tag{7}$$

$$\mathit{error}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid \mathrm{opp}(e)\in\hat{E}\}|}{|\mathcal{E}|} \tag{8}$$

$$\mathit{restraint}(\mathcal{E}) := \frac{|\{(e,\hat{E})\in\mathcal{E} \mid \mathrm{opp}(e)\notin\hat{E}\wedge e\notin\hat{E}\}|}{|\mathcal{E}|} \tag{9}$$

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 1:

$$\mathit{profit}(\mathcal{E}) := \frac{\sum_{(e,\hat{E})\in\mathcal{E}}\sum_{\hat{e}\in\hat{E}}\mathit{profit}(e,\hat{e})}{|\mathcal{E}|} = \mathit{recall}(\mathcal{E}) - \mathit{error}(\mathcal{E}) \tag{10}$$

Table 1. Evaluation Profit Matrix

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

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.

6.3 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 of these action were performed by users with the baseline strategy, 334 by users with the personalized recommender. Table 2 shows the number of add-topic-actions for the most popular topics. Figure 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 3 (upper right), overall the personalized recommender correctly recommended more than 55% of the user actions, while the baseline achieved less

Table 2. Most Popular Topics

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

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

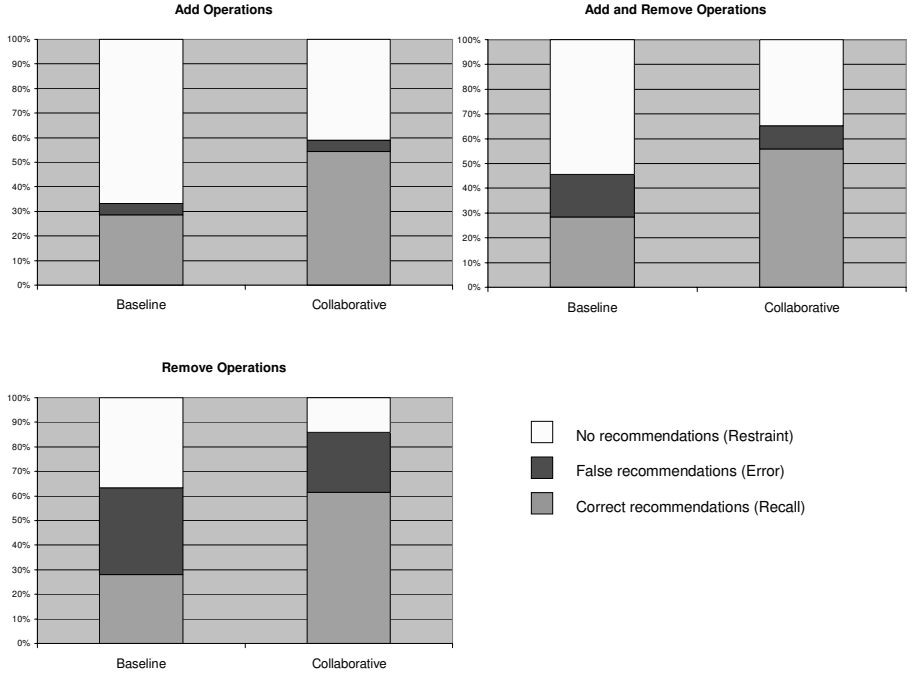

Fig. 3. Performance measures of the recommender

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 higher amount of *error* recommendations for remove operations in comparison to the a really small amount of it for the add recommendations while the correct recommendations are comparable for both operations (cf. Figure 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 $\mathit{profit}(\mathcal{E}) = 0.11$ for the baseline recommender. For the collaborative recommender, we obtain a significantly better value of $\mathit{profit}(\mathcal{E}) = 0.47$. Concluding we can state that the personalized recommender function provides substantially more useful recommendations.

7 Conclusion and Future Work

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.

Acknowledgments

Research reported in this paper has been partially financed by the EU in the IST project SEKT (IST-2003-506826) (`http://www.sekt-project.com`). We would like to thank our colleagues for fruitful discussions.

References

1. M. Balabanović and Y. Shoham. Fab - content-based, collaborative recommendation. *CACM*, 40(3):66–72, 1997.
2. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User Adapted Interaction*, 12/4:331–370, 2002.

3. M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Sys.*, 22(1):143–177, 2004.
4. M. Ehrig, P. Haase, and N. Stojanovic. Similarity for ontologies - a comprehensive framework. In *Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM 2004*, DEC 2004.
5. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *CACM*, 35(12):61–70, 1992.
6. N. Guarino. Formal ontology and information systems. In N. Guarino, editor, *Proc. 1st Int. Conf. on Formal Ont. in Inf. Sys. (FOIS)*, volume 46 of *Frontiers in AI and App.*, Trento, Italy, 1998. IOS-Press.
7. P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004*, NOV 2004.
8. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Sys.*, 22(1):5–53, 2004.
9. I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. *Journal of Web Semantics*, 1(4), 2004.
10. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Very Expressive Description Logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.
11. Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1):1–31, 2003.
12. 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, Aug.9, 2003.
13. S. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. on Inf. Systems*, 22:54–88, 2004.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. P. D. Stotts and R. Furuta. Dynamic adaptation of hypertext structure. In *Hypertext'91 Proc., San Antonio, TX, USA*, pages 219–231. ACM, 1991.
21. 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.