Application of Ontology Techniques to View-Based Semantic Search and Browsing

Eero Hyvönen, Samppa Saarela, and Kim Viljanen

Helsinki Institute for Information Technology (HIIT) / University of Helsinki
P.O. Box 26, 00014 UNIV. OF HELSINKI, FINLAND
{Eero.Hyvonen, Samppa.Saarela, Kim.Viljanen}@cs.Helsinki.FI
http://www.cs.helsinki.fi/group/seco/

Abstract. We show how the benefits of the view-based search method, developed within the information retrieval community, can be extended with ontology-based search, developed within the Semantic Web community, and with semantic recommendations. As a proof of the concept, we have implemented an ontology- and view-based search engine and recommendation system Ontogator for RDF(S) repositories. Ontogator is innovative in two ways. Firstly, the RDFS-based ontologies used for annotating metadata are used in the user interface to facilitate view-based information retrieval. The views provide the user with an overview of the repository contents and a vocabulary for expressing search queries. Secondly, a semantic browsing function is provided by a recommender system. This system enriches instance level metadata by ontologies and provides the user with links to semantically related relevant resources. The semantic linkage is specified in terms of logical rules. To illustrate and discuss the ideas, a deployed application of Ontogator to a photo repository of the Helsinki University Museum is presented.

1 Introduction

This paper addresses two problems encountered when using keyword search. Firstly, the precision and recall of keyword-based search methods is lowered since they are based on words instead of the underlying concepts [4]. For example, a keyword in a document does not necessarily mean that the document is relevant, relevant documents may not contain the explicit keyword, synonyms lower recall rate, homonyms lower precision rate, and semantic relations such as hyponymy, meronymy, and antonymy [6] are not taken into account. A prominent solution approach to these problems is to use *ontology-based information retrieval* [21, 22]. Secondly, keyword search methods are not easy to use in situations where the user does not know the terminology used in annotating the contents or does not have an explicit target to be retrieved in mind but rather wants to learn what the database in general contains. A prominent solution approach to information retrieval in this kind of situations is the *multi-faceted* or *view-based search* method[1] [19, 9]. Here the idea is to organize the terminological keywords of the underlying database into orthogonal hierarchies and use them extensively in the

[1] See http://www.view-based-systems.com/history.asp for a historical review of the idea.

user interface in helping the user to formulate the queries, in navigating the database, and in grouping the results semantically.

We describe a system called Ontogator, a semantic search engine and browser for RDF(S)[2] repositories. Its main novelty lays in the idea of combining the benefits of ontology-based and view-based search methods with semantic recommendations. To test and validate the ideas presented, Ontogator has been applied to a real-life system Promoottori that is in daily use at the Helsinki University Museum[3].

In the following, keyword, view-based and ontology-based approaches to information retrieval are first discussed. After this Ontogator and its application to Promoottori are discussed. In conclusion, contributions of this paper are summarized, related work discussed, the lessons learned listed, and a further application of Ontogator to a deployed semantic web portal is pointed out.

2 View-Based Search

The content of data records in a database is often described by associating each record with a set of keywords. Keywords can be selected from controlled vocabularies or thesauri [7] in order to create coherent annotations and to ease image retrieval. In view-based information retrieval, the keywords—to be called *categories*—are organized systematically into a set of hierarchical, orthogonal taxonomies, such as "Artifacts", "Places", "Materials" etc. The taxonomies are called subject *facets* or *views*.

A search query in view-based search is formulated by selecting categories of interest from the different facets. For example, by selecting the category "Floora's day" from a time facet, and "Building" from a location facet, the user can express the query for retrieving all images that are taken during Floora's day *and* at *any* building that is defined as a subcategory of "Building" (at any depth), such as "Old Student Union house". Intuitively, the query is a conjunctive constraint over the facets with disjunctive constraints over the sub-categories in each facet.

More formally, if the categories selected are $C_1,...,C_n$ and the subcategories of $C_i, i = 1...n$, including C_i itself are $S_{i,1}, S_{i,2}...,S_{i,k}$, respectively, then this selection corresponds to the following boolean AND-OR-constraint:

$$(S_{1,1} \vee ... \vee S_{1,k}) \wedge (S_{2,1} \vee ... \vee S_{2,l}) \wedge ... \wedge (S_{n,1} \vee ... \vee S_{n,m}) \quad (1)$$

Facets can be used for helping the user in information retrieval in many ways. Firstly, the facet hierarchies give the user an overview of what kind of information there is in the repository. Secondly, the hierarchies can guide the user in formulating the query in terms of appropriate keywords. Thirdly, the hierarchies can be used to disambiguate homonymous query terms. Fourthly, the facets can be used as a navigational aid when browsing the database content [9]. Fifthly, the number of hits in every category that can be selected next can be computed *beforehand* and be shown to the user [19]. In this way, the user can be hindered from making a selection leading to an empty result set—a recurring problem in IR systems—and is guided toward selections that are likely to constrain (or relax) the search appropriately.

[2] http://www.w3.org/RDF

[3] http://www.helsinki.fi/museo/

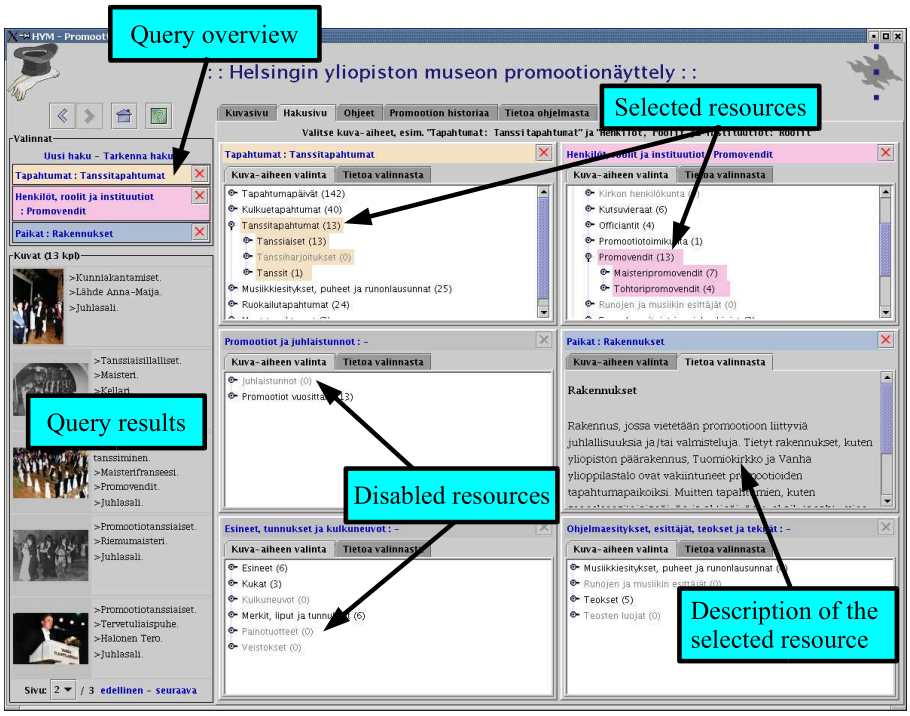

Fig. 1. Ontogator user interface for view-based multi-facet search in Promoottori.

Figure 1 shows the search interface of Ontogator in the Promoottori application. It is used by museum clients for finding photographs related to the historical promotion ceremonies of the University of Helsinki. The contents of the photos are semantically complicated and the vocabulary used in describing the ceremonies largely unknown to the users. The goal of Promoottori is to provide the museum guest with an easy to use image information retrieval system for investigating the contents of the promotion photo database, and in this way to illustrate the inner life and traditions of the university.

On the right, six facet hierarchies are shown (in Finnish): “Events”, “Promotions”, “Performances”, “Persons and roles”, “Physical objects”, and “Places”. For example, the Events facet (Tapahtumat) classifies the various traditional events that take place during the ceremonies. It gives the user an overview of the whole ceremony process and the vocabulary for formulating queries. The facet hierarchies are visualized like hierarchical folders in Windows Explorer. By clicking on the symbol in front of the category name, the category is expanded into sub-categories.

A query is formulated by selecting (sub-)categories in hierarchies, at most one selection from each facet. A category is selected into the query by clicking on its name. When the user selects a new category c, the system constrains the search further by leaving in the current result set only such images that are annotated with some sub-category of c. After each selection the result set is recomputed for each category in the opened

hierarchies, and a number n is shown to the user. It tells that if the category is selected next, then there will be n images in the result set. A selection leading to empty result set ($n = 0$) is disabled and shown in gray color. The query can be relaxed by making a new selection on a higher level of the facets or by dismissing the facet totally from the query.

The idea of view-based search idea has been used, e.g., in the HiBrowse system [19] in the 90's. A later application of the approach is the Flamenco system [9] and the first web-based prototype of Ontogator [11]. Extensive user studies [16, 5] have recently been carried out to show that a direct Google-like keyword search interface is preferred over view-based search if the users know precisely what they want. However, if this is not the case, then the view-based search method with its "browsing the shelves" sensation is clearly preferred over keyword search or using only a single facet. The latter approach is commonly used for finding resources on the web, e.g., in the Open Directory Project[4] and in Yahoo.

3 Extending Views with Ontologies

View-based search is based on hierarchically organized category labels. They are related with each other by the hierarchical inclusion relation within a single classification. By using semantically richer ontologies the following benefits can be obtained. Firstly, ontologies can be used to describe the domain knowledge and the terminology of the application in more detail. For example, relations between categories in different views can be defined. Secondly, ontologies can be used for creating semantically more accurate annotations [21, 22] in terms of the domain knowledge. Thirdly, with the help of ontologies, the user can express the queries more precisely and unambiguously, which leads to better precision and recall rates. Fourthly, through ontological class definitions and inference mechanisms, such as property inheritance, instance-level metadata can be enriched semantically.

Ontogator combines the benefits of view-based and ontology-based search methods. It provides the user with a view-based interface by which he can easily get an overview of the database contents, learn the terminology in use, and formulate the queries (cf. figure 1). However, the categories in the views are not keyword hierarchies but projected resources from the underlying ontologies by which the contents have been annotated. The domain knowledge, annotations and information retrieval is based on semantically rich ontological structures instead of simple keyword classifications. As a result, more developed inference mechanisms can be employed for performing the search and for creating additional services. In Ontogator, for example, a semantic recommendation system has been implemented for browsing the data resources.

Figure 2 depicts the overall architecture of Ontogator. The system is used by the Content Browser and is based on two information sources: Domain Knowledge and Annotation Data. Domain Knowledge consists of ontologies that define the domain concepts and the individuals. Annotation Data describes the metadata of the data resources represented in terms of the annotation and domain ontologies. The subject of a

[4] http://dmoz.org

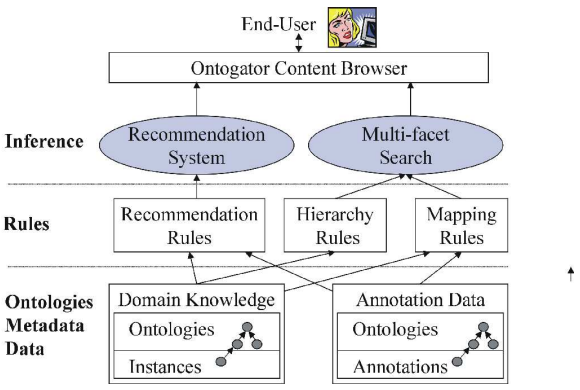

Fig. 2. Architecture of Ontogator.

data resource (image in Promoottori) is described by associating it with a set of RDF(S) resources of the domain knowledge that describe its content (in Promoottori, these resources occur in the image). The difference with keyword annotations is that the associated resources are not words but URIs of the domain ontology knowledge base, which disambiguates the meaning of annotations (synonym/homonym problem) and provides additional implicit semantic metadata through the RDF graph. The domain and annotation ontologies used in the Promoottori application are described in more detail in [13].

Based on the domain knowledge and the annotation data, Ontogator provides the user with two services:

Multi-facet search The underlying domain ontologies are projected into facets that facilitate multi-facet search.

Recommendation system After finding an image of interest by multi-facet search, Domain Knowledge and Annotation Data are used to recommend the user to view other related data resources shown as hypertext links. The labels of the links are used to explain the semantic relation to the user. For example, in Promoottori links to photos of the relatives of a person in a photo are recommended.

These two services are connected with the information sources by three sets of rules: Hierarchy Rules, Mapping Rules, and Recommendation Rules (cf. figure 2).

3.1 Hierarchy Rules

The hierarchy rules tell how to construct the facet hierarchies from the domain ontologies. Hierarchy rules are needed in order to make the classifications shown to the user independent from the design choices of the underlying Domain Ontologies. The view-based search system itself does not differentiate between differently projected hierarchies.

The specification of a facet hierarchy consists of the following parts: 1) Selection of the top resource (root) for the facet in a domain ontology. 2) Specification of the relation through which the (sub)categories are found from the root in the domain ontology. In Ontogator and Promoottori, as described in this paper, hierarchy projections are created using Java but in the recent server version on Ontogator, hierarchy projections are specified in logic and Prolog is used as in [15].

An obvious way to extract a facet hierarchy from the RDF(S)-based domain knowledge is to use the subclass-of hyponymy relation. Then the inner nodes of the hierarchy consist of the classes of the domain ontology, and the leaves are the direct instances of these classes. Using only hyponymy for facet projections would, however, be a limitation in the general case. For example, places may constitute a part-of hierarchy, and this would be a natural choice for a facet in the user interface.

Hierarchy rules tell how the views are projected logically. A separate question is how these hierarchies should be shown to the user. Firstly, the ordering of the sub-resources may be relevant. In Promoottori, for example, the sub-happenings of an event should be presented in the order in which they take place and persons be listed in alphabetical order. In Ontogator, ordering of the sub-nodes can be specified by a configurable property; the sub-categories are sorted based on the values of this property. Second, one may need a way to filter unnecessary resources away from the user interface. For example, in Promoottori the ontology was created partly before the actual annotation work and had more classes and details than were actually needed. In Ontogator, empty categories can be pruned out. A hierarchy may also have intermediate classes that are useful for knowledge representation purposes but are not very natural categories to the user. Such categories should be present internally in the search hierarchies but should not be shown to the user. Third, the names for categories need to be specified. For example, in Promoottori the label for a person category should be constructed from the last and first names represented by distinct property values.

3.2 Mapping Rules

In Promoottori, an image is annotated by associating it with a set of domain knowledge resources describing its content. This set is, however, not enough because there may be also *indirect* relations between images and the annotations. Mapping rules can be used to specify what indirect resources describe the images in addition to the direct ones. Through such rules it is possible to achieve a search system that is independent of both the annotation scheme and the domain ontology design. The search system does not make any distinction between the ways in which data resources may be related with their annotations.

For example, in Promoottori there are the classes Role and Person in the domain ontology. The subclasses of Role, such as Master and Doctor Honoris Causa, are used to annotate the role in which a person appears in a picture. If the role r of a person p is known, then the picture is annotated by an instance of the Role. As a result, the picture is found using r in the multi-facet search through the "Roles" facet, but not with p through the "Persons" facet, which would be unsatisfactory. The problem can be solved by using a mapping rule telling that the images, that are about an instance of Role are also images about the person (in that role).

Mapping rules are given as RDF traversal paths using the N3 notation[5]. For example, the description below tells that any instance *p* of the class Person (or its subclasses) is mapped to an image, if there is a `related_instances` arc from the image to resource *r* and then a `persons_in_role` arc from there to p (notation ^ denotes traversal through an RDF arc in the opposite direction). The tag `labelRegex` illustrates how category labels are constructed in Ontogator from different resources using templates. The label for the person categories is concatenated from the property values of `lastName` and `firstName`, separated by a comma and space (e.g., "Smith, John").

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
 <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
 <!ENTITY Promotion
        'http://www.cs.helsinki.fi/Promotion#'>
 <!ENTITY yom 'http://www.cs.helsinki.fi/yomuseo/yom#'>
 <!ENTITY rdfs 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>
]>
<rdf:RDF xmlns:yom="&yom;" xmlns:rdf="&rdf;"
         xmlns:Promotion="&Promotion;" xmlns:rdfs="&rdfs;">
...
<rdf:Description rdf:about="&Promotion;Person">
  <yom:relatedDocumentMapping>
    ^Promotion:persons_in_role
    ^Promotion:related_instances
  </yom:relatedDocumentMapping>
  <yom:labelRegex>${Promotion:lastName}, ${Promotion:firstName}
  </yom:labelRegex>
</rdf:Description>
...
</rdf:RDF>
```

All mappings between facet resources and the data resources are determined when constructing the system's internal representation of the facet hierarchies. Computing mappings during the startup makes the search system faster but at the price of the memory needed for the search data structures.

4 Recommendation System

Ontogator's Recommendation System (figure 2) is a mechanism for defining and finding semantic relations in the underlying RDF(S) knowledge base, i.e. Domain Knowledge and Annotation Data. The relations of interest to the user in an application are described by a set of logical Recommendation Rules. The recommendations are shown as labeled navigational links relating data resources with each other. Figure 3 illustrates their usage in Promoottori. The user has selected a photo from the Query results provided by the multi-facet search engine on the left. Ontogator shows the image with its metadata in the middle. On the right, the Recommendations for the selected image are seen in groups. Each group is based on a recommendation rule. A group may contain several images that can be viewed one after another in the group pane. The title of the group pane gives a general, rule level explanation for the semantic recommendation, such as

[5] http://www.w3.org/DesignIssues/Notation3.html

"next event" (seuraava tapahtuma). In addition, every recommended photo is described and (if possible) given a natural language explanation of how the current image is related to the recommended one. For example, if the current image presents Ms. Laura Hautamäki, then the system recommends images containing images about her father with an explanation "The father of Laura Hautamäki" under the group title "Related persons".

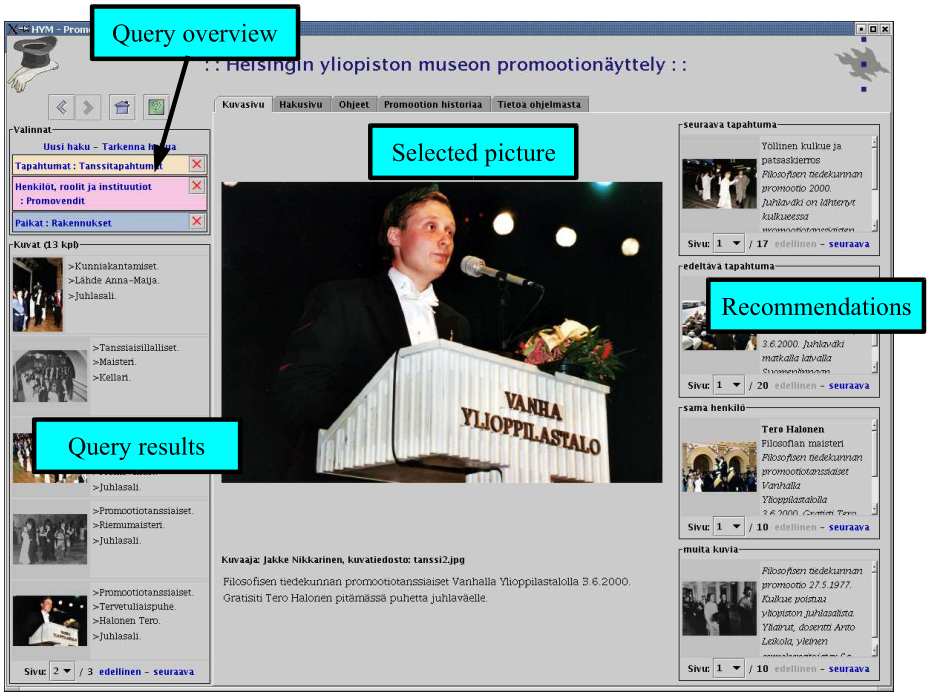

Fig. 3. Screenshot of the recommendation system in Promoottori.

Promoottori currently contains rules for determining photos of the preceding event and the next event based on the promotion procedure, rules for determining photos depicting related persons (family relations), rules for showing other images of the persons in the selected image, rules for showing photos taken in the same place, rules for showing images from the same promotion, and rules for showing photos from the same era.

4.1 Recommendation Rules

The recommendation rules are defined in terms of logical Horn clauses and are implemented using SWI-Prolog[6] and its RDF parser. For example, the "Related persons"

[6] SWI-Prolog version 5.1.5, http://www.swi-prolog.org/

-rule links a person with other persons through family relations described in the underlying RDF graph. If the user selects an image exposing a person *p*, then images exposing persons in different family relations with *p* are recommended to the user. The following predicate is used for defining the recommendation rule:

```
rec_related_persons(X,Y,RecType,Relation,Desc,RevDesc,Priority)        (2)
```

Here X is the selected resource that we are searching links for, Y is the recommended resource, RecType is the topic title for this recommendation rule to be shown in the group pane in the user interface, Relation is the property (URI) which connects X and Y (if exists), Desc is a natural language description for the relation X $\rightarrow$ Y (e.g., "X is the father of Y"), RevDesc a reverse description for the relation Y $\rightarrow$ X (e.g., "Y is a child of X"), and Relevance is an integer that is used to sort the recommendations in a relevance order in the user interface. In essence, the rule defines 1) a set of RDF triples (recommendation links) between the resources and 2) attaches with each triple additional information to be used in showing the recommendation to the end-user. The same predicate structure is used for all recommendation rules. The definition of the predicates can be programmed freely in Prolog and depends on the ontologies and metadata of the application.

The Promotion ontology contains a class Persons that has the properties father-of, mother-of, and spouse. The related persons may be found by simply following the RDF triplet connecting two person instance resources. The central part in defining the predicate `rec_related_persons` is to specify when two persons are related with each other by some of these properties. In this case, the simple definition below can be used:

```
related_persons(X, Y, Relation) :-
   rdf(X, rdf:type, 'http://www.cs.helsinki.fi/Promotion#Persons'),
   rdf(X, Relation, Y),
   rdf(Y, rdf:type, 'http://www.cs.helsinki.fi/Promotion#Persons').
```

Here the predicate `rdf` matches RDF triples and is provided by the SWI-Prolog RDF parser. If needed, other rules could be defined for finding more complicated family relations, such as grand children or cousins.

The `rec_related_persons` predicate also has to create for each recommendation triple the explanatory strings RecType, Expl, and RevExpl to be used in the user interface visualization, and a value for the relevance. These values can either be given explicitly or computed, e.g., by concatenating the labels of the RDF graph.

For reasons of efficiency, the recommendations are generated into an XML-file in a batch process before using the application. This recommendation file is loaded into the Ontogator browser at the application startup. A limitation of this static approach is that it is not possible to create on-line dynamic recommendations based on the user's profile and usage of the system. (An online-version of the relation generation engine has been implemented in [10].)

Two recommendations for a photo resource are shown in XML below. The first recommendation (`rec` tag) is based on the logical relation "same century". The URI of the related resource (`relatedInstance`) is given (here "..." for short) with the label "Promotions in the 19th century". The tag and attribute names used correspond to the

variable names in predicate (2). For example, the priority attribute of the recommendation tells the relevance order in which the recommendations are shown to the user.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<recommendations>
...
<recs about=
  "http://www.cs.helsinki.fi/Promotion#Promotion_02311">
  <rec priority="3">
    <relatedInstance uri="...">Promotions in the 19th century
    </relatedInstance>
      <relation name="same century"/>
      <rectype>same century</rectype>
     <description>Promotions in the 19th century</description>
  </rec>
  <rec priority="8">
      <relatedInstance uri="...">Theological Faculty</relatedInstance>
      <relation name="same promotion"/>
      <rectype>same promotion</rectype>
      <description>Theological Faculty, June 15, 1818</description>
  </rec>
  ...
</recs>
...
</recommendations>
```

The recommendation system is divided into three modules: domain specific recommendation rules, RDF Schema specific rules (such as rules implementing the transitive subclass-of closure), and system specific rules (the "main" program creating the recommendations). The domain specific rules are application dependent and are created by a domain specialist. RDF Schema specific and system specific rules are independent of the application.

When processing the data, the program iterates through all images and their metadata. The recommendation rules are applied to every different resource *r* describing the content of some image. If recommendations are found, they are stored as recommendations for *r*. In order to minimize the number of recommendation links represented in the XML-file the recommendations are created only for each metadata resource *r* and not for each image. The Ontogator browser then shows, as the recommendations of an image, the recommendations related to each resource *r* used in the image's metadata.

The recommendation system creates recommendation descriptions to the user in natural language using templates such as "*Person X* is a child of *Person Y*". The definition of the reverse description (RevDesc in predicate (2)) facilitates symmetric usage of the recommendation associations. The texts are based on the labels of the resources defined in the RDF descriptions and some simple Prolog rules describing typical Finnish conjugation rules. For example, the genetive form for the last name "Virtanen" is "Virtasen" but "Mannerheim" is in genetive form "Mannerheimin".

4.2 Strategies for Creating Recommendations

Recommendations can be created in various ways [20]. In our work, we have been considering the following alternatives:

User *profile-based recommendations* are based on information collected by observing the user, or in some cases by asking the user to explicitly define the interest profile. Based on the user's profile, recommendations are then made to the user either by comparing the user's profile to other users' profiles (collaborative filtering/recommending) or by comparing the user's profile to the underlying document collection (content-based recommending). The strength of user profile-based recommendations is that they are personalized and hence serving better the user's individual goals. In our case application, personalization is however difficult, because the users cannot be identified. It is not even known when the user's session begins and when it ends because the users are using the same physical kiosk interface located in the museum. The profiling must be easy for the user because most of the users use the system perhaps only once in their lifetime. Finally, it is difficult to identify whether the user liked or disliked the current image without asking the user to rate every image explicitly. A weakness of collaborative filtering is that explaining the recommendations to the user can be difficult, because they are mostly based on heuristic measures of the similarity between user profiles and database contents, and on the user's actions.

With *similarity-based recommendations* we refer to the possibility to compare the semantical distance between the metadata of resources. The nearest resources are likely to be of more interest and could be recommended to the user. A difficulty of this recommendation method is how to measure the semantical distance between metadata. For example, in Promoottori the most similar image may not the most interesting one but rather just another picture of the same event. One method is to use the count of common or intersecting annotation resources as a distance measure [23].

The idea of *rule-based recommendations* used in Ontogator is that the domain specialist explicitly describes the notion of "interesting related image" with generic logic rules. The system then applies the rules to the underlying knowledge base in order to find interesting images related to the selected one. This method has several strengths. Firstly, the rule can be associated with a label, such as "Images of the previous event", that can be used as the explanation for the recommendations found. It is also possible to deduce the explanation label as a side effect of applying the rule. Recommendation rules are described by the domain specialist. The rules and explanations are explicitly defined, not based on heuristic measures, which could be difficult to understand and motivate. Secondly, the specialist knows the domain and may promote the most important relations between the images. However, this could also be a weakness if the user's goals and the specialists thoughts about what is important do not match, and the user is not interested in the recommendations. Thirdly, the rule-based recommendations do not exclude the possibility of using other recommendation methods but provides an infrastructure for running any rules. For example, the recommendation rules could perhaps be learned by observing the users actions and then used in recommending images for the current or future users.

In the initial version of Promoottori [14], we implemented a profile-based and similarity-based recommendation system that recommended semantically similar images. The recommendations were not static but were modified dynamically by maintaining a user profile and a history log of image selections. Then a rule-based recom-

mendation system was implemented due to the benefits discussed above and is in use in the Promoottori application.

5 Conclusions

5.1 Contributions

We developed methods for combining the benefits of RDF(S)-based knowledge representation, the multi-facet search method, and knowledge-based recommendations. The ideas have been implemented as the Ontogator tool. In Ontogator, facet hierarchies projected from ontologies are used to help the user in formulating the information need and the corresponding query. After finding a relevant document with view-based search, the recommender system provides the user with a semantic browsing facility linking semantically related images. The mapping between the user interface functionalities (searching and browsing) and the underlying knowledge base is specified by hierarchy, mapping, and recommendation rules. By changing the rules, Ontogator can be applied to different domains and annotation schemas. As an example, application of Ontogator to a deployed image retrieval system, Promoottori, was discussed.

5.2 Related Work

The idea of viewing an RDF(S) knowledge base along different hierarchical projections has been applied, e.g., in the ontology editor Protégé-2000[7] where it is possible to choose the property by which the hierarchy of classes is projected to the user. However, in our case a much more complex specification of the projection than a single property is needed. For example, the hyponymy projection already employs two properties (`rdfs:subClassOf` and `rdf:type`). Furthermore, in Ontogator the idea of mapping rules was developed for associating indirectly related resources with views.

The idea of semantic browsing was inspired by the idea of Topic Maps [18, 17]. However, while the links in a Topic Map are given by a map, the links in Ontogator are inferred based on logic rules and the underlying knowledge base. The idea of semantic browsing is also related to research on recommender systems [20]. In Ontogator, the recommendation system is used for searching labeled relations between data resources. This approach is different from knowledge-based recommender systems [1], such as the FindMe systems [2], where browsing is based on altering the characteristics of a found prototype. Logic and dynamic link creation on the semantic web have been discussed, e.g., in [8, 3].

The search interface of Ontogator is based on the HiBrowse model [19]. However, in our case the whole hierarchy, not only the next level of subcategories, can be opened for selections. Moving between hierarchy levels is more flexible because at any point any new selection in the opened hierarchy is possible. In addition, the "browsing the shelves" sensation is provided by a separate recommendation system based in the underlying ontological domain knowledge. This provides a semantically richer basis for

[7] http://protege.stanford.edu

browsing than the keyword hierarchies used in traditional view-based search engines, such as Flamenco [9].

The idea of ontology-based image retrieval has been discussed, e.g., in [21, 22]. By annotating data with concepts instead of words more precise information retrieval is possible. The price to be paid is that more work is needed when constructing the ontologies and during the content annotation phase.

5.3 Lessons Learned

The main difficulty in integrating the view-based and ontology-based search paradigms is how to model and deal with the indirect relations between the images and domain ontology resources, and how to project the facet hierarchies from the RDF(S) knowledge base. If not properly modeled, the precision and recall rates of the system are lowered.

A reason for choosing RDF(S) for the knowledge representation language was its domain independent nature and openness. This makes it possible to apply the content and Ontogator more easily to different applications. During our work, we actually reused the promotion ontology and instance data easily in another application for generating automatically semantically linked web pages from RDF(S) data [15].

In our work, logic programming turned to be a very flexible and effective way to handle RDF(S) data by querying and inferring when compared with RDF query languages, such as RDQL and RQL. The definition of the recommendation rules requires programming skills and may be difficult to a domain specialists who is not familiar with logic languages. A problem encountered there is how to test and verify that the recommendations for all images are feasible without having to browse through the whole database. Computational efficiency and central memory requirements can be a problem if the RDF knowledge base is very large and if the rules are complex.

During our work, Protégé-2000 was used as the ontology editor. Jena's[8] basic main memory -based model (ModelMem) was employed to load the RDF(S)-models into Ontogator's internal representation form. Protégé turned out to be a versatile tool with an intuitive user interface that even for a non-programmer could use for constructing ontologies. A good thing about Protégé is that it is not limited to RDF(S) semantics only, but enables and enforces the use of additional features.

Ontology evolution poses a problem with Protégé-2000 even in the simple case that a name (label) of some class changes. Protégé derives URI's of the classes from their names, and if a name changes then the classes URI (ID) changes also. This leads to configurational problems. Rules and mappings for one version of the ontology do not apply to the new version, even though the actual classes have not changed, only their labels. Multi-instantiations would have been desirable in some situations but this is not possible with Protégé.

The major difficulty in the ontology-based approach is the extra work needed in creating the ontology and the detailed annotations. We believe, however, that in many applications such as Promoottori this price is justified due to the better accuracy obtained in information retrieval and to the new semantic browsing facilities offered to the end-

[8] http://www.hpl.hp.com/semweb/jena.htm

user. The trade-off between annotation work and quality of information retrieval can be balanced by using less detailed ontologies and annotations, if needed.

Evaluating the quality and relevance of recommendations can only be based on the user's opinions. In our case, only a small informal user study of has been conducted using the personnel of the museum. The general conclusion was that the idea seems useful in practice.

5.4 A Further Application on the Web

A server-based version of Ontogator, Ontogator 2, has been developed and is used as the search engine of the "MuseumFinland — Finnish Museums of the Semantic Web" portal [10] that is available on the web[9]. The first version of this application contains 4,000 images of museum collection objects from three different museums. The metadata is given in terms of seven RDF(S) ontologies that consist of some 10,000 concepts and individuals. First experiments with the implementation indicate that the technology scales up to at least tens of thousands of images and ontological resources. In Ontogator 2, the recommendation system has been separated into a SWI-Prolog HTTP-server of its own called Ontodella. Ontodella is used dynamically for creating the view hierarchies (by a combination of hierarchy and mapping rules in Prolog) and for generating the recommendations for a given resource.

Acknowledgments

Kati Heinämies and Jaana Tegelberg of the Helsinki University Museum provided the content material for Promoottori. Avril Styrman created most of the promotion ontologies and annotated the images. Our work was mainly funded by the National Technology Agency Tekes, Nokia, TietoEnator, the Espoo City Museum, the Foundation of the Helsinki University Museum, the National Board of Antiquities, and the Antikvaria-group.

References

1. R. Burke. Knowledge-based recommender systems. In A. Kent, editor, *Encyclopaedia of Library and Information Sciences*. Marcel Dekker, 2000.
2. R. Burke, K. Hammond, and B. Young. The FindMe approach to assisted browsing. *IEEE Expert*, 12(4), 1997.
3. P. Dolong, N. Henze, and W. Neijdl. Logic-based open hypermedia for the semantic web. In *Proceedings of the Int. Workshop on Hypermedia and the Semantic Web, Hypertext 2003 Conference, Nottinghan, UK*, 2003.
4. D. Fensel (ed.). The semantic web and its languages. *IEEE Intelligence Systems*, Nov/Dec 2000.
5. J. English, M. Hearst, R. Sinha, K. Swearingen, and K.-P. Lee. Flexible search and navigation using faceted metadata. Technical report, University of Berkeley, School of Information Management and Systems, 2003. Submitted for publication.

[9] http://museosuomi.cs.helsinki.fi

6. C. Fellbaum, editor. *WordNet. An electronic lexical database.* The MIT Press, Cambridge, Massachusetts, 2001.
7. D. J. Foskett. Thesaurus. In *Encyclopaedia of Library and Information Science, Volume 30*, pages 416–462. Marcel Dekker, New York, 1980.
8. C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual open hypermedia = the semantic web? In *Proceedings of the WWW2001, Semantic Web Workshop, Hongkong*, 2001.
9. M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Lee. Finding the flow in web site search. *CACM*, 45(9):42–49, 2002.
10. E. Hyvönen, M. Junnila, S. Kettula, , E. Mäkelä, S. Saarela, M. Salminen, A. Syreeni, A. Valo, and K. Viljanen. MuseumFinland—Finnish Museums on the Semantic Web. User's perspective. In *Proceedings of Museums and the Web 2004 (MW2004), Arlington, Virginia, USA*, 2004. http://www.cs.helsinki.fi/u/eahyvone/publications/ MuseumFinland.pdf.
11. E. Hyvönen, S. Kettula, V. Raatikka, S. Saarela, and Kim Viljanen. Semantic interoperability on the web. Case Finnish Museums Online. In Hyvönen and Klemettinen [12], pages 41–53. http://www.hiit.fi/publications/.
12. E. Hyvönen and M. Klemettinen, editors. *Towards the semantic web and web services. Proceedings of the XML Finland 2002 conference. Helsinki, Finland*, number 2002-03 in HIIT Publications. Helsinki Institute for Information Technology (HIIT), Helsinki, Finland, 2002. http://www.hiit.fi/publications/.
13. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In Hyvönen and Klemettinen [12], pages 15–27. http://www.hiit.fi/publications/.
14. E. Hyvönen, A. Styrman, and S. Saarela. Ontology-based image retrieval. In Hyvönen and Klemettinen [12], pages 15–27. http://www.hiit.fi/publications/.
15. E. Hyvönen, A. Valo, K. Viljanen, and M. Holi. Publishing semantic web content as semantically linked HTML pages. In *Proceedings of XML Finland 2003, Kuopio, Finland*, 2003. http://www.cs.helsinki.fi/u/eahyvone/publications/ xmlfinland2003/swehg_article_xmlfi2003.pdf.
16. K.-P. Lee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of CHI 2003, April 5-10, Fort Lauderdale, USA*. Association for Computing Machinery (ACM), USA, 2003.
17. Jack Park and Sam Hunting, editors. *XML Topic Maps. Creating and using Topic Maps for the Web*. Addison-Wesley, New York, 2003.
18. Steve Pepper. The TAO of Topic Maps. In *Proceedings of XML Europe 2000, Paris, France*, 2000. http://www.ontopia.net/topicmaps/materials/rdf.html.
19. A. S. Pollitt. The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, UK, 1998. http://www.ifla.org/IV/ifla63/63polst.pdf.
20. J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1/2):115–153, 2001.
21. A. T. Schreiber, B. Dubbeldam, J. Wielemaker, and B. J. Wielinga. Ontology-based photo annotation. *IEEE Intelligent Systems*, 16:66–74, May/June 2001.
22. G. Schreiber, I. Blok, D. Carlier, W. van Gent, J. Hokstam, and U. Roos. A mini-experiment in semantic annotation. In I. Horrocks and J. Hendler, editors, *The Semantic Web – ISWC 2002. First international semantic web conference*, number 2342 in LNCS, pages 404–408. Springer–Verlag, Berlin, 2002.
23. Nenad Stojanovic, Rudi Studer, and Ljiljana Stojanovic. An approach for the ranking of query results in the semantic web. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *The Semantic Web – ISWC 2003. Second international semantic web conference*, number 2870 in LNCS, pages 500–516. Springer–Verlag, Berlin, 2003.