

# Searching Web Resources Using Ontology Mappings

Dragan Gašević and Marek Hatala

School of Interactive Arts and Technology, Simon Fraser University Surrey  
2400 Central City, 10153 King George Hwy, Surrey, BC V3T 2W1, Canada  
Email: {dgasevic, mhatala}@sfu.ca

## ABSTRACT

This paper proposes an ontology mapping based framework that allows searching for web resources using multiple ontologies. The proposed solution uses a mapping ontology that is a part of a recent Semantic Web initiative called the Simple Knowledge Organization System (SKOS). On top of that, we propose the search algorithm that takes arguments from one ontology and generates queries compliant with other ontologies. We evaluated the solution on a web application that allows using a local ontology, which describes content of a web site, to search for web resources in remote digital libraries or object repositories based on more general content ontologies.

## Categories and Subject Descriptors

H.3.3 Information Storage and Retrieval – Information Search and Retrieval

## General Terms

Algorithms, Design

## Keywords

Ontology, ontology mapping, search, interoperability

## 1. INTRODUCTION

In a past few years large collections of web resources became available either through the digital libraries (such as ACM Portal), community-based object repositories, or more importantly as widely dispersed web resources in many individual institutions. Several interoperability initiatives are trying to address the issue of searching across multiple object collections. However, the effectiveness of searching is hampered by the fact that individual web resources are typically not interconnected into the web and therefore lacking the context which makes the Google's PageRank algorithm [7] so effective. The libraries and repositories are overcoming this lack of context by providing explicit semantic information in the form of subject categories, taxonomies, or ideally richer ontologies.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission by the copyright owners.

Copyright 2005

However, one can hardly find two different object repositories relying on the same classification. Furthermore, the previous research showed that community members have real difficulty of making annotations of their objects using subject taxonomies [11]. On the other hand, they are more comfortable using their own application domain space as well as with their local context than multiple ontologies used in remote repositories.

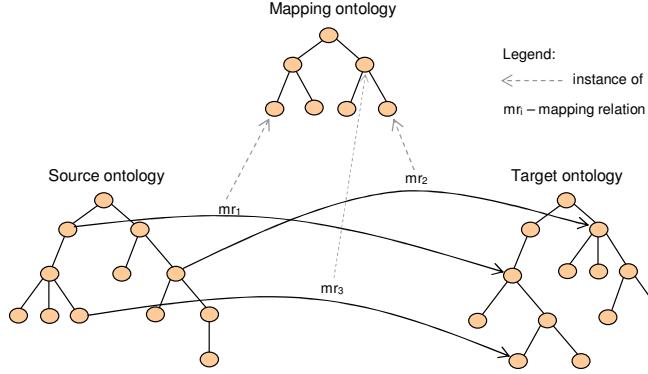
In order to address this problem here we propose the use of ontology mappings to define relations between concepts from different ontologies [9, 16, 20]. On top of such mapping relations we developed a search algorithm that uses concept of one ontology (i.e. the source ontology) as query arguments, generates queries compliant with another ontologies (i.e. target ontologies), and finally gets ranked search results semantically relevant for the source ontology. To define mapping relations among ontologies we use another ontology – mapping ontology – that specifies a set of relations for relating concepts from different ontologies. Actually, we use the Mapping Vocabulary [17] of the Simple Knowledge Organization System (SKOS), a recent W3C RDF-based initiative [18]. We implemented the search algorithm using Jess [10] and OWLJessKB [15] as a component. The component can be used in different semantic web application such as a federated search engine of object repositories/digital libraries annotated with different classifications; or applications that allow using a local web application content ontology to get relevant results from remote digital libraries based upon another ontologies.

## 2. METADATA, ONTOLOGIES, AND WEB RESOURCES

Although the present semantic web research try to improve most of interoperability issues, some problems still exist.

### 2.1 Web metadata and domain ontologies

Web resource metadata and domain ontologies (i.e. taxonomies) are often defined at different ontological levels. In order to underpin this statement let us consider an example of combining Dublin Core (DC) metadata schema [4] and domain ontologies. Technically, the DC metadata of a web resource is an instance (i.e. RDF) of the DC RDF Schema. Additionally, the metadata is enriched with keywords defined in an domain ontology (e.g. for computer science domain based on the ACM Computing Classification System – CCS [2]). If we refer to keywords



**Figure 1.** A general purpose mapping ontology as a way to define mappings among multiple ontologies

that are defined as classes in an RDF Schema, we annotate the metadata (i.e. schema instances) with ontology classes (i.e. schema). Those keywords are listed in the *subject* element of the DC metadata schema. Since ontology languages do not have a strict separation between ontological levels [6] this approach is completely applicable. In fact, this problem of representing classes as properties values has already been recognized by W3C Semantic Web Best Practices Working Group as classes as property values [19]. While OWL Full and RDF Schema do not put any restriction on using classes as property values, OWL DL and OWL Lite do not generally allow this use, and thus restrict the use of some Semantic Web reasoners. Apart from the solutions listed in the W3C note, we can use specialized ontologies for defining domain taxonomies with a rich set of properties for defining concept hierarchies such as SKOS [18].

## 2.2 Mapping among multiple domain ontologies

Currently, there are many different domain ontologies developed for the use on the Web. Very often developers are not able to reuse existing ontologies, as they were built for different purposes. For instance, some sources (e.g. object repositories, digital libraries) where we look for some web resources are based on different classifications (e.g. the ACM CCS in the ACM Digital Library [3]). We often need to build application-specific ontologies. For example, in the e-learning domain we can build an ontology of a course curriculum (e.g. Information Management course) [22] to organize web resources related to the course. However, the main issue is how to use an application-specific ontology to search for web resources annotated with another ontology. In order to overcome such diversities we have to introduce an additional level of interoperability among ontologies [13]. One solution is to employ ontology mappings to define how concepts from different ontologies relate each other.

Here we describe only one way for defining mappings, although there are many practically used ontology mapping techniques [13]. It regards the use of a mapping ontology – an ontology containing classes and properties (i.e.

primitives) that can express relations between ontology concepts and properties (see Figure 1). This principle is suitable for implementation since semantic web reasoning tools (e.g. FaCT, OWLJessKB) represent the mapping ontology in the same way (i.e. like facts) as both the source and target ontologies. Historically, this approach originates from the explicit representation of relationships between domain (i.e. source) and method (i.e. target) ontologies assembled in a specific knowledge application [21]. An example of such a mapping ontology was developed as a part of the project on reusable problem-solving components [9]. MAFRA (MAppling FRamework) is another solution for mapping distributed ontologies [16]. Apart from a very detailed mapping ontology called the semantic bridge ontology, MAFRA also defines two-dimensional process (5 horizontal and 4 vertical modules) ontology mappings process. Note also that a mapping ontology is used in the PROMPT Tab, a plug-in of the Protégé ontology editor for merging and mapping ontologies, to save discovered mappings [20]. However, none of these mapping ontologies is standardized. Furthermore, they do not possess a wide range of primitives for defining different levels of mappings (e.g. exact match), which can be useful in ranking search results.

## 3. REPRESENTATION OF ONTOLOGIES AND MAPPINGS

In order to address two problems listed in previous section our solution uses the Simple Knowledge Organization System (SKOS) [18] for defining different types of ontologies (e.g. classifications, taxonomies, thesaurus) as well as mappings of concepts between different domain ontologies. The SKOS consists of the three RDF vocabularies that are still under the active development at the W3C:

- *SKOS Core* – for expressing the basic structure and content of concept schemes (taxonomies, terminologies, etc);
- *SKOS Mapping* – for describing mappings between concept schemes;
- *SKOS Extension* – containing extensions to the SKOS Core useful for specialized applications.

<pre> &lt;rdf:RDF&gt;   &lt;skos:ConceptScheme rdf:id="acm-ccs:acm-ccs"&gt;     &lt;skos:hasTopConcept rdf:resource="#acm-ccs;A" /&gt;     &lt;!-- ... --&gt;     &lt;skos:hasTopConcept rdf:resource="#acm-ccs;K" /&gt;   &lt;/skos:ConceptScheme&gt;   &lt;!-- ... --&gt;   &lt;skos:Concept rdf:id="acm-ccs;H.3"&gt;     &lt;skos:prefLabel xml:lang="en"&gt;Information Storage and Retrieval&lt;/skos:prefLabel&gt;     &lt;skos:inScheme rdf:resource="acm-ccs:acm-ccs" /&gt;     &lt;skos:broaderGeneric rdf:resource="#acm-ccs;H" /&gt;     &lt;skos:narrowerGeneric rdf:resource="#acm-ccs;H.3.1" /&gt;     &lt;!-- ... --&gt;     &lt;skos:narrowerGeneric rdf:resource="#acm-ccs;H.3.m" /&gt;   &lt;/skos:Concept&gt;   &lt;!-- ... --&gt;   &lt;skos:Concept rdf:id="acm-ccs;H.3.3"&gt;     &lt;skos:prefLabel xml:lang="en"&gt;Information Search and Retrieval&lt;/skos:prefLabel&gt;     &lt;skos:inScheme rdf:resource="acm-ccs" /&gt;     &lt;skos:broaderGeneric rdf:resource="#acm-ccs;H.3" /&gt;     &lt;skos:narrowerGeneric rdf:resource="#acm-ccs;H.3.3.1" /&gt;     &lt;!-- ... --&gt;     &lt;skos:narrowerGeneric rdf:resource="#acm-ccs;H.3.3.6" /&gt;   &lt;/skos:Concept&gt;   &lt;!-- ... --&gt;   &lt;skos:Concept rdf:id="acm-ccs;H.3.3.1"&gt;     &lt;skos:prefLabel xml:lang="en"&gt;Information Filtering&lt;/skos:prefLabel&gt;     &lt;skos:inScheme rdf:resource="acm-ccs:acm-ccs" /&gt;     &lt;skos:broaderGeneric rdf:resource="#acm-ccs;H.3.3" /&gt;   &lt;/skos:Concept&gt;   &lt;!-- ... --&gt; &lt;/rdf:RDF&gt; </pre>	a)	<pre> &lt;!-- IM1.1 - History and motivation for information systems --&gt; &lt;skos:Concept rdf:about="#imc;IM1.1"&gt;   &lt;map:minorMatch&gt;     &lt;skos:Concept rdf:about="#acm-ccs;H.5.m" /&gt;   &lt;/map:minorMatch&gt; &lt;/skos:Concept&gt;  &lt;!-- IM1.2 - Information storage and retrieval (IS&amp;R) --&gt; &lt;skos:Concept rdf:about="#imc;IM1.2"&gt;   &lt;map:exactMatch&gt;     &lt;skos:Concept rdf:about="#acm-ccs;H.3" /&gt;   &lt;/map:exactMatch&gt; &lt;/skos:Concept&gt;  &lt;!-- IM1.6 - Search, retrieval, linking, navigation --&gt; &lt;!-- Union of H.3.3 - Information Search and Retrieval E.2.3 - Linked representations --&gt; &lt;skos:Concept rdf:about="#imc;IM1.6"&gt;   &lt;map:majorMatch&gt;     &lt;map:OR about="#OR1" /&gt;   &lt;/map:majorMatch&gt; &lt;/skos:Concept&gt; &lt;map:OR ID="#OR1"&gt;   &lt;map:memberList rdf:parseType="Collection"&gt;     &lt;skos:Concept rdf:about="#acm-ccs;H.3.3" /&gt;     &lt;skos:Concept rdf:about="#acm-ccs;E.2.3" /&gt;   &lt;/map:memberList&gt; &lt;/map:OR&gt; </pre>	b)
---	----	--	----

**Figure 2.** The use of SKOS: a) An excerpt of the ACM CCS in the XML/RDF format of the SKOS. The classification comprises the 11 top level concepts marked with letters from A to K. Most of the top level concepts are further subdivided into three more levels with numbers being added to their identifiers; b) an excerpt of the mappings between the ACM CCS and another ontology (an Information Management course curriculum ontology) defined in the SKOS Mappings

*SKOS Core* provides a model for expressing the basic structure and content of concept schemes. Basically, the SKOS Core defines a set of both RDFS properties and RDFS classes that can be used to express the content and structure of a concept scheme (such as *Concept*, *broader*, *narrower*, *related*, *subject*, *isSubjectOf*). For example, the *broader* property is used to specify that a concept is broader than another one. In order to define subclass/superclass relations we can use the *SKOS Extension* vocabulary and its properties *narrowerGeneric* and *broaderGeneric* that are sub-properties of the *narrower* and *broader* properties, respectively. The *narrowerGeneric* property is semantically equivalent to the *rdfs:subClassOf* property, and thus has a slight different meaning from the *narrower* property.

### 3.1 Ontology representation

We use ACM CCS to illustrate the use of SKOS to define domain ontology. The ACM CCS is probably the most comprehensive classification in the domain of computer science [2]. An excerpt of the classification in the SKOS is shown in Figure 2a. Note that we use the *SKOS Extension* properties *broaderGeneric* and *narrowerGeneric* in order to have subclass/superclass relations between concepts.

### 3.2 Ontology mapping

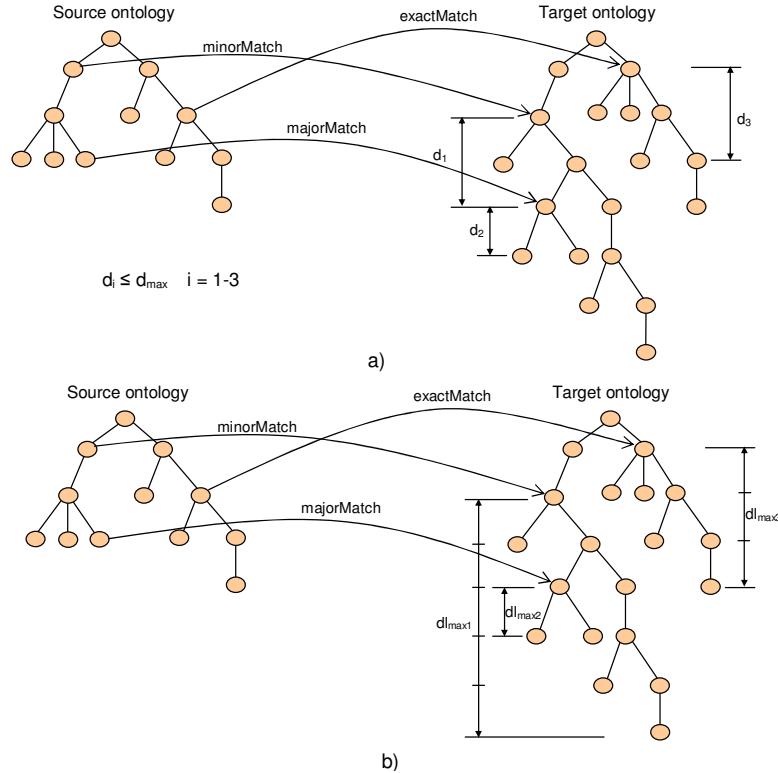
The *SKOS Mapping* vocabulary contains a set of properties for specifying mapping relations among concepts from different domain ontologies (*broadMatch*, *narrowMatch*,

*exactMatch*, *majorMatch*, *minorMatch*). Such a rich set of semantic relations for expressing mapping is useful in ranking search results to reflect the weight of the mapping. Apart from the properties, the SKOS Mapping has the three classes for defining: intersection of concepts (the AND class), union of concepts (OR), and negation (NOT).

In Figure 2b we show how we have used the SKOS Mapping to express the mapping between an e-learning relevant course curriculum ontology and the ACM CCS ontology. The curriculum ontology captures the *Information Management* course [1]. The course contains 14 units (top level SKOS concepts), and each unit contain several topics (sub-concepts of top level concepts in SKOS). One can see different match levels between concepts (i.e. *minorMatch*, *majorMatch*, and *exactMatch*) in Figure 2b. We also show how one defines the mapping relation between a concept (e.g. *IM1.6*) and a union (e.g. *OR1*) of other concepts (e.g. *H.3.3* and *E.2.3*). As mappings relations are not symmetric [17] we have to provide two mapping relations for each pair of concepts in case of two-way mappings.

## 4. ONTOLOGY MAPPING BASED SEARCH ALGORITHM

The substance of having mappings among different ontology-based vocabularies is to enable the use of the ontology A to search web resources annotated with concepts from another ontology B. Accordingly, we dedicate this section to the search algorithm we developed.



**Figure 3.** The search algorithm based on ontology mappings: a) only those child nodes  $d_{\max}$  levels below the matched node are used; b) all child nodes of the matched node in the target ontology are used

#### 4.1 Starting presumptions

The algorithm is based on the following presumptions:

- Input arguments of the search algorithm are concepts of the source ontology;
- Results of the search algorithm are concepts of the target ontology;
- Mapping relations among concepts from both source and target ontologies are defined using the SKOS Mappings;
- For each input argument the search algorithm looks for target ontology concepts that have defined mappings. We call those target ontology concepts – matched concepts.
- The search algorithm also looks for child concepts of matched concepts.
- When ranking search results, different kinds of the SKOS Mappings relations should be taken into account.

#### 4.2 Initial algorithm

The input argument of the initial algorithm is a concept from the source ontology. The algorithm searches for matched concepts in the target ontology based on all types of SKOS mappings relation types. Next, the algorithm looks for child concepts of the matched concepts, but only a predefined number of levels ( $d_{\max}$ ) below the matched concept in the target ontology (see Figure 3a).

The algorithm creates 5 different lists of matched concepts called *clusters* (one for each mapping relation type) as well as 5 clusters of child concepts ( $d_{\max}$  levels below) of the matched concepts. Finally, the algorithm merges all clusters respecting the order of clusters listed in the *cluster-names*

variable in Figure 4 (NB Figure 4 does not illustrate this algorithm version, but the next one). In fact, the merging is performed by connecting clusters using the union operator.

Although the algorithm in a rather simple way searches for the matched concepts in the target ontology as well as ranks the resulting set of matched concepts, it still has some open issues: the resulting concept list is completely discrete structure due to the simple merging; the ranking procedure treats all children of the matched concepts within the same cluster in the same way, so concepts within a cluster are randomly ordered; and searching for child concepts a predefined number ( $d_{\max}$ ) levels below the matched node can take out of consideration some relevant child concepts.

#### 4.3 Improved algorithm

First, the improved algorithm uses all the children of matched concepts in the target ontology regardless their depth level (see Figure 3b). Second, it uses the weight factor to determine ranks of both matched concepts and their children in the resulting list of concepts. The algorithm calculates the weight factor of a matched concept according to the type of the mapping relation connecting it with the source ontology concepts. The weight factor for each type of mapping relation is predefined (i.e. a constant number) and is subject to change depending on the tree structure of the target ontology (i.e. it can be fine tuned). Note also that referent weight factor is the *exactMatch* relation, while others (i.e. major, minor, and broad) are calculated relatively to it. That is the reason why that value is also an

```

function search-concept (input-concept, WFEM)
    cluster-names := {"exactMatch", "broadMatch", "exactMatchChildren",
                      "broadMatchChildren", "narrowMatch", "narrowMatchChildren", "majorMatch",
                      "majorMatchChildren", "minorMatch", "minorMatchChildren"};
    clusters := create-hash-map();
    result := {};
    for-each name in cluster-names
        matched-concepts := get-matched-concepts(name, input-concept);
        clusters[name] := matched-concepts;
    end-for-each
    for-each name in cluster-names
        for-each concept in clusters[name]
            put-in-sorted-list(result, concept, calculate-WF(concept, name));
        end-for-each
    end-for-each
    return result;
end-function

```

**Figure 4.** The search algorithm – considers all child concepts of the matched concept in the target ontology. It ranks the resulting list of concepts relying on the weight factor of the mapping relation type of the matched concept as well as the distance of child concepts from the matched (parent) concepts

input argument of the search algorithm (*WF<sub>EM</sub>*). The weight factor for every matched concept child depends on (see Figure 3b): the maximal depth level of the matched (parent) concept; the distance of the child concept from its parent; the weight factor of its parent. Accordingly, the weight factor of the child concept is calculated using the following formula:

$$WF_{ch} = WF_p - (WF_p / (1 + dl_{max})) * d_{ch} \quad (1)$$

where:

- $WF_{ch}$  – weight factor of the child concept;
- $WF_p$  – weight factor of the matched (parent) concept;
- $dl_{max}$  – maximal depth level of the matched (parent) concept;
- $d_{ch}$  – distance of the child concept from the matched (parent) concept.

Relaying on the aforementioned facts we revise the algorithm. In Figure 4 we show a high level description of the search function (i.e. *search-concept*) in an informal pseudo-code. The first part of the algorithm is similar to the previous version. The difference is that the clusters are not merged like in the first algorithm, but they are stored to be members of a hash map – a memory structure keeping the track about all clusters. Once all clusters are created, the algorithm puts the concepts from each cluster in the resulting list (*result*) using the *put-in-sorted-list* procedure. Concepts in the resulting list are sorted according to their weight factors. Since the same concept

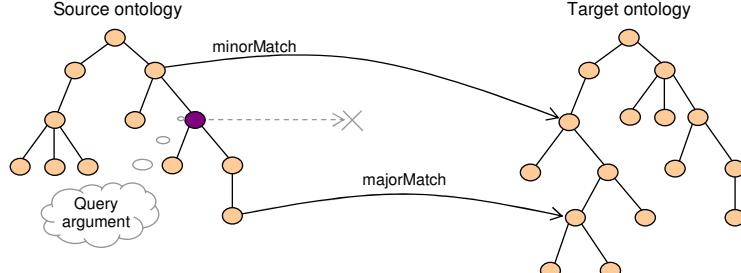
can be in more than one cluster (e.g. *broadMatchChildren* and *majorMatchChildren*), the procedure prevents the repetition of the same concept in the resulting list by using its best weight factor.

Although this variant of the algorithm solves the most the problems we have mentioned for the first one, the algorithm still has some limitations that are referred in detail in the next subsection.

#### 4.4 Final algorithm

The search algorithm presented in the previous section does not solve the case when mapping is not defined between the query argument and the target ontology (see Figure 5). Although the previous search algorithm variants look for children of matched concepts in the target ontology, it does not expand the query arguments that are parts of the source ontology. In fact, the solution works properly if mapping is defined between the query argument and one or more concepts from the target ontology. However, if there are no mappings defined for the query argument then the query will return an empty resulting list.

To overcome this issue, we additionally improved the search algorithm. The algorithm looks for both child and parent concepts in the source ontology that have defined mappings with concepts of the target ontology when the query argument has no defined mappings. In order to calculate weight factors of result concepts, the algorithm takes into account the fact that the distance between the



**Figure 5.** The case when mapping is not defined between the query argument and concepts of the target ontology

```

function search-concept-no-direct-match (input-concept, WFEM)
    result := search concept(input-concept, WFEM);
    if result == {} then
        children := get-subconcepts-with-mapping(input-concept);
        parents := get-superconcepts-with-mapping(input-concept);
        for-each c in children
            WF := calculate-WF(c, input-concept);
            put-in-ordered-list(result, search-concept(input-concept, WF));
        end-for-each
        for-each c in parents
            WF := calculate-WF(c, input-concept);
            put-in-ordered-list(result, search-concept(input-concept, WF));
        end-for-each
    end-if
    return result;
end-function

```

**Figure 6.** The final version of the search algorithm that captures the case exemplified in Figure 5 when none of mapping relations is defined among the query argument and concepts of the target ontology

query arguments and all its child and parent concepts with defined matching relations is not the same. Relying on that fact we calculate the value of the weight factor ( $WF_i$ ) for exact match (see the previous subsection) for each parent and child concept of the query argument in the source ontology using the following formula:

$$WF_i = WF_{EM} - \text{abs}(dl_{sc} - dl_i) * \text{step} \quad (2)$$

where:

- $WF_{EM}$  – weight factor of the exact match relation predefined for the case when there is a mapping relation between the query argument and the target ontology;
- $dl_{sc}$  – depth level of the query argument;
- $dl_i$  – depth level of a parent/child concept of the query argument that has a mapping relation with the target ontology;
- step – predefined value that specifies the impact of the distance between the query argument and its child/parent concept  $i$ .

Once we calculate the weight factor for the exact match relation, the weight factors of other mapping relations can be calculated as we have already explained in the previous subsection. In Figure 6 we show the final version of the search algorithm capturing the explanation given in this subsection and employing the search algorithm shown in Figure 4.

#### 4.5 A Jess-based implementation of the proposed search algorithm

We implemented the algorithm using OWLJessKB [15], a Semantic Web reasoning tool, and Jess, a Java-implemented rule-based inference engine [10]. The use of the implemented algorithm regards invoking the corresponding Jess function whose input parameters are: a concept from the source ontology; and the weight factor for the exact match mapping relation. The function returns a ranked list of matched concepts as well as their child concepts.

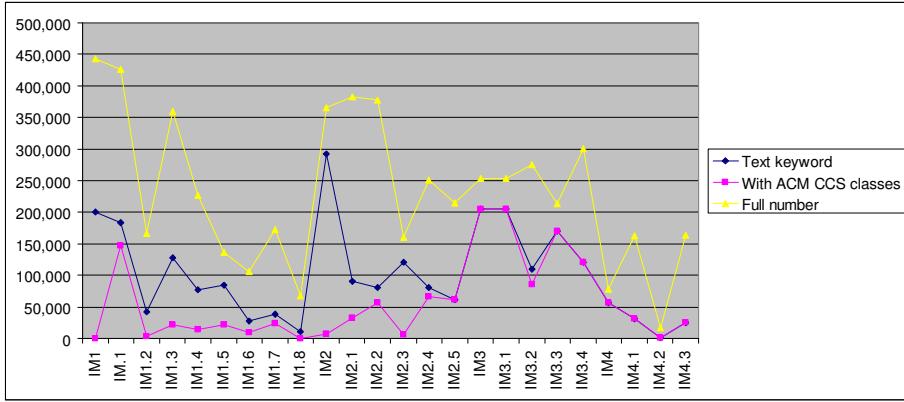
### 5. EVALUATION

In order to evaluate the search algorithm we developed a web-based application in the domain of e-learning for an

information management course. The course is based on the ACM/IEEE computer science curriculum recommendation [1]. The application has a typical organization – the left pane contains the course structure and the right pane holds the content of the one particular unit. In fact, the course structure is represented as a SKOS ontology. The bottom part of the right pane contains the context sensitive search for two different collections of web resources: the ACM Digital Library (DL) [3] and Merlot learning object repository (<http://www.merlot.org>). The ACM DL relays on the ACM CCS ontology while Merlot has its own classification. We encoded both classifications in SKOS. The students can search both collections of web resources by providing search keywords. However, the search action collects annotation of the current page in the course ontology (embedded in the web page in the RDF form) and applies the ontology mapping based algorithm to the annotation. Finally, the application sends an expanded query along with the keywords to chosen collection of web resources. The results received from collections of web are related to the current web page within the course.

Figure 7 contains the diagram comparing the search results we obtain when searching the ACM DL by using keywords and the combination of keywords and ACM CCS class identifiers. Note that the “full number” for each query means the overall number of objects that contain any of searched keywords in the keyword and ACM CSS classification fields of their metadata (see [3] for details). It is obvious that the combination of the keywords and ACM CCS class identifiers reduces the number of found objects, and hopefully helps find more relevant web resources. However, we noticed several peculiarities due to the use of a specific search engine (i.e. the ACM DL Advanced Search):

- *The number of results is decreased when increasing the number of classifiers in a query.* It was completely opposite to expectations as those query arguments are connected with the OR logical operator. The reason for such a behavior is that the ACM DL Advanced Search uses the Verity indexing engine (<http://www.verity.com>),



**Figure 7.** Comparison of the search results obtain from the ACM Digital Library by using text-based keywords and the combination of the text-based keywords and ACM CCS class identifiers

which selects only those objects whose weight factors pass over a specific threshold. Since their weight factors depend on the number of classification parameters, the less number of found objects can pass over the threshold

- *The last level of classification is omitted from the queries in the web application.* Due to a high number of classification concepts used in queries, the ACM DL Advanced Search can return an empty list of found concepts. The effect is especially stressed when using top level concepts as query arguments. The ACM DL search engine selects only those objects whose weight factors pass over a specific threshold. Increasing the number of classification parameters also increases the threshold and therefore eliminating some of the objects. However, this does not affect the best matching results.

Table 1 contains results obtained by applying the algorithm to search the Merlot learning object repository using the combination of the course ontology concepts and keywords. Unlike the ACM DL, in this case search results are in accordance with expectations, the greater number of classification tags in the query, the greater number of the found objects. Note also that the number of Merlot classification tags is not so high comparing to the experiment with ACM DL, since mapping relations between two ontologies are defined for the bottom level concepts of the Merlot classification. We found that most of

concept from the course ontology did not have direct mapping relations with the target ontology, but they just inherited mapping relation from their parents.

Finally, say that we could not rank search results according to our ranking algorithm in either experiment, since we used two different digital libraries where we did not have any control in ranking of the found resources. In order to evaluate our ranking algorithm we are developing screen-scraping functions of both ACM DL and Merlot web pages showing search results.

## 6. CONCLUSIONS

The paper presented a way to achieve semantic interoperability when searching for web resources in web sources annotated by different domain ontologies. That way, users can get semantically relevant search results using classifications they are familiar with. The presented method exploited the idea of having one separated mapping ontology as it was already shown in [9, 16, 20]. Relations among different ontologies are not encoded into their structures, but they are represented separately. Accordingly, reusability of related ontologies is not decreased. Additionally, the evaluation examples showed benefit to have a combined keyword search with ontology annotated content to in order to provide more relevant web resources. Although we discussed the case of mapping between two ontologies, the described approach scales up to support

**Table 1.** Evaluation of the ontology mapping based algorithm to search Merlot learning object repository, which is based on its own classification, using a course curriculum ontology

Concept	IM1	IM.1	IM1.2	IM1.3	IM1.4	IM1.5	IM1.6	IM1.7	IM1.8	IM2	IM2.1	...
Keyword-based search	9814	10782	9760	2094	9769	9578	114	9760	9542	540	10797	...
Ontology-based search	55	59	85	22	53	80	1	9	52	25	35	...
Percent	0.56	0.55	0.87	1.05	0.54	0.84	0.88	0.09	0.54	4.63	0.32	...
Num. of classification tags	1	1	3	2	1	1	1	1	1	2	2	...
Defined match or not	Y	Y	Y	Y	Y	Y	N	Y	N	Y	N	...

Concept	IM2.2	IM2.3	IM2.4	IM2.5	IM3.2	IM3.3	IM3.4	IM4	IM4.1	IM4.2	IM4.3	
Keyword-based	...	9449	1321	9638	12782	9614	418	1140	72	12788	9544	9563
Ontology-based	...	36	26	35	38	85	14	40	6	38	31	31
Percent	...	0.38	1.97	0.36	0.30	0.88	3.35	3.51	8.33	0.30	0.32	0.32
Num. of classification tags	...	2	2	2	2	3	3	3	2	2	2	2
Defined mapping or not	...	N	N	N	N	N	N	Y	N	N	N	...

multiple ontologies, provided we defined mapping between the source ontology and any number of target ontologies.

Comparing the search algorithm with other solutions we can find similarities with the Intelligent Product Information Search that employs ontology mapping to search for products using web services of several sellers based on different product ontologies [14]. However, this method has mappings defined in a table, while search procedure just considers direct mapping without consideration of child concepts. It further uses run-time discovery of mapping relations based upon lexical similarities defined in WordNet. Another similar approach tries to enable the use of user personalized ontologies to annotate web pages in order to compose web services [5]. The mapping rules between ontologies are defined in F-Logic. To the best of our knowledge, the approach just uses simple matching between related concepts from different concepts, without consideration of their child concepts.

In the future we plan to integrate the developed search algorithm into the eduSource Communication Layer (ECL) being developed in our research laboratory as a part of its federated search engine [12]. We also plan to research how we can automatically generate ontology mapping relations the search algorithm relays on. The idea is to apply the concept of semantic signatures as well as content of web resources to discover relation among ontology concepts [8].

## 7. ACKNOWLEDGMENTS

This work was funded by Canada's LORNET NSERC Research Network. The authors would especially like to thank Bernard Rous and Robert Okajima from the ACM Headquarter Office who provided details about the Advance Search engine of the ACM Digital Library.

## 8. REFERENCES

- [1] Information Management Course Curriculum Recommandation (2001). <http://www.computer.org/education/cc2001/final/im.htm> (2001)
- [2] ACM Computing Classification System (CCS), <http://www.acm.org/class/1998/> (2005)
- [3] Association for Computing Machinery (ACM) Digital Library, <http://www.acm/dl> (2005)
- [4] Dublin Core Metadata Schema, <http://dublincore.org/documents/dcmi-terms/> (2005)
- [5] Agarwal, S., Handschuh, S. and Staab, S. Surfing the Service Web, in Proc. of the 2nd Int'l Semantic Web Conf. (Sanibel Island, FL, USA, 2003), 211-226.
- [6] Bechhofer, S., van Harmelen, F. and Hendler, J., et al. OWL Web Ontology Language Reference, <http://www.w3.org/TR/owl-ref/> (2004)
- [7] Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine, Computer Networks and ISDN Systems 30, 1-7 (1998), 107-117.
- [8] Choi, A. and Hatala, M. Towards Browsing Distant Metadata Using Semantic Signature, in Proc. of the KCAP2005 Workshop on Integrating Ontologies (Banff, Canada, 2005), (forthcoming).
- [9] Crubézy, M. and Musen, M.A. Ontologies in Support of Problem Solving, *Handbook on Ontologies in Information Systems* (Berlin: Springer-Verlag, 2003), 321-341.
- [10] Friedman-Hill, F. *Jess in Action* (Greenwich, USA: Manning Publications Co., 2003).
- [11] Greenberg, J. and Robertson, W.D. Semantic Web Construction: An Inquiry of Authors' Views on Collaborative Metadata Generation, in Proc. of the Int'l Conf. on Dublin Core and Metadata for e-Communities 2002 (Florence, Italy, 2002), 45-52.
- [12] Hatala, M., Richards, G., Eap, T. and Willms, J. The Interoperability of Learning Object Repositories and Services: Standards, Implementations and Lessons Learned, in Proc. of the 13th World Wide Web Conf. (NY, USA, 2004), 19-27.
- [13] Kalfoglou, Y. and Schorlemmer, M. Ontology mapping: the state of the art, *The Knowledge Engineering Review* 18, 1 (2003), 1-31.
- [14] Kim, W., Choi, D.W. and Park, S. Product Information Meta-search Framework for Electronic Commerce Through Ontology Mapping, in Proc. of the 2nd European Semantic Web Conf. (Heraklion, Crete, Greece, 2005), 408-422.
- [15] Kopena, J. and Regli, W.C. DAMLJessKB: A Tool for Reasoning with the Semantic Web, *IEEE Intelligent Systems* 18, 3 (2003), 74-77.
- [16] Maedche, A., Motik, B., Silva, N. and Volz, R. MAFRA - A MAppling FRamework for Distributed Ontologies, in Proc. of the 13th Int'l Conf. on Know. Engineering and Know. Management, Ontologies and the Semantic Web (Siguenza, Spain, 2002), 235-250.
- [17] Miles, A. and Brickley, D. SKOS Mapping Vocabulary Specification, W3C Document, <http://www.w3.org/2004/02/skos/mapping/spec> (2004)
- [18] Miles, A. and Brickley, D. SKOS Core Vocabulary Specification, W3C Working Draft, <http://www.w3.org/TR/swbp-skos-core-guide> (2005)
- [19] Noy, N.F. Representing Classes As Property Values on the Semantic Web, W3C Working Group Note, <http://www.w3.org/TR/swbp-classes-as-values/> (2005)
- [20] Noy, N.F. and Musen, M.A. The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping, Int'l Journal of Human-Computer Studies 59, 6 (2003), 983-1024.
- [21] Park, J.Y., Gennari, J.H. and Musen, M.A. Mappings for Reuse in Knowledge-based Systems, in Proc. of the 11th Workshop on Knowledge Acquisition for Knowledge-Based Systems (Banff, Canada, 1998)
- [22] Rasseneur, D., Jacoboni, P. and Tchounikine, P. Enhancing a Web-based Distance-Learning Curriculum with Dedicated Tools, in Proc. of the IEEE/WIC Int'l Conf. on Web Intelligence (Halifax, Canada, 2003), 285-291