Ontology Selection for the Real Semantic Web: How to Cover the Queen's Birthday Dinner?

Marta Sabou, Vanessa Lopez, and Enrico Motta

Knowledge Media Institute (KMi) & Centre for Research in Computing,
The Open University, Milton Keynes,
{r.m.sabou, v.lopez, e.motta}@open.ac.uk

Abstract. Robust mechanisms for ontology selection are crucial for the evolving Semantic Web characterized by rapidly increasing numbers of online ontologies and by applications that automatically use the associated metadata. However, existing selection techniques have primarily been designed in the context of human mediated tasks and fall short of supporting automatic knowledge reuse. We address this gap by proposing a selection algorithm that takes into account 1) the requirements imposed by two applications that explore large scale, distributed markup and 2) some properties of online ontology repositories. We conclude that the ambitious context of automatic knowledge reuse imposes several challenging requirements on selection.

1 Introduction

The effort of the Semantic Web community to migrate and apply its semantic techniques in open, distributed and heterogeneous Web environments has paid off: the Semantic Web is evolving towards a *real* Semantic Web. Not only has the number of ontologies dramatically increased, but also the way that ontologies are published and used has changed. Ontologies and semantic data are published on the open Web, crawled by semantic search engines (e.g., Swoogle [3]) and reused by third parties for other purposes than originally foreseen (e.g., Flink [9] derives social networks from automatically crawled FOAF profiles). Many ontology based tools are evolving from relying on a single, fixed ontology to harvesting the rich ontological knowledge available on the Web [6].

Robust mechanisms for selecting ontologies are crucial to support knowledge reuse in this large scale, open environment. Obviously, the context of reuse has a major influence on the requirements for the selection algorithm and should be taken into account when developing such algorithms. We further discuss and contrast the requirements imposed by human mediated and automatic reuse. As a context for our discussion, consider the following news snippet[1]:

The Queen will be 80 on 21 April and she is celebrating her birthday with a family dinner hosted by Prince Charles at Windsor Castle.[2]

[1] Example inspired by Her Majesty's birthday coinciding with the submission deadline.
[2] http://news.billinge.com/1/hi/entertainment/4820796.stm

Human mediated tasks. Imagine a person wishing to annotate this news snippet and in search of an ontology containing the *Queen, birthday* and *dinner* concepts. When queried for these terms, a selection mechanism is expected to return an ontology that best covers them. It is not a problem if the returned ontology contains only a subset of the terms (partial coverage) as the user can extend the ontology according to his needs. It is also admissible for the system to make mistakes when mapping between the query terms and ontology concepts as the user can filter out such errors (imprecise coverage). For example, ontologies containing the concept *Queen* as a subclass of *Bee* or *dinner_fork* as an approximation for *dinner* will be rejected as irrelevant for the user's context. Finally, users are willing to wait some minutes for reusable ontologies, since this time is negligible compared to that needed to build an ontology from scratch.

Automatic knowledge reuse. As opposed to the previous scenario, imagine that the output of the selection is automatically processed. For example, a semantic browser such as Magpie [4] which identifies and highlights entities of a certain type in Web pages needs to find an ontology according to which to describe the page above. The requirements are much stricter than before. First, a *complete coverage* of the query terms is needed to fully support the sense making activity offered by the browser. If no completely covering ontology is found, a set of ontologies that jointly cover the query could be returned. Or, alternatively, an ontology with more generic concepts such as *woman, event* and *meal* could be useful, provided that a machine interpretable explanation of the relation between the query terms and the returned concepts is available (e.g., a *dinner* is a kind of *meal*). Indeed, another requirement relates to the quality of mappings between terms and concepts. Errors such as those described in the context of human mediated tasks are not admissible. Finally, a quick response becomes more important when the selection is used at run time as in this case.

The four selection mechanisms that we are aware of (see [11] for a detailed description and comparison) have been developed in the context of human mediated tasks. OntoKhoj [10] and Swoogle [3] complement automatically populated ontology libraries and use a PageRank-like algorithm on semantic relations between ontologies (e.g., imports) to select the most popular ontology containing a certain term. OntoSelect [2], also part of an ontology library, combines measures about coverage, popularity and the richness of ontology structure. Finally, the ActiveRank [1] algorithm is independent of an ontology library and uses a set of ontology structure based metrics. These metrics were inspired by and reflect characteristics that human ontology builders find important, for example, coverage, compactness, density (richness of knowledge structure).

All these approaches function well in the context of human mediated tasks, but are insufficient when it comes to automatic knowledge reuse. Regarding the level of coverage, none of the existing approaches enforces complete coverage. In fact, Swoogle and OntoKhoj can be queried only with one term at a time. Further, the quality of the mapping between query terms and concept labels is quite low. All these approaches rely only on syntactic matches between query terms and ontology concepts. For example, ActiveRank, currently the most advanced

algorithm, uses a fuzzy match between terms and concept names (i.e., *project* is mapped to *projectile*) but makes no provision to filter out obviously irrelevant hits. The position of the concept in the hierarchy is not considered by any of the approaches. Finally, our only indication about performance is that ActiveRank needs 2 minutes to evaluate each ontology - a baseline that needs improvement in the case of automated tasks.

We consider that, while ambitious, the context of automatic knowledge reuse is the one that selection algorithms should aim for. In this paper we derive a set of requirements imposed by two applications that are extended to perform automatic knowledge reuse (Section 2) and we present an initial design of a selection algorithm that balances between obtaining a complete and precise coverage and offering a good performance (Section 4). Our design also considers characteristics of online ontologies explored through a set of indicative experiments (Section 3).

2 Requirements for Supporting Automatic Knowledge Reuse Scenarios

While current ontology selection tools primarily target human users, we are working on two Semantic Web tools (Sections 2.1 and 2.2) that are evolving from using a single, rich and manually crafted ontology to exploring and combining ontologies available on the Web. These tools rely on automatic ontology selection on which they pose a set of requirements (Section 2.3).

2.1 Ontology based Question Answering

AquaLog [7] is an ontology based question answering system which relies on the knowledge encoded in an underlying ontology to disambiguate the meaning of questions asked using natural language and to provide answers. To shortly give an impression about how the system operates, consider that it is aware of an ontology about academic life[3] which has been populated to describe KMi related knowledge[4]. Also, suppose that the following question is asked[5]:

Which projects are related to researchers working with ontologies?

In a first stage the system interprets the natural language question and translates it to triple-like data structures. Then, these triples are compared to the underlying ontology centered knowledge base using a set of string comparison methods and WordNet. For example, the term *projects* is identified to refer to the ontology concept *Project* and *ontologies* is assumed equivalent to the *ontologies* instance of the *Research-Area* concept. After the modifier attachment is resolved by using domain knowledge, two triples are identified:

(projects, related to, researchers) and *(researchers, working, ontologies)*

[3] `http://kmi.open.ac.uk/projects/akt/ref-onto/`.
[4] See the populated ontology at `http://semanticweb.kmi.open.ac.uk`.
[5] See the AquaLog demo at `http://kmi.open.ac.uk/technologies/aqualog/`.

The relations of the triples are also mapped to the ontology. For example, for the second triple, there is only one known relation in the ontology between a *Researcher* and a *Research-area*, namely *has-research-interest*. This relation is assumed to be the relevant one for the question. However, when disambiguating the relation that is referred to by *related to*, the system cannot find any syntactically similar relation between a *Project* and a *Researcher* (or between all more generic and more specific classes of the two concepts). Nevertheless, there are four, alternative relations between these two concepts: *has-contact-person, has-project-member, has-project-leader, uses-resource*. The user is asked to choose the relation that is closest to his interest. Once a choice is made, the question is entirely mapped to the underlying ontological structure and the corresponding instances can be retrieved as an answer.

While the current version of AquaLog is portable from one domain to another, the scope of the system is limited by the amount of knowledge encoded in the ontology used at that time. This new implementation of AquaLog, PowerAqua [6], overcomes this limitation by extending the system in the direction of open question answering, i.e., allowing it to benefit from and combine knowledge from the wide range of ontologies that exist on the Web. One of the challenges is the selection of the right ontology for a given query from the Web.

2.2 Semantic Browsing

The goal of semantic browsing is to exploit the richness of semantic information in order to facilitate Web browsing. The Magpie [4] Semantic Web browser provides new mechanisms for browsing and making sense of information on the semantic Web. This tool makes use of the semantic annotation associated with a Web page to help the user get a quicker and better understanding of the information on that page. Magpie is portable from one domain to another as it allows the user to choose the appropriate ontology from a list of ontologies that are known to the tool. However, similarly to AquaLog, the current version relies on a single ontology being active at any moment in time. This limits the scope of the sense making support to the content of the current ontology.

Our current research focuses on extending Magpie towards open browsing. This means that the tool should be able to bring to the user the appropriate semantic information relevant for his browsing context from *any* ontology on the Web. This extension relies on a component that can select, at run time, the appropriate ontologies for the given browsing context.

In the case of Magpie, the query for the ontology selection is more complex than for AquaLog as it is defined by the current browsing context. This includes the content of the currently accessed Web pages and, optionally, the browsing history and the profile of the user. Web pages typically span several different topics. For example, the following short news story[6] is both about trips to exotic locations and talks. Therefore, the query sent to the selection mechanism is likely to contain terms drawn from different domains.

[6] `http://stadium.open.ac.uk/stadia/preview.php?s=29&whichevent=657`

"For April and May 2005, adventurer Lorenzo Gariano was part of a ten-man collaborative expedition between 7summits.com and the 7summits club from Russia, led by Alex Abramov and Harry Kikstra, to the North Face of Everest. This evening he will present a talk on his experiences, together with some of the fantastic photos he took."

2.3 Requirements for Ontology Selection

Hereby we formulate the requirements imposed by our applications on ontology selection and discuss to which extent they are addressed by current approaches. These requirements drove the design of our selection algorithm (Section 4).

1. **Complete coverage.** A complete coverage is probably the most important requirement for our applications. Because in these applications the retrieved knowledge is automatically processed, they require that all the needed knowledge should be retrieved. While existing approaches rank ontologies that cover most terms as best, they do not enforce a complete coverage.
2. **Precise coverage.** Automatic knowledge integration requires a rigorous mapping between query terms and ontology concepts as well as a formal representation of the mapping relation (i.e., equivalence, more generic, more specific). Assuming that a human user would filter out (and eventually enrich) the returned ontologies, current approaches treat the comparison between query terms and ontology concepts rather superficially, relying only on (often only approximate) lexical comparisons.
3. **Returning ontology combinations.** Our preliminary experiments indicate that the sparseness of knowledge on the Web often makes it impossible to find a single ontology that covers all terms (Section 3). However, it is more likely to find ontology combinations that jointly cover the query terms. Currently, however, all selection mechanisms return only a single ontology.
4. **Performance.** Our applications rely on the results of selection at run time and therefore require a good performance. While simple selection tools perform rather well, the more complex ActiveRank needs 2 minutes per ontology to compute all its metrics. This is acceptable for supporting ontology building, but needs to be improved in an automatic scenario.
5. **Dealing with relations.** Our applications, especially PowerAqua, illustrate a need for considering relations and not just concepts when selecting an ontology. Currently, only OntoSelect considers relations.
6. **Dealing with instances.** Our applications help users in their information gathering activities. Most often, people are interested in finding out things about certain entities rather than generic concepts. This requires that selection should consider instances as well (i.e., match between instances in a query and those in online ontologies). Matching instances is a difficult problem in itself given the large number and high level of ambiguity when dealing with instances (e.g., many instances can share the same or similar names).
7. **Modularization.** Knowledge reuse is closely related to ontology modularization. Indeed, our tools would require selection mechanisms to return a

relevant ontology module rather than a whole ontology. Note that the work in [1] has already considered this issue when introducing a metric to measure how close the hits in an ontology are (assuming that this indicates the existence of a module). As with instance mappings, ontology modularization is a difficult and as yet unsolved issue, though a large amount of work in this area [13] could be reused to some extent.

3 Preliminary Experiments

To better design our algorithm, we wanted to get an insight in the characteristics of the ontological data available online. Since the requirement of complete and precise coverage of the query terms was identified as the most important one in the context of automatic knowledge reuse, our experiments are centered towards 1) exploring the factors that hamper obtaining a complete coverage and 2) getting an insight in the nature of compound concept labels in preparation to provide a more precise mapping to query terms. We performed both experiments on top of Swoogle[7] because it is currently the largest ontology repository. It is important to note that our experiments have an exploratory role rather than trying to rigourously test our hypotheses.

3.1 Experiment 1 - Obtaining Complete Coverage

The goal of this experiment is to get an indication about how difficult it is to find a completely covering ontology when using Swoogle. One of the motivations for this experiment was that, while important, complete coverage has not been investigated in any previous work. In fact, with the exception of OntoSelect, all selection algorithms are tested for the rather trivial case of one or two query terms. On the contrary, our tools require ontologies that cover at least three query terms (e.g., AquaLog translates each question in one or more triples).

Our intuition was that the *number, topic relatedness* and *type* of the query terms will influence the success of obtaining an all covering ontology. Namely, a single, all covering ontology is difficult to find if 1) there are many query terms, 2) if query terms are drawn from different topic domains or 3) relations are considered. According to these considerations, we devised four sets of queries. The first three queries represent an optimal scenario where few concepts are drawn from the same domain (we chose a well covered domain in terms of online ontologies, the academic domain). The second set of queries (queries 4 - 6) have terms drawn from different (and less covered) topic domains. They were inspired by the actual text snippets in Section 1 and Section 2.2, therefore being representative for real life scenarios encountered in the context of Magpie. The queries in set three (7 - 10) have terms drawn from the same domain but, unlike the first set, contain a relation as well (these are typical AquaLog queries). Our final queries (11 - 14) explore overcoming failure of finding a completely covering ontology by replacing query terms in queries 4, 6, 9 and 10 with their hypernyms.

[7] We use Swoogle 2005 as our software was written before Swoogle 2006 was released.

The experimental software queries Swoogle for ontologies that contain concepts/properties that *exactly* match the query terms (variations in capitalization are allowed). For each query, the software outputs the number of ontologies that cover each term, their pairwise combinations and all terms.

Query	(t_1, t_2, t_3)	(t_1)	(t_2)	(t_3)	(t_1, t_2)	(t_1, t_3)	(t_2, t_3)	(t_1, t_2, t_3)
1	(project, article, researcher)	84	90	24	19	13	9	8
2	(researcher, student, university)	24	101	64	16	15	38	13
3	(research, publication, author)	15	77	138	8	5	36	4
4	(adventurer, expedition, photo)	1	0	32	0	1	0	0
5	(mountain, team, talk)	12	25	9	2	1	1	1
6	(queen, birthday, dinner)	0	9	2	0	0	1	0
7	(project, relatedTo, researcher)	84	11	24	0	13	0	0
8	(researcher, worksWith, Ontology)	24	9	52	0	3	0	0
9	(academic, memberOf, project)	21	36	84	0	3	5	0
10	(article, hasAuthor, person)	90	14	371	8	32	2	0
11 (4+)	(person, trip, photo)	371	7	32	1	20	1	1
12 (6+)	(woman, birthday, dinner)	32	9	2	1	1	1	1
13 (9+)	(person, memberOf, project)	371	36	84	16	46	5	5
14 (10+)	(publication, hasAuthor, person)	77	14	371	2	52	2	2

Table 1. Number of ontologies retrieved for a set of queries.

The results are summarized in Table 1. Notice that as the number of terms increases less completely covering ontologies are found. The drop in the number of returned ontologies is significant when adding even one extra term. This phenomena is evident throughout the table even in our optimal scenario where terms were chosen from the same, well covered domain.

Our second set of queries containing terms drawn from different topic domains return less ontologies than previously (mostly zero). At a closer look, however, one might argue that the null results are caused by the fact that the domains from which the terms were drawn are weakly covered in Swoogle in the first place (indicated by the low number of ontologies returned for individual terms). While this observation does not necessarily undermine the intuition that topic heterogeneity has negative effects, it indicates that the knowledge currently available online is sparse, as many domains are weakly covered (or not at all). Therefore, null results can be expected even when query terms are topically related but refer to a weakly covered topic.

The third set of experiments indicates that the presence of relations seriously hampers retrieving an all covering ontology even when the query terms are chosen from the same, well represented domain.

In the last four queries, by refining query terms through hypernym replacement, better results were obtained. An obvious worry is that if the refinement

uses too generic terms (e.g., entity) the returned ontologies will be too generic to be of any use for the concrete knowledge reuse task at hand.

While only preliminary, our experiments do indicate that query size, topic heterogeneity and type might influence the chance to find an all covering ontology. They have also revealed the sparseness of the online knowledge. As a bottom line, independently of having verified our intuitions, we can observe that the chance to find an all covering ontology is rather low, especially in scenarios such as those provided by Magpie (many terms, drawn from different, possibly weakly represented domains) and AquaLog (properties as query terms).

3.2 Experiment 2 - Dealing with Compound Labels

Considering the results of the previous experiment, some mechanisms might be needed to expand the search for potentially relevant ontologies. Besides the synonym/hypernym extension, the more lexical oriented strategy of selecting concepts whose labels partially match the query terms can be explored. For example, Swoogle's fuzzy search functionality returns concept labels that contain the query term as a substring. This mechanism is rather brittle, and, while it returns several important hits (e.g., *GraduateStudent* when searching for *Student*), it also generates clearly invalid hits (e.g., *update* when searching for *date*, or *Team, Steam_engine* and *Teach* when searching for *tea*).

To ensure our second requirement referring to precise coverage, all the compound labels returned by fuzzy search need to be interpreted in order to understand their relation with the query term. A special case of compound labels are those containing conjunctions (e.g., *BlackAndWhite*). Some researchers have proposed a set of rules to interpret such labels [8]. Naturally, reading, splitting and interpreting all these labels can seriously hamper the performance, thus questioning the usefulness of performing a fuzzy search at all.

In this experiment we explore the feasibility of performing fuzzy search. We illustrate some cases when it pays off and some when it does not. We also evaluate how frequently conjunctions are used in compound labels.

To support our experiments we implemented a program (LabelSplitter) that splits compound labels according to the most common naming conventions and checks if a given term is a well formed part of that label (i.e., its base form is the same as the base form of one of the components of the label). For example, *TeachingCourse* is a relevant compound label (CL) for the term *teach*, but an irrelevant one for the term *tea*. In Table 2 we summarize the results obtained when querying some random terms and then some conjunctions showing the total number of hits returned by Swoogle, which is broken down into the number of exact matches, relevant and irrelevant CLs.

As expected, fuzzy search is a good mechanism to broaden the search space as it can return a lot of broader hits that contain the term. In general, in the case of longer words (less likely to be substrings of other words) more relevant than irrelevant compound labels are found. This is not true in the case of shorter words such as *tea* where an overwhelming number of irrelevant hits are returned.

Word	Total	Exact	Relevant CLs	Irrelevant CLs
project (clarifying example)	-	project	PastProject ProjectPartner	Projectile Projector
project	644	90	413	141
student	190	84	97	9
tea	492	3	23	466
mountain	36	12	21	4
university	86	64	22	0
and	2363	37	444	1882
or	18178	11	184	17983
of	6353	4	4743	1606
not	840	23	77	740
except	45	0	0	45
but not	0	0	0	0

Table 2. Analysis of the appearances of some conjunctions and other terms.

Therefore, taking into account that fuzzy search is rather expensive, it should be used only when absolutely necessary.

Regarding the frequency of conjunctions, in current online ontologies "or" appears the most frequently but in the large majority of cases as a substring and not a well formed part. While the "of" conjunction appears less often than "or" it is the most frequently used as a proper part of the compounds (mostly as part of property labels). "And" appears quite frequently as well in its role of well formed part (444). Surprisingly, negation and disjunction indicators appear infrequently or at all in the collection that we have queried. We conclude that interpretation rules for some conjunctions have to be written.

4 The Algorithm

In this section we present the design of an algorithm which aims to address some of the requirements stated in Section 2.3 and also draws on our conclusions regarding the nature of online ontologies detailed in the previous section. We first give an overview of the method in which we motivate our main design choices and then explore each major step of the algorithm in detail. The algorithm has been entirely specified and partially implemented (with the exception of the ideas reported in Sections 4.4 and 4.6) .

4.1 Overview

For our first implementation we wish to satisfy the first five requirements: we aim to identify ontologies (or combinations of ontologies - R3) that completely and precisely cover our query (R1 and R2). The query can contain both concepts and relations (R5). The performance of the algorithm should be such that it can

be used by other applications at run time (R4). The final two requirements, related to instances and modularization, are not addressed yet.

From our experiments we have learned that in cases when query terms are drawn from different domains or when they represent relations it is challenging to find ontologies that would cover all terms (therefore R1 is not so easy to fulfill). We have also seen that in such cases the search space can be expanded either 1) by query expansion with semantically related terms or 2) by searching for labels that incorporate the query term. However, our second experiment indicates that fuzzy search should be used only when absolutely needed.

Given these considerations, we have designed an algorithm that adapts itself to the particular context and can employ increasingly complex methods in order to achieve a complete coverage. The algorithm in Figure 1 executes increasingly complex combinations of a couple of main steps until complete coverage is achieved. We will first explain the role of each step and then describe how they are combined in increasingly complex stages.

Step1: Query Expansion. This step supplements the query terms with their semantically related terms such as synonyms an hypernyms.

Step2: Ontology identification. In this step we identify ontologies that cover to some extent the query terms. After an initial syntactic mapping between query terms (either exact or fuzzy) and ontology concepts, we perform a more in depth analysis of these mappings and define their semantic type (i.e., exact, generic or more specific). We call this task semantic match.

Step3: Identify ontology combinations. Using the output of the previous step, here we decide on the ontology combinations that provide a complete coverage of the query terms.

Task4: Generality Ranking. The ontologies that are returned contain hits that can be more generic or more specific than the query terms. In this step we evaluate the ontology combinations according to their level of generality and choose those with the appropriate level of abstraction.

These basic steps are combined in the following increasingly complex and expensive stages. The algorithm enters in a new stage only if the previous stage has failed:

Stage I relies on the simplest combination of the main steps. It uses an exact match to identify relevant ontologies thus circumventing complex semantic matching and the generality ranking step. This stage is likely to succeed only if the query terms are few or drawn from the same, well covered domain.

Stage II is used only if Stage I fails (no ontology was found for at least one term) and some kind of broadening of the search space is needed. Query expansion is used for the problematic terms and then the same ontology identification and combination steps as in stage I are performed. Notice that at this stage we can already use the generality ranking step because query broadening is likely to identify hypernyms for some of the query terms.

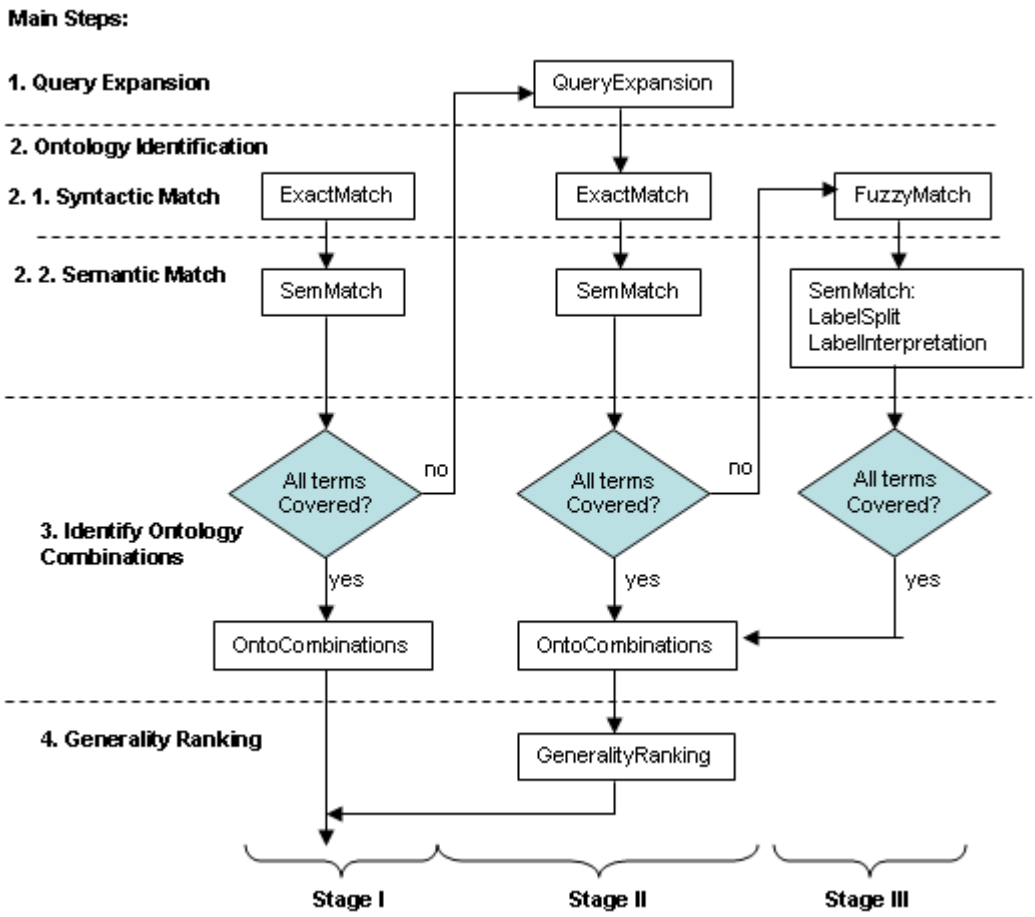

Fig. 1. The main tasks and stages of the selection algorithm.

Stage III is the most complex one, as besides query expansion, it also relies on more flexible syntactic matching to identify even more concepts potentially related to the query terms. This fuzzy match complicates the semantic matching step as the retrieved compound labels need to be split and interpreted. After the semantic match has identified the semantic relations between query terms and ontology concepts we apply the ontology combination and generality ranking steps.

4.2 Step1: Query Expansion

Query expansion is needed in order to broaden the search space for ontologies in cases when no or few ontologies are returned for a term. Our experiments indicate that such cases will be often encountered given the knowledge sparseness of online

ontology collections. Term expansion allows searching not just for the term but for all its semantically related terms (e.g., synonyms, hypernyms). This can be allowed because we aim to perform a semantic rather than a syntactic selection and therefore synonyms that denote the same concept as the query term are relevant. Currently, we use WordNet to augment each query term with their synonyms and hypernyms (we assume that the senses of query terms are known prior to running the selection). The only system that uses a similar expansion approach is OntoKhoj [10].

4.3 Step2: Ontology Identification

In this step we identify ontologies that contain the concepts specified in our query. This is in essence a mapping stage between the query terms and the concepts of the ontologies. We distinguish two substages:

Step 2.1. Syntactic Match. The syntactic match identifies lexically similar concept labels. It can be either *exact* (the query term is exactly the same as the concept label) or *fuzzy* (the query term is a substring of the concept label, e.g., the term *Student* is part of *GraduateStudent*). In the case when a fuzzy match is performed, this step is also responsible for splitting the compound labels and returning only the compound labels that are relevant for the given term (as done by the LabelSplitter module described in Section 3.2). Current ontology selection techniques only use syntactic matches when identifying relevant ontologies.

Step 2.2 Semantic Match. Semantic matching goes beyond the state of the art in ontology selection as it checks the soundness and the semantic nature of the previously identified syntactic mappings. Concretely, the input to this step is a term and a concept in an ontology that is lexically related to the term. The task is to find out the semantic relation between the term and the concept. This can be equivalence, more specific or more general.

An obviously relevant body of work is that on mapping techniques. However, according to a recent survey of mapping techniques [12] most matchers return a probability coefficient to describe the significance of a mapping rather than its semantic interpretation. A notable exception is the S-Match algorithm which returns the semantic category of each mapping in terms of (among others) the exact, more generic or more specific operators [5]. Following the general model of the S-Match algorithm, we distinguish two steps to obtain a semantic matching:

A. Acquiring the sense of the concept label also taking into account its position in the hierarchy.
B. Deriving the semantic relations between the term and the concept.

A. Acquiring the sense of the concept label. We use information about the position of a concept in the ontology to determine its sense according to a method originally presented in [8]. In a nutshell, given a concept c and either one of its ancestors or descendants r all WordNet synsets for both labels are retrieved. Then, if any of the senses for c is related to any of the senses of r either by being a synonym, hypernym, holonym, hyponym or a meronym, then that sense of c is considered the right one.

B. Deriving Semantic Relations. After identifying the sense of the concept, we derive semantic relations between the terms and the concepts such as equivalence, more generic or more specific. We use a WordNet based comparison between the senses of the term and that of the concept label. Therefore, equivalence is established when two terms share a synset, and more general/more specific relations are indicated when hyponym/holonym (or even meronym/holonym) relations exist between their senses. In cases when none of these relations hold we investigate weather there is any similarity at all between the terms (and return a weaker "related" relationship). For this we investigate whether there exists an allowable "is-a" path in WordNet connecting their synsets by relying on the *depth* and *common parent index (C.P.I)* measures described in [7].

Matching relations. Our previous experience in AquaLog [7] was that mapping relations is more difficult than mapping concepts. One of the reasons is that many relations are vaguely defined (a classical example is *relatedTo* which can have a variety of meanings) and therefore can have a large number, hard to automatically predict lexical variations. Also, the meaning of a relation is given by the type of its domain and its range so the precondition of a mapping between two relations is that their domain and range classes match to some extent.

Inspired by our previous work [7], we treat relations as "second class citizens" and concentrate on finding matches for the classes that denote their domain and range first. Then, if only one relation exists between these classes we adopt it as such. If more relatione exists we attempt a lexical based disambiguations of the one that is closest to the relation that we seek. An interesting case is when some relations are present in other ontologies as concepts (e.g., *hasAuthor* can be modeled as a concept *Author* in another ontology). This case is also explored.

4.4 Step3: Identifying relevant ontology combinations.

Ideally, one would expect that the selection mechanism finds a single ontology which contains all the query terms. However, in practice this is seldom the case. Most often query terms are spread over two or more ontologies. Unfortunately, previous approaches provide a set of ontologies ranked by the coverage of each *individual* ontology. Our task therefore is to identify the best combinations of ontologies that cover the query.

There are two criteria to rank ontology combinations. On one hand, the number of ontologies should be minimal. On the other hand, the number of terms that they cover should be maximal. The ultimate best is one ontology covering all terms, and the worst is a collection of ontologies each covering a single term. We are currently working on an optimal implementation of this multiple criteria optimization problem.

4.5 Step4: Generality Ranking

Due to our semantic matching, the returned concepts can be more generic or more specific than the query terms. In this step we identify the ontology combinations that are closest in terms of abstraction level to the query terms.

We are not aware of any work that directly addresses the issue of measuring the generality of an ontology or evaluating the generality of an ontology with respect to a set of terms. A recent publication investigates evaluating the generality of a document with respect to a query [14]. After concluding that most of the generality related work in the field of IR is based on statistical measures, they propose a method to compute the generality of a document with respect to a domain ontology (in that case, Mesh) by relying on the depth and proximity of the concepts in the domain ontology (i.e., the deeper and closer the concepts are in the ontology the more specific the document/query is). Generality is computed both for the query and the document and then the obtained scores are compared. The major drawbacks of this approach are that (1) it is time consuming because all terms need to be looked up in the oracle and their positions have to be computed and (2) it depends on the coverage of the used oracle.

We agree with [14] that generality computation should be based on the meaning of the terms rather than on statistical measures. Instead of computing generality both for the query and an ontology and then comparing them, we assume that the query provides the baseline and we only compute the generality deviation of the ontology from this baseline. Another simplification is that e we circumvent the use of an external oracle by reusing the generality relation between terms and concepts as established by the semantic mapping step (we consider a function *genRel* between a term and its hit which returns -1 if the concept is more specific, 0 if it is equivalent and 1 if it is more generic than the query term).

$$RD(T,O) = \frac{\sum_{i=1}^{n} |genRel(t_i,c_i)|}{n}; GS(T,O) = \sigma(\sum_{i=1}^{n} genRel(t_i,c_i))$$

Given a set of n query terms ($t_{1,n}$) and their semantically related concepts ($c_{1,n}$) we compute the relative generality ($RD(T,O)$) of the ontology/ontologies containing these concepts with respect to the query as the mean of the absolute value of the *genRel* function. We also compute the sign of the generality deviation as the sign of the sum of all the values of the *genRel* function.

4.6 Extending Semantic Match to Deal with Compound Labels

Compound labels derived in Stage III complicate semantic matching. Hereby we describe some of the problems and the solutions that we are investigating.

*A. Acquiring the sense of a **compound** concept label.* Establishing the sense of compound labels by using WordNet is difficult as WordNet does not have an extensive coverage of compound words. We are currently investigating the strategy of interpreting the meaning of compound labels in terms of logical relations that hold between the senses of their constituents (similarly to work in [8] and [5]). According to this previous work, compound labels can be interpreted as the conjunction of their constituents and according to these rules:

Rule1. Commas and coordinate conjunctions are interpreted as a disjunction;

Rule2. Prepositions like *in* and *of* are interpreted as a conjunction;
Rule3. Exclusion expressions (e.g., *except*, *but not*) translate into negation.

However, we are not convinced that all these rules are useful in the context of online ontologies. For example, only five labels returned by Swoogle contain commas, so this is just an isolated case. Also, we found that no labels contain "except" and "but not", thus making the third rule redundant.

B. Deriving Semantic Relations between ***compound*** *terms.* The limited multi-word coverage of WordNet also prohibits using it to derive semantic relations between compound labels. We investigate a solution along the lines of that presented in [5] where compound labels, after being interpreted as logical formulas, are compared with the help of a reasoner.

5 Discussion and Future Work

Taking a step back from the details of the algorithm, we believe that the key contribution of this paper is that of exploring ontology selection in a more complex context than has been done so far. Indeed, as discussed in the introduction, current selection techniques have limited themselves to support less complex, human mediated tasks. We consider that it is time to set the goals higher by integrating selection in the context of tools that automatically reuse its output.

We have analyzed the requirements of two Semantic Web tools, a question answering tool and a semantic browser, and concluded that current approaches only marginally address them. This is not a surprise in itself because these requirements raise hard to address research issues. In fact, our proposed algorithm limits itself to tackle only five of the seven requirements. These requirements indicate that selection will need to adapt techniques from currently developing research directions such as ontology evaluation, mapping and modularization.

Ontology mapping has been the focus of the proposed algorithm which balances between providing a complete, precise coverage and an acceptable performance. Our strategy is to use a self-adaptation metaphor, the algorithm adapts its complexity to the case of each query by invoking increasingly complex stages as necessary. As such, the simplest stage is just a bit more complicated than state of the art techniques, while the most complex stage raises yet unsolved research issues. The major difference from existing approaches is the emphasis on the correctness of the mapping between query terms and ontology concepts. We go beyond current techniques which exclusively rely on lexical matches by performing a semantic match. Naturally, establishing a semantic mapping at run-time without interpreting the entire ontology structure is a challenging issue by itself.

While, obviously, there are several complex issues to address, our immediate future work will concentrate on finalizing the implementation of a first prototype. In parallel, we will adapt our tools to use this selection algorithm. They will be used as a case study for evaluating selection in an automatic knowledge reuse scenario, thus paving the way towards a selection mechanism that fits the needs of the real Semantic Web.

Acknowledgements. We thank Yuangui Lei and Victoria Uren for their comments on this paper. This work was funded by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01, and the Open Knowledge and NeOn projects sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant numbers IST-FF6-027253 and IST-FF6-027595.

References

1. H. Alani and C. Brewster. Ontology Ranking based on the Analysis of Concept Structures. In *Proceedings of the Third International Conference on Knowledge Capture(K-CAP 05)*, Banff, Canada, 2005. ACM.
2. P. Buitelaar, T. Eigner, and T. Declerck. OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection. In *Proceedings of the Demo Session at the International Semantic Web Conference*. Hiroshima, Japan, 2004.
3. L. Ding, R. Pan, T. Finin, A. Joshi, Y. Peng, and P. Kolari. Finding and Ranking Knowledge on the Semantic Web. In Y. Gil, E. Motta, V.R. Benjamins, and M.A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference*, volume 3729 of *LNCS*, pages 156 – 170. Springer-Verlag GmbH, 2005.
4. M. Dzbor, J. Domingue, and E. Motta. Magpie - towards a semantic web browser. In *Proceedings of the Second International Semantic Web Conference*, 2003.
5. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. SMatch: An Algorithm and Implementation of Semantic Matching. In *The Semantic Web: Research and Applications. Proceedings of the First European Semantic Web Conference*, volume 3053 of *LNCS*, pages 61 – 75. Springer-Verlag, 2004.
6. V. Lopez, E. Motta, and V. Uren. PowerAqua: Fishing the Semantic Web. In *Proceedings of the Third European Semantic Web Conference*, 2006.
7. V. Lopez, M. Pasin, and E. Motta. AquaLog: An Ontology-portable Question Answering System for the Semantic Web. In *Proceedings of the European Semantic Web Conference*, 2005.
8. B. Magnini, L. Serafini, and M. Speranza. Making explicit the semantics hiden in schema models. In *Proceedings of the Human Language Technology for the Semantic Web workshop at ISWC'03*, 2003.
9. P. Mika. Flink: Semantic Web Technology for the Extraction and Analysis of Social Networks. *Journal of Web Semantics*, 3(2), 2005.
10. C. Patel, K. Supekar, Y. Lee, and E. K. Park. OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification. In *Proceeding of the Workshop On Web Information And Data Management*. ACM, 2003.
11. M. Sabou, V. Lopez, E. Motta, and V. Uren. Ontology Selection: Ontology Evaluation on the Real Semantic Web. In *Proceedings of the Evaluation of Ontologies on the Web Workshop, held in conjunction with WWW'2006*, 2006.
12. P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. *Journal of Data Semantics*, IV:146 – 171, 2005.
13. S. Spaccapietra. Report on modularization of ontologies. Knowledge Web Deliverable D2.1.3.1, July 2005.
14. X. Yan, X. Li, and D. Song. Document Generality: Its Computation and Ranking. In *Proceedings of the Seventeenth Australasian Database Conference*, 2006.