Book Title: Ontology Learning and Population
Book Editors: P.Buitelaar, P. Cimiano, B. Magnini
IOS Press, 2005

Learning Web Service Ontologies: an Automatic Extraction Method and its Evaluation

Marta Sabou [a,1],
[a] *Department of Artificial Intelligence, Vrije Universiteit Amsterdam*

Abstract. Successful employment of semantic Web services depends on the availability of high quality ontologies to describe the domains of these services. Building such ontologies is difficult and costly, thus hampering Web service deployment. As a solution, we describe an automatic extraction method that learns domain ontologies from textual documentations attached to Web services. We evaluate our method in two different domains, that of RDF ontology storage tools and that of bioinformatics services. The quality of the extracted ontologies is verified against high quality hand-built ontologies of the corresponding domains. We conclude that our method allows extracting a considerable amount of information for a domain ontology and that it is applicable across different domains.

Keywords. ontology learning, Web services, evaluation

1. Introduction

The promise of the emerging Semantic Web Services field is that machine understandable semantics augmenting Web services will facilitate their discovery and integration. Several projects used semantic Web service descriptions in very different application domains (bioinformatics grid [38], Problem Solving Methods [25]). A common characteristic of these descriptions is that they rely on a generic description language, such as OWL-S [10], to specify the main elements of the service (e.g., inputs, outputs) and on an ontology containing knowledge about the domain of the service such as the types of offered functionalities (e.g., *TicketBooking*, *CarRental*) or the types of service parameters (e.g., *Ticket*, *Car*). We refer to these ontologies as *domain ontologies*.

The quality of the used domain ontologies influences the complexity of reasoning tasks that can be performed with the semantic descriptions. For many tasks (e.g., matchmaking) it is preferable that Web services are described according to the same domain ontology. This implies that the domain ontology used should be *generic* enough to be used in many Web service descriptions. Domain ontologies also formally depict the complex relations that exist between the domain concepts. Such *rich* descriptions allow performing complex reasoning tasks such as flexible matchmaking. We conclude that build-

[1]Correspondence to: Marta Sabou, Vrije Universiteit, De Boelelaan 1081A, T3.12, 1081 HV Amsterdam, The Netherlands. Tel.: +31 (0)20-5987752; Fax: +31 (0)20-5987653; E-mail: Marta.Sabou@cs.vu.nl.

ing quality (i.e., generic and rich) domain ontologies is at least as important as designing a generic Web service description language such as OWL-S.

The acquisition of semantic Web service descriptions is a time consuming and complex task whose automation is desirable, as signaled by many researchers in this field, for example [38]. Pioneer in this area is the work reported in [18] which aims to learn Web service descriptions from existing WSDL[1] files using machine learning techniques. They classify these WSDL files in manually built task hierarchies. Complementary, we address the problem of building such hierarchies, i.e., Web service domain ontologies.

Despite their importance, few domain ontologies for Web service descriptions exist and building them is a challenging task. A major impediment is *the lack of guidelines* on what knowledge such ontologies should contain and what design principles they should follow. In the bioinformatics domain, for example, different communities used different approaches to build very different ontologies for semantically describing Web services [19]. Further, in order to build a generic and rich domain ontology one would ideally need to inspect a large number of Web services in that domain. Several domains witnessed a rapid increase in the number of available Web services to several hundreds (600+ in bioinformatics). Therefore tools that support ontology curators to get a quick insight in these *large and dynamic data sets* become crucial.

Our approach to the problem of building quality domain ontologies is motivated by the observations that textual sources attached to Web services (e.g., short descriptions of these services, the documentation of the code of the underlying software) (1) contain valuable information for building ontologies and that (2) they use natural language in a specific way. In fact, such texts belong to what is defined as a sublanguage in [15]. A sublanguage is a specialized form of natural language which is used within a particular domain or subject matter and characterized by a specialized vocabulary, semantic relations and syntax (e.g., weather reports, real estate advertisements). We implemented an ontology extraction method that leverages the sublanguage nature of these texts.

In what follows, we start by describing the difficulties of domain ontology building in two domains that served as case studies for applying and evaluating our method (Section 2). Then we present the ontology learning method (Section 3), some considerations about its evaluation (Section 4) and the experimental results (Section 5). We list related work in Section 6, then summarize the paper and point out future work in Section 7.

2. Two Case Studies

In this Section we describe the various difficulties in building domain ontologies in the domains of RDF(S) storage tools and bioinformatics services.

2.1. Case Study 1: RDF(S) Storage tools

In the context of two projects semantic description of ontology storage tools was required: the KAON Application Server, a middleware system which facilitates the interoperability of semantic Web tools [35], and the AgentFactory project that performs configuration of semantically described Web services using agent-based design algorithms [33].

[1] WSDL is the industry standard for syntactic Web service descriptions.

The major impediment in building a domain ontology for describing RDF(S) storage tools was that, while there are many tools offering ontology storage (a major survey [14] reported on the existence of 14 such tools), only very few are available as Web services (two, according to the same survey). Therefore, it is problematic to build a good domain ontology by analyzing only the available Web services. Our approach in this situation relied on the observation that, since Web services are simply exposures of existing software to Web-accessibility, there is a large overlap (often one-to-one correspondence) between the functionality offered by a Web service and that of the underlying implementation. Accordingly, we built a domain ontology by analyzing the APIs of three tools (Sesame [4], Jena [24], KAON RDF API [21]). The time consuming process of reading, understanding the documentation, identifying overlapping functionalities offered by the APIs of these tools and modelling them in an ontology required three weeks (for one person). The ontology contains a hierarchy of concepts denoting functionalities (e.g., *AddData*, *AddOntology*) as well as elements of the RDF Data Model (e.g., *Statement, Predicate, ReifiedStatement*) and their relations (see a snapshot in Fig. 1).

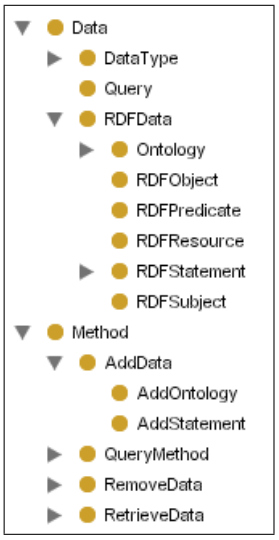

Figure 1.: RDF(S) Storage Domain Ontology Snapshot.

2.2. Case Study 2: ^{my}Grid Bioinformatics Services

^{my}Grid is a UK EPSRC e-Science pilot project building semantic grid middleware to support *in silico* experiments in biology. The experimental protocol is captured as a workflow, with many steps performed by Web services. Core to the infrastructure is an ontology for describing the functionality of these services and the semantics of the manipulated data. A key role of the ontology is to facilitate user driven discovery of services at the time of workflow construction.

Several factors hampered the building of this ontology, transforming the process into a time consuming and difficult activity. First, ontology building itself is *time consuming*. The ontology was initially built with two months of effort by an ontology expert with four years experience in building description logic based biomedical ontologies. The ontology was built manually, initially by using the documentation for 100 EMBOSS services as a source of relevant terms[2]. A second impediment was the *dynamic nature* of the field. The exponential rise in the number of bioinformatics Web services over the past year required a further two months effort to maintain and extend the ontology. However, its content currently lags behind that needed to describe the 600+ services available to the community. Thirdly, *lack of tools* hampered the process. At the time of development, tool support for handling separate ontology modules was minimal, hence the existence of one substantial ontology with distinct subsections covering the domains of molecular biology, bioinformatics, informatics and generic tasks, all under a common upper level structure.

[2]The EMBOSS (European Molecular Biology Open Software Suite) service collection is available at http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/

A fourth impediment that the ontology curator encountered was the *lack of guidelines* on how to build the domain specific ontology, or indeed how to relate it to upper level ontologies. Lacking guidance from the Web services field, the curator relied on design principles employed in other large open source biomedical ontologies such as openGALEN [31] and the TAMBIS ontology [2]. Currently only a part of this ontology (accounting for 23% of its concepts) provides concepts for annotating Web service descriptions in a forms-based annotation tool Pedro[3] and is subsequently used at discovery time with or without reasoning to power the search [38].

Summarizing Section 2, we note that the ontology building activity in both domains was hampered by the large number of documents to be analyzed, the lack of guidelines about ontology building and the lack of tools to support the process. We conclude that, with the increasing number of available Web services, (semi-)automatic tool support for ontology curators will become crucial. The extraction method presented here addresses this need for ontology building support tools. We benefitted from work done in these two case studies to evaluate the extraction method. The corpora used for the creation of the manual ontologies were used as a basis for extraction while the manually built ontologies themselves served as Gold Standards to evaluate the extracted ontologies.

3. The extraction process

Software documentation in general, and Web service descriptions in particular, employ natural language in a specific way. Our extraction method exploits the syntactic regularities which are inherent from the sublanguage nature of Web service documentations. Note that in many cases, for example as demonstrated by the presented case studies, the Web service descriptions also reflect the particularities of the domain sublanguage, for example bioinformatics in the second case study. Therefore, characteristics of the domain sublanguage can influence the extraction process.

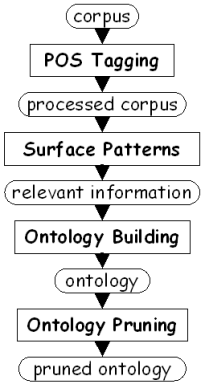

Figure 2.: The Extraction process.

The ontology extraction consists of several steps, as depicted in Figure 2. During the first step of the extraction we annotate the corpus with Part of Speech (POS) linguistic information. In the second step, a set of *surface patterns* are applied on this linguistic knowledge to identify potentially interesting information for ontology building. The next step, ontology building, transforms the extracted relevant information into ontological constructs. Finally, a pruning step excludes potentially uninteresting concepts from the ontology. In what follows we detail the last three extraction steps.

3.1. Surface Patterns.

A set of patterns identify potentially interesting information for ontology building. They are surface patterns because, besides the POS tag linguistic information, they rely on surface knowledge such as the position of words in the sentence. We distinguish two major categories of patterns used to derive different types of information.

[3] http://pedrodownload.man.ac.uk/

1. Identifying domain concepts. Domain concepts are depicted by the nouns in a corpus. We extract entire noun phrases where a noun phrase consists of a head noun preceded by an arbitrary (0 or more) number of nouns or adjectives known as its modifiers. To perform this extraction we used JAPE [12], a rich and flexible regular expression based rule mechanism. For example, this is the JAPE rule that identifies noun phrases:

```
( (DET)*
  (ADJ|NOUN|POS)*
  (NOUN) ):np
-->:np.NP={}
```

The pattern in the left hand side of the rule identifies all sequences of words starting with 0 or more determiners (e.g., the, a), 0 or more adjectives, nouns or possession indicators in any order and mandatorily finishing with a noun. DET, ADJ, NOUN and POS are placeholders for other rules identifying words that are part of these categories. The left hand side of the rule annotates the identified sequence as a noun phrase (NP).

2. Identifying functionalities. Besides domain concepts, a domain ontology used for Web service descriptions needs to specify types of functionalities that are frequently offered in that domain. We observed that, in the majority of cases, verbs identify the functionality performed by a service and nouns following these verbs refer to data structures that are involved in some way in that functionality. Therefore our extraction pattern identifies verbs and following noun phrases as potential information to be added to the domain ontology. Having identified and annotated noun phrases (NP) and verbs (VB) with two previous rules, the JAPE rule for identifying and annotating verbs and following noun phrases is straightforward.

```
( {VB}{NP} ):funct
-->:funct.Functionality = {}
```

3.2. Ontology Building

The ontology building step collects the results of the previous pattern based extraction. The extracted terms are used for building two different aspects of the domain ontology. Noun phrases are a basis for deriving a data structure hierarchy and the functionality information is used for building a functionality hierarchy. Lemmatization of the words returned by the patterns is performed before the terms are used for ontology building.

Building the data structure hierarchy. We observed that many of the terms mentioned in the analyzed corpora (and especially in the bioinformatics corpus) expose a high level of compositionality, in the sense that they incorporate other meaningful terms as proper substrings. Our observation is backed up by a recent study of the Gene Ontology terms which proved that 63,5% of all terms in this domain are compositional in nature [28]. Another observation, also proven by this study, is that compositionality indicates the

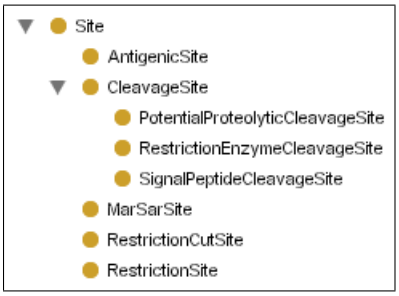

Figure 3.: The Site concept.

existence of a semantic relationship between terms. Namely, a term A contained as a proper substring of a term B is more generic than term B. This translates in the ontological subsumption relationship. The hierarchy building algorithm reflects our observations. If a concept A's lexicalization is a proper substring of another concept B's lexicalization (e.g., *Site* in *AntigenicSite*) then A is more generic than B and the corresponding subsumption relationship is added to the ontology. Also, if the lexicalization of two concepts B and C contain a common substring we speculate that this substring represents a valid domain concept (even if the string does not appear as a stand alone term in the corpus) and add it as a parent concept of B and C. As an illustration, Figure 3 depicts the data structure hierarchy for the *Site* concept. Such compositionality based hierarchy building has also been used in other ontology learning approaches ([6,37]).

Building the functionality hierarchy. There are no clear guidelines in the field of semantic Web services about how functionality hierarchies should look like. Major semantic Web initiatives such as OWL-S [10], IRS [25] and WSMO[4] model functionalities by including both the verb of the action and a directly involved data element in the functionality (e.g., *BuyTicket*). This modelling style was followed in case study 1. On the other hand, in the bioinformatics domain ontology from case study 2, functionality concepts denote action (e.g., *Aligning*) without mentioning the involved data structures. We provide ontology building modules that produce functionality hierarchies fulfilling either of these two modelling styles.

3.3. Ontology Pruning

The pruning module filters out irrelevant concepts from the extracted ontologies. We employ a *baseline pruning* strategy which advocates that frequent terms in a corpus denote domain concepts while less frequent ones lead to concepts that can be safely eliminated from the ontology [20]. We consider the average frequency of the terms as a threshold value and prune all concepts that have a lower frequency than this value. Another heuristic for the pruning is based on the observation that noun phrases included within a Functionality annotation by our rules are more likely to denote domain concepts. Therefore, if a low frequency data structure concept's lexicalization was identified within a Functionality annotation and the corresponding functionality concept was not pruned then the data structure concept will not be pruned either.

3.4. Implementation

The extraction method[5] is implemented using the GATE [11] framework.

The linguistic preprocessing was entirely performed using processing resources offered by GATE: a tokenizer, a sentence splitter and the Hepple POS tagger [17]. We implemented our extraction patterns using the JAPE mechanism which is part of the framework. Other modules of our method (e.g., Ontology Building) were declared as GATE Processing Resources. The data used by our method (such as the linguistic knowledge or the structures identified by patterns) is represented as *annotations* on the analyzed texts. Both patterns and individual modules operate on these anno-

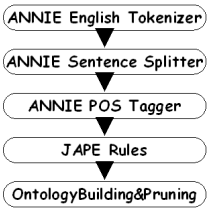

Figure 4.: The GATE implementation.

[4]http://www.wsmo.org

[5]The implementation can be downloaded from http://www.cs.vu.nl/~marta/experiments/extraction.html.

tations and represent their output as annotations. We greatly benefitted from the support of GATE to (partially) perform the evaluation of our prototype by using its *data storage* and *evaluation* facilities.

4. Evaluation criteria

Evaluation of ontology learning is an important but largely unsolved issue, as reported at the workshop [5] upon which this volume is based. Two evaluation stages are typically performed when evaluating an ontology learning method. First, *term level evaluation* assesses the performance of extracting domain relevant terms from the corpus. Second, an *ontology quality evaluation* stage assesses the quality of the extracted ontology.

While term level evaluation can be performed by using the well-established recall/precision metrics, ontology quality evaluation is more subtle and there is no standard method for performing it. One approach is to compare an automatically extracted ontology with a Gold Standard ontology which is a manually built ontology of the same domain ([32] and in this volume), often reflecting the knowledge existing in the corpus used for the extraction ([9]). The goal of this approach is to evaluate the degree to which the ontology covers the analyzed domain. Another approach is to evaluate the appropriateness of an ontology for a certain task. Initial experiments with such a task-based ontology evaluation are reported in [30] (and in this volume). As a third approach, a per-concept evaluation by a domain expert is used in [27] (and in this volume). We consider that all these types of evaluation, depicted in Fig. 5, cover important and complementary aspects such as:

1. *What is the performance of the learning algorithm? - extraction performance*
2. *Is the extracted ontology a good basis for ontology building? - expert evaluation*
3. *Does the extracted ontology cover the analyzed domain? - domain coverage*
4. *Does the extracted ontology support a certain task? - appropriateness for a task*

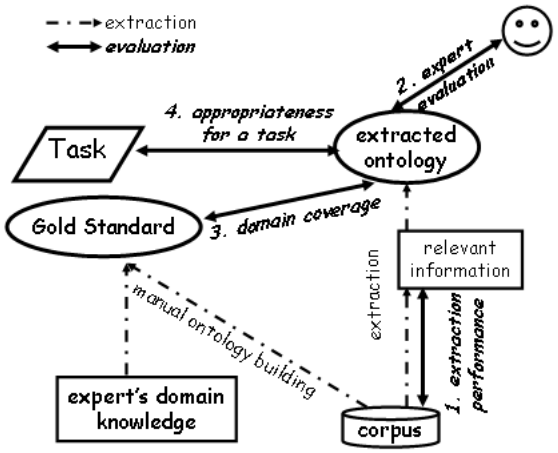

Figure 5. Evaluation Strategies.

In our case studies we perform three out of the four types of evaluation. We first perform a term level evaluation (1). Then, we rely on the domain experts' concept per concept based evaluation of the ontologies to conclude their usefulness for supporting the ontology building task (2). The domain experts in both cases are the curators of the corresponding Gold Standard ontologies. Finally, we compare the extracted ontologies to the corresponding Gold Standards to assess their domain coverage (3). In what follows, we present the methodology and metrics for performing each type of evaluation.

1. Extraction Performance. To measure the performance of the extraction modules we manually identified all the relevant terms to be extracted from the corpus. Misspelled terms were not considered for extraction. Then, using the Benchmark Evaluation Tool offered by GATE, we compared this set of terms with the ones that were identified through pattern based extraction. We use Term Recall (TRecall) to quantify the ratio of (manually classified) relevant terms that are extracted from the analyzed corpus ($correct_{extracted}$) over all terms to be extracted from the corpus (all_{corpus}). Term Precision (TPrecision) denotes the ratio of correctly extracted terms over all extracted terms ($all_{extracted}$).

$$TRecall = \frac{correct_{extracted}}{all_{corpus}} \quad ; \quad TPrecision = \frac{correct_{extracted}}{all_{extracted}}$$

2. Expert Evaluation. Assessing whether the extracted ontology offers a useful basis for building a domain ontology is important since the main goal of our research is supporting the ontology building task. During the concept per concept analysis of the extracted ontologies the domain experts rated concepts $correct$ if they were useful for ontology building and were already included in the Gold Standard. Concepts that were relevant for the domain but not considered during manual ontology building were rated as new. Finally, irrelevant concepts, which could not be used, were marked as $spurious$. The higher the ratio between all relevant concepts and all concepts of the ontology, the better it supports ontology building. We express this ratio as ontology Precision ($OPrecision$):

$$OPrecision = \frac{correct + new}{correct + new + spurious}$$

Very useful side effects of the expert evaluation were the opinion and suggestions of the experts. They provided valuable ideas for further improvements.

3. Domain Coverage. We evaluate domain coverage by comparing the extracted ontologies to the corresponding Gold Standard ontologies. According to [22], one of the few works on measuring the similarity between two ontologies, one can compare two ontologies at two different levels: lexical and conceptual. Lexical comparison assesses the similarity between the lexicons (set of terms denoting concepts) of the two ontologies. At the conceptual level the taxonomic structures and the relations in the two ontologies are compared. In this paper we only perform a lexical comparison of the two ontologies.

Our first metric denotes the shared concepts between the manual and extracted ontology. This metric was originally defined in [22] as the *relative number of hits* (RelHit), then renamed in [9] to Lexical Overlap (LO). Let L_{O_1} be the set of all *domain relevant* extracted concepts (which is a subset of all extracted concepts rated $correct$ and new by the domain expert) and L_{O_2} the set of concepts of the Gold Standard ontology. The lexical overlap is equal to the ratio of the number of concepts shared by both ontologies

(i.e., the intersection of these two sets - also noted *correct*) and the number of all Gold Standard ontology concepts (noted *all*).

$$LO(O_1, O_2) = \frac{|L_{O_1} \cap L_{O_2}|}{|L_{O_2}|} = \frac{correct}{all} \quad ; \quad OI(O_1, O_2) = \frac{|L_{O_1} \setminus L_{O_2}|}{|L_{O_2}|} = \frac{new}{all}$$

It is interesting to analyze the concepts that are not part of this overlap. Often, the extracted ontology can bring important additions to the manual ontology by highlighting concepts that were ignored during manual creation. We are not aware of any previously defined metric for measuring these additions. Therefore we define Ontological Improvement (OI) as the ratio between all domain relevant extracted concepts that are not in the Gold Standard ontology (noted *new*) and all the concepts of the Gold Standard ontology.

5. Experiments

5.1. Experimental setup

We applied the extraction method in the context of both case studies[6]. The ontology building algorithm was adjusted to follow the modelling principle employed by each Gold Standard ontology. We then evaluated the extraction results using the criteria discussed in the previous section. All evaluations were performed for both the DataStructure and Functionality part of the extracted ontologies as well as on the ontologies as a whole. In order to get an insight in the efficiency of our pruning heuristics we evaluated both a pruned and an un-pruned version of the extracted ontologies. By comparing the performance of the extraction method in the two case studies we wish to gain an insight in its applicability on data sets from different domains and of different level of complexity.

5.2. Corpora

Case study 1: RDF(S) Storage Tools. The first corpus, *Corpus 1*, contains the documentation of the tools used to build the manual ontology (Jena, KAON RDF API, Sesame) and has 112 documents. Each document in the corpus contains the javadoc description of one method. Previous work showed that the short textual descriptions of these methods contain the most information and other javadoc elements such as the method syntax and the description of the parameters introduce a lot of noise severely diminishing the performance of the extraction [34].

Case Study 2: Bioninformatics services. Our experimental corpus for this domain (*Corpus 2*) consisted of 158 individual bioinformatics service descriptions as available at the EMBOSS Web site[7]. We worked only on the short method descriptions since they are significant for Web service descriptions in general being very similar to descriptions found in online Web service repositories as SalCentral[8] and XMethods[9]. The detailed descriptions of the EMBOSS services present a specific layout which makes extraction

[6] All experimental data (corpora, extracted and gold standard ontologies) can be downloaded from http://www.cs.vu.nl/~marta/experiments/extraction.html.

[7] http://www.hgmp.mrc.ac.uk/Software/EMBOSS/Apps/

[8] http://www.salcentral.com

[9] http://www.xmethods.net

much easier however using it would have biased our extraction methods towards this particular kind of documentation.

5.3. Results

1. Extraction Performance.

The results of the first evaluation step, depicted in Table 1, indicate that we can extract the desired lexical information with a good confidence from both corpora. We observed that errors are mostly due to mistakes in the POS tagger's output. Most often verbs at the beginning of the sentence are mistaken for nouns causing a lower recall for functionality pairs compared to the recall of data structures. A second source for errors are spelling and punctuation mistakes in the corpora. *Corpus 1* has, from this perspective, a lower quality than *Corpus 2* and, indeed, this affects the Term Precision.

	Corpus 1			***Corpus 2***		
	Data	**Functionality**	**All**	**Data**	**Functionality**	**All**
TRecall	0.8	0.66	**0.77**	0.85	0.44	**0.71**
TPrecision	0.71	0.83	**0.73**	0.78	0.94	**0.81**

Table 1. Results of quantitative evaluation.

2. Expert Evaluation.

The results for the expert evaluation for the two case studies are depicted in Table 2 and Table 3 both for the ontology as originally extracted and its pruned version. Observe that the value of the term precision has a direct influence on the value of the ontology precision. A term precision value of 0.73 for the first corpus resulted in an ontology precision value of only 0.36, while for the second corpus a slightly higher term precision (0.81) resulted in an almost double value for ontology precision (0.63). Within both ontologies, the same influence can be observed for the data and functionality parts of the ontology: the data part has a lower extraction performance than the functionalities, in both cases, and this leads to lower ontology precisions.

The pruning mechanism increased the ontology precision in both cases: in the first case study this value almost doubled, and in the second case it increased with 10 units leading to precisions over 0.6. This means that more than half of the concepts of both pruned ontologies are relevant for the analyzed domain.

	Not Pruned			**Pruned**		
	Data	**Functionality**	**All**	**Data**	**Functionality**	**All**
OPrecision	0.28	0.64	**0.36**	0.51	0.75	**0.60**
LO	0.79	0.39	**0.5**	0.57	0.39	**0.44**
OI	3.14	0.72	**1.4**	2.07	0.5	**0.94**

Table 2. Ontology building support and ontology comparison for the RDF(S) case study.

Besides our quantitative results, we derived several interesting observations from the comments of the domain experts.

Recall vs. Precision. It seams that the cleanness of the ontology is not of major importance for the ontology engineer. Often even concepts that are not included in the final ontology are useful to give an insight in the domain itself and to guide further abstraction

activities. We should therefore concentrate to increase the recall of our extraction process even at the expense of precision.

Synonymy. During the evaluation, the expert recognized several potential synonym sets such as: *{find, search, identify, extract, locate, report, scan}, {fetch, retrieve, return}, {pick, select}, {produce, calculate}* or *{reverse, invert}*. Synonymy information is an important piece of knowledge for semantic Web services. Especially search and matchmaking algorithms would benefit from knowing which concepts are equivalent. Automatic acquisition of synonymy information remains an important are of future work.

Abstractions. The experts often redistributed the extracted domain concepts according to their domain view. For example, two subclasses identified for *Protein* belong to different domains, molecular biology and bioinformatics, and have to be placed in the corresponding hierarchies accordingly. Such abstractions need to be still manually created according to the ontology engineers view on the domain. However, the abstraction step is considerably supported if the expert has an overview of relevant domain concepts.

Support. The curators considered the extracted ontologies as a useful start for deriving a domain ontology. Several complex structures could be included in a final ontology without further changes (e.g., the *Site* hierarchy in Figure 3), or provided helpful hints about how certain concepts interrelate. The most appreciated contribution was that the learned ontologies even suggested new additions for the manually built ontologies.

3.Domain Coverage

The unpruned RDF(S) ontology identified half of the concepts existing in the manually built ontology and many new concepts (see Table 2). Lexical overlap and ontological improvement registered higher values for the DataStructure part of the ontology probably as a direct result of the higher extraction recall for these elements from the corpus than for functionality terms. The pruning mechanism behaved optimally in this case. While it almost doubled the ontology precision (from 0.36 to 0.6) it only slightly affected the lexical overlap (from 0.5 to 0.44). As expected, ontological improvement was more affected (from 1.4 to 0.94) because many of the newly identified concepts have a low domain relevance. In this case our heuristic distinguishes between important domain relevant concepts and less important concepts.

	Not Pruned			**Pruned**		
	Data	**Functionality**	**All**	**Data**	**Functionality**	**All**
OPrecision	0.59	0.87	**0.63**	0.71	0.94	**0.74**
LO	0.19	0.26	**0.2**	0.08	0.13	**0.1**
OI	1.15	0.96	**1.12**	0.51	0.52	**0.51**

Table 3. Ontology building support and ontology comparison for the Bioinformatics case study.

The comparison with the bioinformatics application ontology registered less success than in the previous case study (see Table 3). The unpruned ontology covered only 20% of the manual ontology, even if it suggested many new possible concepts. Pruning was less optimal in this case: it reduced both the overlap and the improvement to half while resulting in a low precision increase (from 0.63 to 0.74). One of the major reasons for this behavior is that, in bioinformatics, due to the compositionality of terms, deep DataStructure concept hierarchies are created where the frequency of the concepts decreases with their generality. These low frequency specialized concepts were pruned even if they are important. To avoid this, the pruning threshold should be decreased as advancing deeper

into the hierarchy. Also, since the precision of the ontology was already high without pruning, we might have adopted a lower value for the pruning threshold as a whole.

Our results suggest that the ontology curator worked rather independently from the given corpus during the building of the Gold Standard as he missed many concepts named in the corpus. Post-experimental interviewing of the curator revealed that the examination of the corpus was not meticulous. He used just in time ontology development: concepts were added if needed for describing a certain service. Note also that he worked on a subset of our analyzed corpus (100 service descriptions instead of 158 analyzed by us). Name changes could also account for the mismatch. The curator expanded abbreviations or created a preferred term for several synonyms (e.g., *Retrieving* for fetch, get, return). He acknowledged that the automatic approach leads to a more faithful reflection of the concepts the community uses to describe their services. The major cause for ontological losses was that the curator also included concepts about the fields of biology, bioinformatics and informatics that are not present in the corpus (see Section 2).

5.4. Conclusions

The evaluation suggests that the extraction method is domain independent (performing similarly in two different domains) even if it is influenced by particularities of the analyzed corpora. In both case studies the method (1) efficiently extracts important terms from the corpus and (2) builds ontologies containing a majority of domain relevant concepts. Punctuation and spelling mistakes lead to a low extraction precision, and consequently, to a less precise ontology. The extracted ontologies (3) contain a fair part of the manually identified concepts and (4) suggest a lot of new additions complementing manual ontology building that tends to ignore much of this knowledge. The pruning mechanism (5) performs less optimally in the case of the second corpus suggesting that cleaner corpora need a lower pruning threshold.

6. Related Work

The problem of automating the task of Web service semantics acquisition was addressed by the work of two research teams. Hess and Kushmerick [18] employ the Naive Bayes and SVM machine learning algorithms to classify WSDL files (or Web forms) in manually defined task hierarchies. Our work is complementary, since we address the acquisition of such hierarchies. Also, our method does not rely on any manually built training data as the machine learning techniques do. Patil et al. [29] employ graph similarity techniques to determine a relevant domain ontology for a WSDL file and to annotate its elements. Currently they determine the semantics of the service parameters and plan to concentrate on functionality semantics in the future. They use existing domain ontologies and acknowledge that their work was hampered by the lack of these ontologies.

The ontology learning field offers a wide range of different approaches to ontology acquisition. While most work is targeted on specific domains we are not aware of any efforts that analyze software documentation style texts. Several generic ontology learning tools exist, namely Text-To-Onto [23], OntoLearn [26] or OntoLT [6]. We tried to extract a domain ontology from our corpora using Text-to-Onto, the only tool publicly available at the time of the experiments. The results were suboptimal due to the strong particular-

ities of our corpora which hampered the efficiency of the generic methods implemented by the tool. In contrast, our method is tailored for dealing with Web service descriptions while, as our experiments prove in this paper, it is applicable across domains.

Pattern based term extraction is a core part of our method. Pattern based techniques have been first used to derive semantic relations from large corpora. A pioneer in this direction of research was the work of Hearst which introduced the idea of learning ***hyponymy*** relations using **lexico-syntactic patterns**. Lexico-syntactic patterns are defined on both lexical and basic syntactic information (POS tags). As such, they allow extracting relations after shallow text processing only. For example, the hyponymy relationship suggested by *Bruises, wounds, broken bones or other injuries* could be extracted using the *NP, NP*, or other NP* pattern [16]. As a follow up of this work, Charniak developed a set of lexico-syntactic patterns that identify ***meronymy*** (partOf) relations [3].

Naturally, such patterns have a clear relevance for ontology learning. Indeed, Hearst-style patterns have been used in the work of Cimiano [8] or in the CAMELEON tool which incorporates over 150 generic patterns for the French language [36,1]. While such generic patterns work well in general corpora they often fail in small or domain specific corpora. In these cases domain-tailored patterns provide a much better performance [1]. Besides using domain tailored patterns one can enlarge the extraction corpora. For example, World Wide Web data can be used for pattern based learning [7]. In several ontology learning approaches, as in our work, pattern based extraction is just a first step in a more complex process [32,13,6]. In these cases patterns identify potentially interesting terms in the corpus and the next processing steps derive relevant semantic structures from them.

7. Summary and Future Work

While domain ontologies are of major importance for semantic Web services, their acquisition is a time consuming task. We have developed an extraction method that relies on the sublanguage characteristics of textual sources attached to Web services to extract such domain ontologies. We have evaluated the ontology learning algorithm in two different domains and concluded that it is applicable across different domains. The experts indicate that the extracted ontologies represent more faithfully the knowledge in the corpus and that they provide a useful start for building a Web service domain ontology.

One could imagine the use of our method in several application scenarios. We believe it is useful for providing a first insight into a domain and providing a sketch ontology that can be the basis for ontology construction. During our experiments it turned out that such a tool is also useful when an ontology already exists in a domain because it can suggest new additions that were ignored during manual ontology building. Also, as the domain changes such a method can help to discover newly available domain knowledge and to integrate it in the existing ontology. Finally, it can be the basis of harmonization efforts when two existing ontologies are merged by being applied on the union of two different corpora that underlaid the creation of each ontology.

As future work we wish to extend the method with more fine-grained extraction patterns to complement the current high coverage patterns. We can also enhance the step of ontology building. Use of synonymy information during the conceptualisation step is an important development possibility. We would also like to concentrate on strategies that enrich the basic extracted ontology. For example defining different views on a set

of domain concepts or providing a set of patterns that act on the extracted semantic information. Yet another development dimension is the use of external knowledge sources (such as WordNet for synonymy detection or WSDL files for input/ output information) to complement the small corpora. Further, the usability of the extraction tool could be enhanced by providing auxiliary information for each concept such as text segments in the corpus where it appears and the frequency of its appearance.

Acknowledgments. We thank C. Goble and C. Wroe for providing us with the *my*Grid experimental data and evaluating the extracted ontology. We thank the members of the GATE group for their support in the development of our prototype.

References

[1] N. Aussenac-Gilles. Supervised text analyses for ontology and terminology engineering. In *Proceedings of the Dagstuhl Seminar on Machine Learning for the Semantic Web*. 2005.

[2] P.G. Baker, C.A. Goble, S. Bechhofer, N.W. Paton, R. Stevens, and A. Brass. An Ontology for Bioinformatics Applications. *Bioinformatics*, 15(6):510–520, 1999.

[3] M. Berland and E. Charniak. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, 1999.

[4] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In I. Horrocks and J. A. Hendler, editors, *Proceedings of the First International Semantic Web Conference*, LNCS, Sardinia, Italy, 2002.

[5] P. Buitelaar, S. Handschuh, and B. Magnini. ECAI Workshop on Ontology Learning and Population: Towards Evaluation of Text-based Methods in the Semantic Web and Knowledge Discovery Life Cycle. Valencia, Spain, August 2004.

[6] P. Buitelaar, D. Olejnik, and M. Sintek. A Protege Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. In *Proc. of European Semantic Web Symposium (ESWS)*, 2004.

[7] P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In *Proceedings of the 13th World Wide Web Conference*, May 2004.

[8] P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning Taxonomic Relations from Heterogeneous Evidence. In *Proceedings of ECAI2004 Workshop on Ontology Learning and Evaluation*, 2004.

[9] P. Cimiano, S. Staab, and J. Tane. Automatic Acquisition of Taxonomies from Text: FCA meets NLP. In *Proceedings of the ECML/PKDD Workshop on Adaptive Text Extraction and Mining, Cavtat–Dubrovnik, Croatia*, 2003.

[10] The OWL-S Services Coalition. OWL-S: Semantic Markup for Web Services. White Paper. http://www.daml.org/services/owl-s/1.0/owl-s.pdf, 2003.

[11] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[12] H. Cunningham, D. Maynard, and V. Tablan. JAPE: a Java Annotation Patterns Engine (Second Edition). Research Memorandum CS–00–10, Department of Computer Science, University of Sheffield, November 2000.

[13] D. Faure and C. Nedellec. ASIUM:learning subcategorization frames and restrictions of selection. In Yves Kodratoff, editor, *Proceedings of Workshop on Text Mining, 10th European Conference on Machine Learning (ECML 98)*, 1998.

[14] A. Gomez Perez. A survey on ontology tools. OntoWeb Delieverable 1.3, 2002.

[15] R. Grishman and R. Kittredge, editors. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*. Lawrence Erlbaum Assoc., Hillsdale, NJ, 1986.

[16] M.A. Hearst. Automatic Acquisition of Hyponyms in Large Text Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, 1992.

[17] M. Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.
[18] A. Hess and N. Kushmerick. Machine Learning for Annotating Semantic Web Services. In *AAAI Spring Symposium on Semantic Web Services*, March 2004.
[19] P. Lord, S. Bechhofer, M.D. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying Semantic Web Services to bioinformatics: Experiences gained, lessons learnt. In *Proceedings of the Third International Semantic Web Conference*, 2004.
[20] A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.
[21] A. Maedche, B. Motik, and L. Stojanovic. Managing Multiple and Distributed Ontologies in the Semantic Web. *VLDB Journal*, 12(4):286–302, 2003.
[22] A. Maedche and S.Staab. Measuring similarity between ontologies. In *Proceedings of European Knowledge Acquisition Workshop (EKAW)*. Springer, 2002.
[23] A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems. Springer, 2004.
[24] B. McBride. Jena: A Semantic Web Toolkit. *IEEE Internet Computing*, 6(6):55–59, 2002.
[25] E. Motta, J. Domingue, L. Cabral, and M. Gaspari. IRS-II: A Framework and Infrastructure for Semantic Web Services. In *Proceedings of the Second International Semantic Web Conference*, LNCS. Springer-Verlag, 2003.
[26] R. Navigli and P. Velardi. Learning Domain Ontologies from Document Warehouses and Dedicated Websites. *Computational Linguistics*, 30(2), 2004.
[27] R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri. Quantitative and Qualitative Evaluation of the OntoLearn Ontology Learning System. In *ECAI Workshop on Ontology Learning and Population*, Valencia, Spain, August 2004.
[28] P.V. Ogren, K.B. Cohen, G.K. Acquaah-Mensah, J. Eberlein, and L. Hunter. The Compositional Structure of Gene Ontology Terms. In *Proceedings of the Pacific Symposium on Biocomputing*, 2004.
[29] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. METEOR-S Web service Annotation Framework. In *Proceeding of the 13th World Wide Web Conference*, 2004.
[30] R. Porzel and R. Malaka. A Task-based Approach for Ontology Evaluation. In *ECAI Workshop on Ontology Learning and Population*, Valencia, Spain, 2004.
[31] A.L. Rector and Rogers J.E. Ontological issues in using a description logic to represent medical concepts: Experience from GALEN. *IMIA Working Group 6 Workshop*, 1999.
[32] M.L. Reinberger and P. Spyns. Discovering Knowledge in Texts for the Learning of DOGMA-Inspired Ontologies. In *ECAI Workshop on Ontology Learning and Population*, Valencia, Spain, August 2004.
[33] D. Richards, S. Splunter, F. M.T. Brazier, and M. Sabou. Composing Web Services using an Agent Factory. In *Workshop on Web Services and Agent-Based Engineering, AAMAS03*, Melbourne, Australia, July 14/15 2003.
[34] M. Sabou. From Software APIs to Web Service Ontologies: a Semi-Automatic Extraction Method. In *Proceedings of the Third International Semantic Web Conference*, 2004.
[35] M. Sabou, D. Oberle, and D. Richards. Enhancing Application Servers with Semantics. In *Proceedings of AWESOS Workshop*, Australia, April 2004.
[36] P. Seguela and N. Aussenac-Gilles. Extraction de relations semantiques entre termes et enrichissement de modeles du domaine. In *Actes de la Conference IC'99 - Plate-forme AFIA*.
[37] P. Velardi, M. Missikoff, and P. Fabriani. Using Text Processing Techniques to Automatically enrich a Domain Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems*, pages 270–284. ACM Press, 2001.
[38] C. Wroe, C. Goble, M. Greenwood, P. Lord, S. Miles, J. Papay, T. Payne, and L. Moreau. Automating Experiments Using Semantic Data on a Bioinformatics Grid. *IEEE Intelligent Systems*, 19(1):48–55, 2004.