OntoCase-Automatic Ontology Enrichment Based on Ontology Design Patterns*

Eva Blomqvist

StLab, ISTC-CNR, via Nomentana 56, 00161 Roma, Italy
eva.blomqvist@istc.cnr.it

Abstract. OntoCase is a framework for semi-automatic pattern-based ontology construction. In this paper we focus on the retain and reuse phases, where an initial ontology is enriched based on content ontology design patterns (Content ODPs), and especially the implementation and evaluation of these phases. Applying Content ODPs within semi-automatic ontology construction, i.e. ontology learning (OL), is a novel approach. The main contributions of this paper are the methods for pattern ranking, selection, and integration, and the subsequent evaluation showing the characteristics of ontologies constructed automatically based on ODPs. We show that it is possible to improve the results of existing OL methods by selecting and reusing Content ODPs. OntoCase is able to introduce a general top structure into the ontologies, and by exploiting background knowledge the ontology is given a richer overall structure.

1 Introduction

Ontology engineering, for the Semantic Web and other application fields, is a tedious and error-prone process requiring expertise in knowledge modelling and logical languages. With the emergence of the Semantic Web we have seen an increase in popularity of light-weight ontologies that provide just a bit of formal semantics to a data set. Additionally, general web developers have become more and more interested in ontology engineering. To exploit the full benefits of the Semantic Web, ontologies need to be easy to construct, perhaps automatically. In line with this, we have focused on automatic methods (so called ontology learning - OL) for light-weight ontology construction, and especially how the ontology quality can be improved by applying ontology design patterns (ODPs).

Typically, learnt ontologies are quite shallow (in a taxonomical sense), sparse (with respect to the number of relations) and contain a large set of unconnected concepts. An inherent problem when OL is attempted based on text corpora is the fact that most information is actually implicit in the text

* This research was primarily conducted at the Information Engineering research group, Jönköping University (SE). Financially supported through projects SEMCO (KK-foundation grant no. 2003/0241), MediaIlLOG (grant from Carl-Olof och Jenz Hamrins Stiftelse), and DEON (STINT). Evaluations were also conducted within the NeOn project, funded by the European Commission, grant no. IST-2005-027595.

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 65–80, 2009.

(e.g. as observed in [1]). The consequence is that learnt ontologies lack an overall general structure representing this background knowledge. We believe that some of this knowledge can be added by means of ODPs, in particular Content ODPs. Such patterns can also assist in structuring the existing learnt knowledge.

In this paper we show a particular method for enriching learnt ontologies, by means of Content ODPs. The method is not language dependent as such, but is implemented based on current de facto standards, e.g. OWL. Evaluations attempt to show the two main features of this method: 1) The ability to add a (taxonomical) top structure to the ontology, representing background knowledge implicit in the texts that were the basis of the input ontology. 2) The ability to add more structure to the learnt ontology, i.e. to increase the taxonomical depth and the relation-to-concept ratio. In the following sections we present some background and related work with respect to OL and ODPs. In section 3 we discuss the OntoCase methods, and section 4 presents experiments validating the OntoCase approach. We conclude and discuss future work in section 5.

1.1 Ontology Learning

So far OL approaches have largely dealt with element extraction (see overviews [2,3]), i.e. extraction of single concepts or relations from text corpora, such as in [4]. Approaches have mostly focused on adapting techniques from NLP, computational linguistics, machine learning, and text mining. An example of a state-of-the-art OL tool is Text2Onto[1] [5]. A few recent approaches, e.g. [6,7], have attempted to extract complex axioms, but primarily based on structured input, such as dictionary entries. An approach sharing our goal of providing a more structured output is [8]. The approach uses semantic frames, from linguistics, to extract knowledge from text and transform the frames into small ontologies.

Even though many of the techniques used in OL have been around for many years, they remain subjects of research. The quality of the ontologies is far from perfect; without manual revision the ontologies are not directly usable. An important issue is improving the output ontology quality of OL systems, i.e. the main focus of OntoCase, but still we are not attempting to replace the ontology engineers, merely provide a better starting point for further development.

1.2 Content Ontology Design Patterns

There exist different types of ODPs having different characteristics, for details on ODP types see [9], but in this paper we focus on Content ODPs. Content ODPs are small ontologies with explicit documentation of design rationales, which can be used as building blocks in ontology design, as shown in [10,11].

As an example we describe a Content ODP that is called *Action*. It represents the relations between different types of actions, the state of the actions, and

[1] http://ontoware.org/projects/text2onto/

defines the relation between a plan and a set of proposed actions, see Fig. 1. Content ODPs are collected and presented in different catalogues, such as the *ODP portal*[2]. In addition to their diagrammatic representation Content ODPs are described using a number of catalogue entry fields (c.f. software pattern templates), such as *name*, *intent*, *consequences*, and *building block* (linking to an OWL realization of the pattern). Reusing Content ODPs is a special case of ontology reuse, when the elements of the Content ODP are specilized.

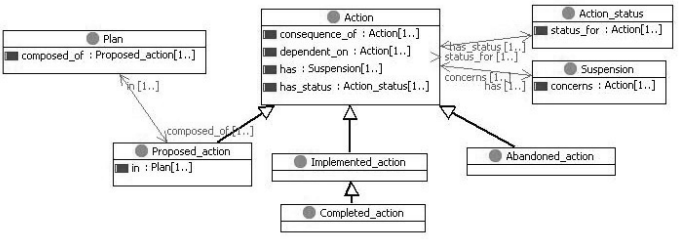

Fig. 1. The Action Content ODP's graphical representation in UML

2 Related Work

Our research is inspired by early AI approaches in analogical reasoning, e.g. scripts and frames, and case-based reasoning (CBR), see [12] for a discussion related to CBR, as well as earlier work in knowledge engineering (KE) and reuse of problem-solving methods. However, recent developments in the KE and Semantic Web fields have led to a situation where a lot of general background knowledge and other knowledge resources, such as ODPs, are now readily available. Consequently we are now able to realize the true potential of such techniques.

Within the ontology engineering field, this research is strongly related to ontology reuse, since Content ODPs are in essence small ontologies. However, no previous approaches specifically target the automatic selection and integration of Content ODPs. Related work exists in ontology search and ranking, e.g. search engines such as Watson[3], SWOOGLE[4], and Sindice[5]. Tools such as the NeOn toolkit in combination with the Watson plug-in let a developer integrate parts of the retrieved ontologies into the one being built, but the matching is simple keyword matching. The problems of selecting the right ontology, e.g. Content ODP, specializing it, and composing several ontologies are largely unaddressed.

An elaborate ranking approach that has inspired our research is AktiveRank [13]. However, this approach incorporates measures such as the centrality of the keywords in the ontology, which are not suitable for small patterns where the notion of centrality is not applicable. The methods in this paper are similar

[2] http://www.ontologydesignpatterns.org
[3] http://watson.kmi.open.ac.uk
[4] http://swoogle.umbc.edu/
[5] http://sindice.com/

to ontology matching, see [14], and methods for analyzing names in ontologies [15]. However, they have been tailored to the specific case of matching a large diverse input ontology on one hand to a small Content ODP on the other hand. To the best of our knowledge no such specific methods have been proposed previously.

3 OntoCase - Retrieve and Reuse

OntoCase is a general framework for pattern-based semi-automatic ontology construction, but in this paper we focus on the retrieve and reuse phases that have been implemented for OWL ontologies. The overall framework can be seen in Fig. 2. The third and fourth phases, revise and retain, are still future work.

3.1 Assumptions and Input

The process is initiated by the input of a data set from which to bootstrap the ontology; "Input" in Fig. 2. The "Element extraction" step uses state-of-the-art OL, i.e. deriving ontology elements from some non-ontological input, such as a text corpus. Since such extraction has been treated in previous research, we simply assume that a preliminary OWL ontology is present. Many OL tools provide a notion of extraction confidence; if present, such values can be used by OntoCase. The methods in this paper cover the "Pattern matching", "Pattern selection", "Pattern adaptation", and "Pattern composition" steps in Fig. 2.

A catalogue of Content ODPs is assumed to be available, "Pattern base" in Fig. 2. Currently the OWL building block itself is used for retrieval and reuse. The current implementation of OntoCase does not treat the pattern construction problem, although the problem is recognized as future work, i.e. the fourth phase (retain). This is not an unreasonable assumption, since recently catalogues of Content ODPs have emerged, such as the ODP portal[6] and catalogues re-engineered from other sources (such as data model patterns [16]), as in [12].

For the matching procedure terms representing both concepts and properties are used, i.e. names or labels (the best match is chosen disregarding if it is a label or a local name). We are assuming that both Content ODPs and the input ontology have human-readable local names (the local part of the resource URI), and/or labels (defined through `rdfs:label`), defined for all classes and properties. Content ODPs represent best practices, hence it should hold for all patterns since this is in itself a good practice, and additionally most OL methods apply some form of term extraction when forming concepts. When matching properties, in the current implementation only the domains and ranges of the properties are used. To explicitly define domain and range is also a best practice, hence most pattern properties have explicit domain and range definitions. For learnt ontologies this may not hold, but again due to the methods commonly applied in OL, such as relation extraction through term co-occurrence, it is very common that domains and ranges are actually present for most properties.

[6] http://ontologydesignpatterns.org

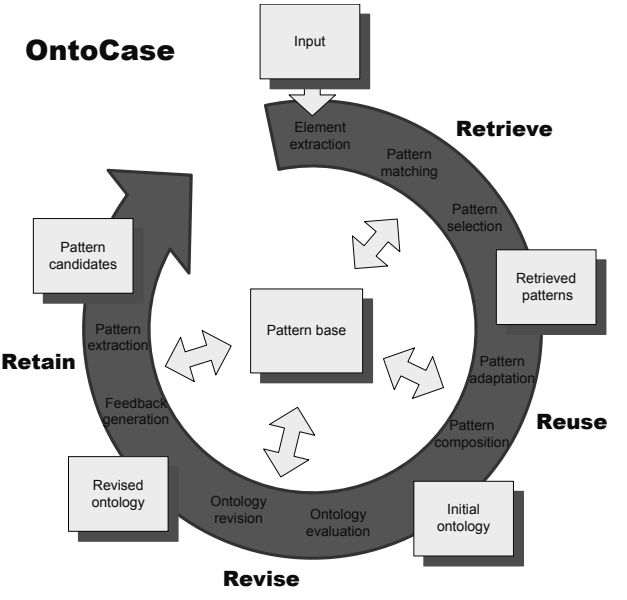

Fig. 2. The overall OntoCase framework

3.2 Example Scenario

As an example, assume that we are constructing a software engineering ontology, and we have extracted the concepts (with local names as follows) "project plan" (with a confidence of 0.5), "schedule" (with a confidence of 0.5), "execution" (with confidence 1.0), and "software engineering task" (with confidence 0.5) from a text corpus. Additionally there is one extracted property named "compose", with domain "software engineering task" and range "project plan". Let us also assume that we have the pattern called *Action* in our pattern catalogue, see section 1.2. This example will be used throughout the description of the method.

3.3 Pattern Ranking

Our pattern ranking and selection method (realizing the "Pattern matching" and "Pattern selection" steps in Fig. 2) takes as input a catalogue of Content ODPs and a preliminary OWL ontology, possibly extracted using some OL tool. The ranking is based on matching between the patterns and the input ontology. Content ODPs are inherently *small* ontologies, hence computationally expensive graph operations can be applied without risking the performance. Content ODP reuse is done through specialization, hence generalizations of pattern concepts are disregarded. The ranking scheme contains three main parts; concept coverage, relation coverage, and utility measures, which are applied as in Fig. 3. Current formulas (details in [12]) are based on related work in ontology ranking, and on analyzing what features intuitively impact pattern suitability.

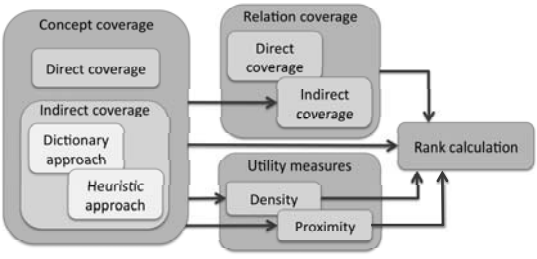

Fig. 3. Parts of the ranking scheme and their dependencies

Concept Coverage. Concept coverage is computed based on direct and indirect term matching. To determine the direct coverage, i.e. discover possible candidates for equivalent concepts, string matching of concept names and labels is used. In order to find only possible equivalences, the matching threshold has to be set quite high. A future extension of this method would be to also study partial inclusion of strings, which would indicate different kinds of relationships between the concepts, or to use "naming patterns" as proposed by [15]. The direct coverage is computed based on the fraction of the pattern concepts that match terms representing a concept in the input ontology O_{input}. The string matching between terms produces a similarity value representing the degree of similarity between two strings; any common normalised string matching measure could be used. These values are then composed into a weighted matching value for each discovered match, using the string matching score and the confidence (if present). For each concept the direct coverage score is computed as the maximum weighted matching value, i.e. the matching score of the best match.

For our example (see section 3.2) we assume the string matching metric is simple string inclusion. Only one term, i.e. "project plan", will match any term, i.e. "Plan", in the *Action* pattern. The term "Plan" constitutes one third of the character string "project plan", hence the resulting score is 0.33. To arrive at the weighted matching score we multiply this with the confidence value, i.e. 0.5, and arrive at a final direct coverage of the "plan"-concept of 0.17.

For the indirect matching, i.e. hints of subclass relations, clues can be found among hypernym relations between terms. Hyponymy/hypernymy denotes a hierarchical relation between terms indicating the specificity of the terms; such a hierarchy is for instance present in the WordNet dictionary. Two approaches are used; hypernym chains in WordNet and the "head heuristic". The "head heuristic" states that a compound term is more specific than the head of the term, i.e. "graduate student" is a specialization of "student".

The dictionary approach starts with a concept in O_{input}. The terms representing this concept are matched against the WordNet dictionary, through exact string matching, and corresponding WordNet terms t_i, if any, are retrieved. The same is done for all pattern concepts. The hypernyms of each t_i are searched for pattern terms. Since t_i can have several senses, there can be several paths, not all leading to a pattern term. The dictionary coverage of a pattern concept

is computed as the sum of the contributions by each O_{input} concept, which in turn depend on the number of paths, the length of the shortest path, and the number of senses. The intuition is that the score increases if several paths are found, while it decreases with an increased length of the shortest path, and with an increased number of senses (i.e. an increased uncertainty).

In our example the terms "schedule" and "execution" can be found in WordNet. The noun "execution" has 7 senses, and for two of the senses we can find a path of hypernym relations connecting them to the term "Action". One path is of length one and the other of length 6. When computing the dictionary coverage of the "Action" concept the number of paths (i.e. two) is divided by the length of the shortest path (i.e. one) and the number of senses (i.e. 7), and weighted with the confidence, arriving at the value 0.29. "Schedule" has two senses in WordNet, and one path of length one leads to the pattern term "Plan". The dictionary coverage of "Plan" is thereby 0.25.

For a concept in O_{input} represented by a multi-word term also the "head heuristic" is applied. The number of modifiers, in our case defined as the number of additional words (disregarding the possibility of multi-word terms being individual modifiers) preceding the head word, are treated analogously to a step in the hypernym chains above. Since we have no information about senses, this is disregarded. In our example the only multi-word term matching a pattern term is "project plan", which matches "Plan" with one modifier (i.e. the word "project"). The head coverage (weighted by the confidence) of "Plan" is 0.5 .

Although more elaborate weighting mechanisms could be applied, currently the total concept coverage of each pattern concept is computed as the sum of the three scores, with a maximum score set to 1. In our example this means that the direct coverage, the dictionary coverage and the head coverage of "Plan" are added, resulting in 0.92. The total concept coverage score of a pattern is the average over all concepts. In the example we have 8 concepts in the pattern, where two of them had matches, resulting in a total concept coverage of 0.15.

Relation Coverage. Matching of properties can be done both based on property names and labels, and on the direct concept matches. The intuition behind this is that neither the property name nor the domain and range is sufficient to determine equivalence (or similarity) of the property, but in combination they give a strong indication. For each pattern property from concept c_d to concept c_r, the best match (if any) is selected from the extracted properties. The score is calculated based on the individual matching scores of the direct concept matches (see above). When matching property names and labels a string similarity measure is used. The score of one pattern property with respect to all properties in O_{input}, is the maximum matching value for any property, which in turn is computed as the average of the string matching score and the combined direct concept coverages (multiplied and weighted by the property confidence).

In our example there is only one direct concept match, hence no pair of such matches connected by a property can be found. The name of the extracted property "compose" can be found similar to the pattern property "composed_of" (with the score 0.64, based on string inclusion). The total relation coverage of the

"composed_of" property is then 0.32. The total relation coverage is the average of the individual matching scores. In our example there are three properties, hence the total coverage is 0.11. A more elaborate strategy for matching relation names could include also common "naming patterns" to identify the possibility of "composed_of" being the inverse of "compose", rather than the same relation.

Utility Measures. The intuition is to assess the "utility" of enriching the input ontology with particular pattern, based on the concept and relation matches. We remind the reader of the aim to give the input ontology a richer structure, thereby utility is interpreted as the ability to add structure. Two utility measures are used; density and proximity, which are inspired by [13].

Density refers to the amount of "structure" that surrounds a certain concept. OntoCase currently considers the number of sub- and superclasses, taxonomical siblings, and concepts directly related through object properties (explicitly defined domains and ranges). In our example, the "Action" concept has three direct subclasses and is related to two other concepts, resulting in the density 5 divided by 8 (the total number of concepts), i.e. 0.63. The density of "Plan" is 0.13. The density values are then weighted using the concept coverage; the weighted density of "Plan" is 0.12 (0.92∗0.13). The complete density of a pattern is the normalized sum of the densities of matched concepts.

The proximity measure considers the distance $dist(c_i, c_j)$ between two matched concepts c_i and c_j in a pattern, which is computed as the length of the shortest path between the concepts, taking into account all relations (subclass and properties explicitly defined with domain and range, disregarding the direction), except paths passing `owl:Thing`. The maximum distance between any two concepts in a pattern is denoted the *pattern diameter*, which is used for normalization together with the fraction of matched concepts. The total proximity value of a pattern is the normalized sum of all the individual proximities.

Ranking and Selection. The three parts are then aggregated into one ranking value. Although more advanced combinations could be imaginable, for simplicity reasons a linear combination is used, currently with equal weights on all measures. The simplest approach for selection is to let the user set a threshold on the ranking value. A more elaborate approach would be to study the total coverage of the patterns over the input.

3.4 Pattern-Based Enrichment

The reuse phase is concerned with adapting, i.e. specializing, and composing the selected Content ODPs, and integrating them into an enriched version of the input ontology (steps "Pattern adaptation" and "Pattern composition" in Fig. 2). More specifically, pattern specialization means adding all the matching results (or those with a score above a certain threshold) as equivalence axioms and subclass relations respectively. In our previous example this means that "execution" would be added as a subclass of "Action" with a confidence of 0.29 (see above), "schedule" added as a subclass of "Plan", and so on. "Project plan"

is ambiguous, i.e. we have evidence of a subclass relation to "Plan" and also an equivalence, hence both relations with their associated confidences are added.

We would additionally like to add the "composed_of"-property, however the "Proposed_action" concept was not matched, only its superconcept "Action". The default heuristic is to only include those parts of a pattern that had some match, not to introduce any unnecessary concepts, but the enrichment can be enhanced by a set of additional heuristics, to create a more well-structured ontology. Firstly, the composition process can be performed in two different modes, pruning or pure enrichment. The pruning mode implies that the input ontology is pruned and only those parts that can be connected to any selected pattern will be included. The pure enrichment leaves the input ontology as it is and only adds the parts of the selected patterns that were matched. Secondly, the (slightly overlapping) heuristics currently available include (some being applicable in only one of the two modes; pruning or pure enrichment):

1. Add all properties (even if not matched) between included concepts.
2. Use the transitive property of subclass relations; if an intermediate concept is missing then add the child directly at the level of the missing concept.
3. Add all extracted subconcepts of concepts in the input ontology, where the parent-concept matched a pattern concept.
4. An object property that originally relates two concepts is added even if one of the concepts is missing if there is a subconcept of the missing concept, which can replace the missing one, present in the ontology.
5. Add all superclasses of matched pattern concepts.

The first heuristic is proposed since relations are harder to extract than terms, hence it is very likely that there will be a number of properties missing in the input ontology. Additionally, we are reluctant to decrease the structural density of the pattern concepts. The second and fourth heuristics are intended to preserve the structure even if there are "gaps" in the matched taxonomy. An alternative strategy would be to include the complete taxonomy, even if only some parts were matched, as suggested in heuristic three and five. Heuristic three is based on the intuition that if a structure exists in the input ontology, this is based on "real-world" evidence (e.g. texts), hence we do not attempt to change this. The intuition for heuristic five, is that we want to use the patterns to add the abstract knowledge that is often missing, even though it might be hard to match it to the input ontology. Adding relations based on the first heuristic even if a concept in the taxonomy is missing, is the focus of the fourth heuristic.

Pattern composition is another task. In the current implementation only simple composition is performed, i.e. relations are not explicitly discovered between patterns. During pattern composition the focus is instead on overlaps between patterns, which are handled using heuristics, for example assuming that two concepts are equivalent if they have the same name and no conflicting axioms. The confidence values resulting from the matching are stored with the ontology.

3.5 Implementation

A first version of the OntoCase method, including the two first phases, has been implemented as a command-line research prototype. The Jena API[7] was used for handling ontologies. Additional external software is the WordNet[8] lexical database. The pattern base is currently deployed as a simple database. The actual pattern ontologies are locally stored as OWL-files, or directly linked to online ontologies on the web. The initial text processing can be done through the Text2Onto[9] tool (an interface to an older version of the tool is provided). The alternative is to provide an input ontology represented as an OWL-file. At the moment a graphical user interface is not provided; plug-in implementations for both Protégé[10] and the NeOn toolkit[11] will be considered in the future.

4 Evaluation

The OntoCase retrieval and reuse phases have been evaluated in three independent settings; the SEMCO project's requirements engineering ontology, the JIBSNet university intranet ontology, and an agricultural ontology of the FAO. All ontologies were constructed using the current OntoCase implementation, set in its pruning mode (described in section 3.4).

4.1 Evaluation Setup

Ontology evaluation methods were used for studying the quality and characteristics of the output of OntoCase. In [17] an overall framework for ontology evaluation is described, consisting of three levels; structural, functional, and usability evaluations. Structural evaluations analyze the quality of the syntax and semantics of the ontology as it is represented. Functional evaluations analyze how well the ontology conforms to the intended conceptualization, i.e. the requirements. Usability evaluations concern the understandability and reusability of the ontology, as well as user satisfaction. We have aimed to cover all levels, although due to practical reasons it has not been feasible for all experiments.

The structural level was analyzed within all experiments, based on measures such as number of concepts, number of concepts at the top level (i.e. root concepts, with no other superclass but `owl:Thing`), number of subclass relations and properties, and average depth of the taxonomy, as suggested in [17,18]. We chose not to apply any formal measure of tangledness, but to evaluate this by inspecting the ontology graphically. The two most well-known approaches for taxonomic evaluation, presented in [19] and OntoClean in [20], were used when feasible (these evaluations were conducted manually by two ontology engineers).

[7] http://jena.sourceforge.net/
[8] http://wordnet.princeton.edu/
[9] http://ontoware.org/projects/text2onto/
[10] http://protege.stanford.edu/
[11] http://www.neon-toolkit.org/

A weakness is that, for two of experiments, only a sample of the elements were evaluated, resulting in the unfeasibility to evaluate, for example, improper semantic leveling, level of detail, and other issues concerning the overall structure.

To evaluate functional characteristics, i.e. the content of the ontologies, a subset of the OntoMetric framework suggested in [21] was used in the SEMCO experiments. Only the dimension *Content* was deemed interesting. This evaluation was performed by two domain experts at the enterprise in question. In the JIBSNet and FAO cases the evaluation was performed using a random sample[12] of classes and properties, whereby the same factors were not applicable. Instead we applied individual assessment of the concepts and properties by domain experts (or ontology engineers in the FAO case). Through a graphical illustration of the concepts, their placement in the taxonomy, and their properties, the experts were asked to classify them into one of five categories; "essential" (i.e. highly relevant for inclusion in the ontology), "accept" (i.e. correct but not essential), "not sure" (i.e. confusing or hard to assess), "not correctly modeled" (i.e. elements that should be included but not in their current form, e.g. too general concepts, or elements wrongly placed in the ontology structure), and "incorrect" (i.e. not to be included). Six domain experts from JIBS participated, representing different roles in the organization and different educational backgrounds. The individual opinions were weighted together, categorizing the elements as "correct" (representing the *essential* and *accept* judgements), "uncertain", and "incorrect" (representing the *not correctly modeled* and *incorrect* judgements). For JIBSNet this can be seen as a usability evaluation, since it was performed by end-users of the application. While, for the FAO case the evaluation was performed by two ontology engineers, using FAO knowledge sources for the evaluation, hence it was a functional evaluation.

4.2 The SEMCO Requirements Engineering Ontology

Within the research project SEMCO an ontology was constructed with the aim to support structuring and retrieval of information and artefacts during the software development process of an enterprise, focussing on the requirements engineering phase. Initially the ontology was aimed at structuring of artefacts within a tool, i.e. ArtifactManager [22]. The aims of this set of experiments were to compare the implementation of OntoCase to both manual ontology engineering, and to an alternative implementation using naive methods.

The naive implementation applies only existing tools, such as string matching and basic heuristics for the enrichment, and resulted in the ontology OA_{naive} as seen in Table 1 (all numbers represent absolute counts). Additionally the resulting ontologies were compared to two versions of a manually constructed ontology, an initial version, $OM_{initial}$, constructed manually based on the same sources used by OntoCase, i.e. a set of documents, and an enriched version also

[12] The sample sizes were between 3 and 72% of the total number of elements (the aim was to stay above 10% but due to practical limitations on subject availability this had to be reduced for the largest ontologies, hence reducing reliability).

refined based on interviews with domain experts, OM_{final}. $OA_{improved}$ is an ontology constructed using the same pattern catalogue but using the current OntoCase implementation, and OA_{final} is the final version of the automatically constructed ontology, constructed using an extended pattern catalogue (including both a domain specific catalogue and the complete set of patterns at that time available from *ontologydesignpatterns.org*).

Table 1. General characteristics of the SEMCO ontologies

Characteristic	$OM_{initial}$	OA_{naive}	OM_{final}	$OA_{improved}$	OA_{final}
Number of concepts	224	85	379	90	150
Number of root concepts	8	35	5	21	13
Number of properties	15	34	246	37	48
Number of subclass relations	224	48	380	95	243
Average depth	2,52	1,95	3,5	2,10	1,62

4.3 The JIBSNet Information Structure Ontology

The second set of experiments were performed in the university domain. JIBSNet is an intranet present at Jönköping International Business School (JIBS). The intranet contains internal documents of all kinds, from personnel instructions to meeting minutes and information for students. An ontology could help to improve classification, presentation and retrieval of information from JIBSNet. The aim of this experiment was to thoroughly evaluate an ontology constructed by means of OntoCase with actual domain experts, i.e. end-users of a tentative application, but also to compare the resulting ontology to the input ontology.

The ontologies were first evaluated using structural measures, see Table 2 (all numbers are absolute counts within each ontology), and then the usability evaluation was performed together with domain experts at JIBS. In Table 2 O_{in} denotes the input ontology, constructed with the OL tool Text2Onto, and O_{out} denotes the output from OntoCase when applying patterns on top of O_{in}.

4.4 The FAO Agricultural Ontology

The third evaluation was set in the agriculture domain, with focus on concepts related to growing rice. The intention of this experiment was mainly to further establish the previous results showing that OntoCase is in fact able to improve the results of existing OL methods. The setting was the Food and Agriculture

Table 2. General characteristics of the JIBSNet ontology

Characteristic	O_{in}	O_{out}
Number of concepts	6535	2576
Number of root concepts	6368	15
Number of properties	218	147
Number of subclass relations	189	4714
Average depth	1.03	2.72

Organization (FAO) of the United Nations and their work on improving the use of agricultural resources around the globe. The organization is trying to improve their processes by moving from simple structures, such as thesauri, to more complex definitions of concepts, such as ontologies.

An existing manually engineered light-weight ontology was used as a starting point, denoted O_1 in Table 3 (all values represent absolute counts within each ontology). Ontologies were also constructed directly from text corpora, using the OL tool Text2Onto; O_2, O_3, and O_4. Two of the text corpora were generated by extracting DBPedia[13] abstracts and comments respectively, collected by using the concepts in O_1 as search terms. A third text corpus was produced by FAO based on article abstracts connected to the AGROVOC thesaurus[14], related to the term "rice". The two final ontologies, O_5 and O_6, were constructed as combinations of O_1 and O_2, and O_1 and O_3, respectively. "+OC" in the table denotes the output ontologies, after applying OntoCase.

Table 3. General characteristics of the agricultural ontologies

Characteristic	O_1	O_{1+OC}	O_2	O_{2+OC}	O_3	O_{3+OC}	O_4	O_{4+OC}	O_5	O_{5+OC}	O_6	O_{6+OC}
No of concepts	266	280	1086	812	365	321	3575	2823	1256	1293	536	570
No of root conc.	155	112	1018	22	290	22	1822	27	1080	758	352	248
No of properties	37	46	17	28	3	16	49	63	53	73	39	62
No of subclass rel.	110	245	89	1471	189	628	1954	4162	199	1679	299	921
Avg depth	1.68	2.19	1.06	2.37	1.34	2.39	1.68	2.84	1.20	3.36	1.57	3.06

4.5 Result Summary

From the SEMCO experiment we see a clear difference between OntoCase and naive methods. Primarily related to the ability to include abstract knowledge, due to the more elaborate matching methods used, hence the ontology is given a more abstract top structure. Although it is not inherently a positive feature to include more abstract concepts, in the specific case of enriching very shallow ontologies, such as the ones learnt by OL tools (even containing unconnected concepts), this is actually an improvement. When compared to manual methods the automatic approach of course does not perform as well, but it has some merits, especially compared to manually constructing an ontology only based on similar (textual) sources. For example, more properties were included in the ontologies constructed by OntoCase. The manually constructed ontologies had even fewer root concepts, i.e. an even more abstract top structure, but it remains to be determined if they are in fact too abstract. Evaluation of the taxonomy (see [19,20]) showed a comparable level of correctness between all ontologies.

The results of the functional and usability evaluations in the JIBSNet and FAO experiments show that with respect to concepts OntoCase performs on the same level of accuracy as Text2Onto, a state-of-the-art OL method. While, with respect to the structure OntoCase considerably improves on the results

[13] http://dbpedia.org/

[14] http://www.fao.org/agrovoc/

of Text2Onto. The top concepts added give a more intuitive structure to the ontology and the relations are deemed correct to a larger extent than the relations of the input ontologies. Table 4 presents the results for the JIBSNet ontology (O_{in} being the input ontology and O_{out} the output of OntoCase). Randomly selected sets were used in the evaluation, for concepts and taxonomic relations one set representing a selection from the top level of the taxonomy and one representing the overall ontology. The reason for this distinction was to show that even the top structure, where most of the pattern concepts and relations were added, is reasonable. The results of the correctness assessment of the FAO agricultural ontologies can be seen in Table 5. The most significant increase in correctness can be seen in the relation assessment (here subclass relations and properties were assessed together).

An increased tangledness of the taxonomy, together with increased redundancy, e.g. a direct subclass relation being present while simultaneously being derivable from a chain of other subclass relations, could be noted in all of the cases. It is important to point out that this might actually be an advantage, if exploited in the right way. As we saw in the example previously, this is due to the ambiguity of the matching results, but it also provides options for an ontology engineer when continuing to refine the ontology, and guidance is given in the form of confidence values. Intuitively the reader may agree that providing some guidance and choices to the user is most likely better than to select one alternative automatically, whereas wrong choices would sometimes be made.

Table 4. Results for JIBSNet concepts and relations

Evaluation set	Assessment	% of concepts	% of subclass relations	% of properties
O_{in} (top structure)	Correct	85.1%	33.4%	N/A
	Uncertain	11.7%	17.5%	N/A
	Incorrect	3.2%	49.2%	N/A
O_{in} (total)	Correct	75.3%	26.8%	50.7%
	Uncertain	15.9%	15.8%	16.2%
	Incorrect	9.2%	57.4%	33.1%
O_{out} (top structure)	Correct	80.6%	58.1%	N/A
	Uncertain	16.1%	38.7%	N/A
	Incorrect	3.2%	3.2%	N/A
O_{out} (total)	Correct	73.7%	53.4%	65.0%
	Uncertain	16.1%	24.9%	11.9 %
	Incorrect	10.2%	21.7%	23.1%

Table 5. Correctness of the agricultural ontologies

Measure	O_1	O_{1+OC}	O_2	O_{2+OC}	O_3	O_{3+OC}	O_4	O_{4+OC}	O_5	O_{5+OC}	O_6	O_{6+OC}
Concepts												
Correct	92.8	97.8	85.5	87.9	94.2	93.0	84.9	88.5	87.8	88.1	92.0	93.6
Unsure	5.4	2.2	3.4	4.6	2.9	4.4	5.7	5.8	5.6	5.6	4.3	3.6
Incorrect	1.8	0.0	11.1	7.6	2.9	2.6	9.4	5.8	6.7	6.4	3.7	2.7
Relations												
Correct	90.0	92.8	61.6	77.5	64.1	86.1	65.9	79.3	76.2	79.1	78.4	88.2
Unsure	5.7	2.9	12.8	2.5	11.5	6.3	13.2	6.9	10.7	8.1	4.9	5.9
Incorrect	4.3	4.4	25.6	20.0	24.4	7.6	20.9	13.8	13.1	12.8	16.7	5.9

To summarize these results we can conclude that OntoCase in fact provides an added structure to the ontologies, and does connect unconnected parts of the ontologies produced by other OL methods. A general top structure is introduced, adding some of the missing general background knowledge not found explicitly in a text corpus. These improvements, conforming to our hypotheses presented in section 1, are achieved without increasing the error-rate of the ontologies.

5 Conclusions and Future Work

Open issues within the OntoCase framework include to cover the complete cycle, i.e. also address the revision and pattern construction problems. A graphical user interface for the revision of OntoCase ontologies would be a great benefit to the user, in order to study the alternative modelling choices present, and to exploit the confidence values as decision support. This would also give the opportunity to experimentally study the amount of manual work required by ontology engineers in different settings. More specific improvements can be found in the use of background knowledge, where WordNet could for example be replaced by domain specific knowledge or knowledge sources available on the Semantic Web, e.g. online ontologies. Some of the rank calculations currently apply quite naive numerical combinations of values, whereas a certain tuning most likely would be beneficial. Also the composition step can benefit from future refinement, for example by explicitly finding relations between patterns, and testing consistency of hypotheses before adding them to the ontology. We do no envision that OntoCase will replace manual editing of the ontologies, aiming for total correctness would require reasoning mechanisms that will most likely not scale. However, as an aid for providing assistance to the user when selecting and integrating Content ODPs we find OntoCase quite useful, despite the simplicity of some of the methods and heuristics currently implemented.

There are many challenges when semi-automatically constructing ontologies from sources such as text corpora. A prime challenge is how to incorporate the "missing information" that is not explicitly stated in domain specific texts. This is both common-sense knowledge and domain knowledge that is assumed and not stated explicitly. We have addressed this challenge by introducing Content ODPs into semi-automatic ontology construction, and we have proposed a general framework for pattern-based semi-automatic ontology construction called OntoCase. Experiments have shown that the ontologies produced are reasonable with respect to their intended domain, and improve the quality of output compared to typical OL methods, primarily with respect to the ontology structure.

References

1. Brewster, C., Ciravegna, F., Wilks, Y.: Background and foreground knowledge in dynamic ontology construction. In: Proc. Semantic Web Workshop, SIGIR (2003)
2. Cimiano, P.: Ontology Learning and Population from Text - Algorithms, Evaluation and Applications. Springer, Heidelberg (2006)

3. Cimiano, P., Buitelaar, P., Magnini, B. (eds.): Ontology Learning and from Text: Methods, Evaluation and Applications. IOS Press, Amsterdam (2005)
4. Ciaramita, M., Gangemi, A., Ratsch, E., Rojas, I., Saric, J.: Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In: Proceedings of IJCAI 2005 (2005)
5. Cimiano, P., Völker, J.: Text2onto - a framework for ontology learning and data-driven change discovery. In: Montoyo, A., Muńoz, R., Métais, E. (eds.) NLDB 2005. LNCS, vol. 3513, pp. 227–238. Springer, Heidelberg (2005)
6. Voelker, J., Vrandecic, D., Sure, Y., Hotho, A.: Learning disjointness. In: Proceedings of the 4th European Semantic Web Conference, Innsbruck (2007)
7. Völker, J., Haase, P., Hitzler, P.: Learning expressive ontologies. In: Ontology Learning and Population: Bridging the Gap between Text and Knowledge. Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam (2008)
8. Coppola, B., Gangemi, A., Gliozzo, A., Picca, D., Presutti, V.: Frame detection over the semantic web. In: Proc. of ESWC 2009. Springer, Heidelberg (2009)
9. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems. Springer, Heidelberg (2009)
10. Gangemi, A.: Ontology Design Patterns for Semantic Web Content. In: Proc. of the Fourth International Semantic Web Conference, Galway. Springer, Heidelberg (2005)
11. Presutti, V., Gangemi, A.: Content ontology design patterns as practical building blocks for web ontologies. Proc. of ER 2008, 128–141 (2008)
12. Blomqvist, E.: Semi-automatic Ontology Construction based on Patterns. PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology (2009)
13. Alani, H., Brewster, C.: Ontology Ranking based on the Analysis of Concept Structures. In: Proceedings of KCAP 2005, Banff, Alberta, Canada (October 2005)
14. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2007)
15. Svab-Zamazal, O., Svatek, V.: Analysing ontological structures through name pattern tracking. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 213–228. Springer, Heidelberg (2008)
16. Hay, D.C.: Data Model Patterns - Conventions of Thought. Dorset House (1996)
17. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Qood grid: A metaontology-based framework for ontology evaluation and selection. In: Proc. of the 4th International EON Workshop, Located at WWW (2006)
18. Yao, H., Orme, A.M., Etzkorn, L.: Cohesion Metrics for Ontology Design and Application. Journal of Computer Science 1(1), 107–113 (2005)
19. Gómez-Pérez, A.: Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases. In: Proc. of KAW 1999, Banff, vol.2 (1999)
20. Guarino, N., Welty, C.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM 45(2), 61–65 (2002)
21. Lozano-Tello, A., Gómez-Pérez, A.: ONTOMETRIC: A Method to Choose the Appropriate Ontology. Journal of Database Management 15(2) (April-June 2004)
22. Billig, A., Sandkuhl, K.: Enterprise ontology based artefact management. GI Jahrestagung P134, 681–687 (2008)