Reusing Ontologies and Language Components for Ontology Generation

Deryle Lonsdale [a,*], David W. Embley [a], Yihong Ding [a], Li Xu [b], Martin Hepp [c]

[a] *Brigham Young University, Provo, UT, USA 84602*

[b] *Department of Computer Science, University of Arizona South, USA*

[c] *E-Business and Web Science Research Group, Bundeswehr University Munich, Neubiberg, Germany*

Abstract

Realizing the Semantic Web involves creating ontologies, a tedious and costly challenge. Reuse can reduce the cost of ontology engineering. Semantic Web ontologies can provide useful input for ontology reuse. However, the automated reuse of such ontologies remains underexplored. This paper presents a generic architecture for automated ontology reuse. With our implementation of this architecture, we show the practicality of automating ontology generation through ontology reuse. We experimented with a large generic ontology as a basis for automatically generating domain ontologies that fit the scope of sample natural language web pages. The results were encouraging, resulting in five lessons pertinent to future automated ontology reuse study.

Key words: Ontology generation, ontology reuse, concept matching, constraint discovery

1 Introduction

Ontology construction is a central research issue for the Semantic Web. Ontologies provide a way of formalizing human knowledge to enable machine

* Corresponding author.

Email addresses: lonz@byu.edu (Deryle Lonsdale), embley@cs.byu.edu (David W. Embley), ding@cs.byu.edu (Yihong Ding), lxu@email.arizona.edu (Li Xu), mhepp@computer.org (Martin Hepp).

interpretability. Creating ontologies from scratch is, however, usually tedious and costly. When the Semantic Web requires ontologies that express Web page content, the ontology engineering task becomes too expensive to be done manually. Many Semantic Web ontologies may have overlapping domain descriptions because many Web sites (or pages) contain information in common domains. It is inefficient to redo ontology engineering for pre-explored domains. These issues illustrate the importance of automated ontology reuse for the Semantic Web.

Ontology reuse involves building a new ontology through maximizing the adoption of pre-used ontologies or ontology components. Reuse has several advantages. First, it reduces human labor involved in formalizing ontologies from scratch. It also increases the quality of new ontologies because the reused components have already been tested. Moreover, when two ontologies share components through ontology reuse, mapping between them becomes simpler because mappings between their shared components are trivial. One can also simultaneously update multiple ontologies by updating their commonly shared components. Ontology maintenance overhead can thus be accordingly reduced.

Despite the many advantages of (automated) ontology reuse, the topic is not well explored in the literature. There are many reasons for this. Before the advent of the Semantic Web, few ontologies existed. Due to the difficulty of constructing ontologies, as well as to the challenges of using ontologies in applications, researchers were less interested in ontology development. With the advance of Semantic Web technologies, the number of ontologies has significantly increased recently. When the use of ontologies in Semantic Web applications improves system performance, more people will appreciate the advantage in using ontologies. In the meantime, most existing ontologies are hard to reuse. The benefits of manual ontology reuse are often unclear since the overhead of seeking and understanding existing ontologies by humans may be even greater than simply building an ontology from scratch. At the same time, many existing ontologies simply do not support effectively automated ontology reuse. The corresponding information in these ontologies is hard to retrieve for automated ontology reuse.

The work we describe below [1] offers three contributions for automated ontology reuse. We first sketch the state of the art in ontology reuse (Section 2). We then present our generic ontology reuse architecture and our implementation (Section 3). Next, we discuss experimental results obtained by using our implementation on real-world examples, as well as five lessons we have learned from this work (Section 4). We conclude with possible future directions (Section 5).

[1] See also www.deg.byu.edu and www.tango.byu.edu.

2 Related Work

Ontology reuse has been studied for years. Most of the earlier research focuses on the study of reusable ontology repositories. In 2001, Ding and Fensel [1] surveyed these earlier ontology libraries. Due to the lack of ontologies, however, very few studies on practically reusing ontologies exist prior to this survey. Uschold and his colleagues [2] presented a "start-to-finish process" of reusing an existing ontology in a small-scale application. According to the authors, the purpose was a "feasibility demonstration only." They concluded that reusing an ontology was "far from an automated process" at that time.

With the growth of semantic web research, more and more ontologies have been created and used in real-world applications. Researchers have started to address more of the ontology reuse problem. Typically, there are two strands of study: theoretical studies of ontology reusability [3–5], and practical studies of ontology reuse [6–8]. Previous studies of ontology libraries showed that it was difficult to manage heterogeneous ontologies in simple repositories. Standardized modules may significantly improve the reusability of ontologies. One major purpose of modular ontology research concerns the reusability of ontologies [3–5]. There are, however, fewer ontology reuse studies quantifying how modular ontologies may improve the efficiency of ontology reuse. Hence one of our purposes is to argue for the use of modular ontologies in real-world, automated ontology reuse experiments.

Meanwhile, there are also several studies on practical ontology reuse. Noy and Musen [7] introduced "traversal views" that define an ontology view, through which a user can specify a subset of an existing ontology. This mechanism enables users to extract self-contained portions of an ontology describing specific concepts. Stuckenschmidt and Klein [8] described another process for partitioning very large ontologies into sets of meaningful and self-contained modules through a structure-based algorithm. Of course, working in the opposite direction is also possible. Pinto and Martins [9] assemble, extend, and integrate small existing ontologies into a large domain ontology by using publicly available ontologies. They discuss two different domains in general terms though not quantifying the size or complexity of the domains or the performance of their ontology reuse process.

Alani et al. [6] coined a new term for reusing existing ontologies: ontology winnowing. The intuition of their research is that individual semantic web applications more profitably use smaller customized ontologies rather than larger general-purpose ontologies. They therefore described a method for culling out—which they called winnowing—useful application-specific information from a larger ontology. Subsequent work of theirs [10] proposes a generic ontology reuse infrastructure that combines ontology search, ranking, segmentation,

mapping, merging, and evaluation. Though technical and experimental details lack in this position paper, this work could well complement our own. In particular, once users build small domain ontologies with our system, they might channel the newly constructed results to Alani's system to assure their future reusability.

A common implicit assumption in all these practical ontology reuse studies is that source ontologies must be reusable for a target domain. Although this assumption simplifies the problem, it does not address the general situation. Besides our work, to the best of our knowledge, the only research that has addressed (albeit implicitly) the domain-specific ontology reuse problem is by Bontas and her colleagues [11]. Their case studies on ontology reuse identified difficulties due to end-user unfamiliarity with the complex source structure. Although this assessment is reasonable, we found a further reason for the difficulty they encountered. Even though source ontologies often declare a target domain, the corresponding information is irretrievable for automated ontology reuse. This is the real bottleneck for automated ontology reuse.

Finally, others concentrate their research on the user's context for ontology reuse. For example, Qin et al. [12] stress the importance of seven salient considerations (user, location, time, activity, service, environment, and platform) and their relationship to ontological structure in reuse. While we recognize the value of this and related lines of investigation, we focus here mainly on the ontology reuse processes that a more general contextualized approach could subsume.

3 Automated Ontology Reuse

Figure 1 sketches our generic architecture for automated ontology reuse. The reuse procedure takes at least two inputs: natural language (NL) documents and source ontologies. NL documents express the projected domains and they can encompass different types. Typical NL documents could include collections of competency questions [13] or collections of sample Web pages [14].

In this architecture, ontology reuse consists of three sequential steps: concept selection, relation retrieval, and constraint discovery. These correspond to the three fundamental components in ontologies: concepts, relationships, and constraints. The concept selection process identifies reusable ontology concepts from source ontologies based on the descriptions in NL documents. NL documents must contain sufficient information for a system to identify all the necessary domain concepts. The identification methodologies vary with respect to different types of NL documents. The relation retrieval process retrieves relationships among selected concepts from the previous step. These

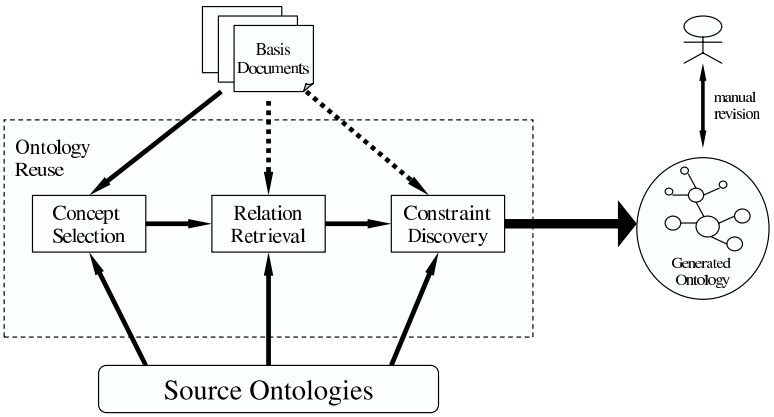


Fig. 1. Generic architecture for automated ontology reuse

relationships can be automatically gathered from source ontologies or perhaps even (optionally) recoverable from the NL documents. Figure 1 represents these optional requirements with dotted lines. The constraint discovery process discovers constraints for previous selected concepts and relationships. An ontology reuse system should be able to gather existing information about constraints from source ontologies or even perhaps from NL documents.

After these three sequential steps, the system composes the selected concepts, relationships, and constraints together into a unified ontology. Human experts then inspect and revise these auto-generated ontologies.

We have implemented a prototype automated ontology reuse system based on this generic architecture. Our system reuses existing ontologies to create small domain ontologies within the scope of describing individual Web pages. In the rest of this section we describe the system in more detail.

3.1 Preparation of Input

We first take a small set of sample Web pages as input NL documents, and pre-process them to focus on the content that reflects the domain of interest. This involves removing extraneous information such as advertisements, side bars, and links to various other pages. Similarly, many pages might combine several records together; in such cases each record is factored out and stored as a separate document. We do some of these processes automatically and others manually; information about how we automate this document preprocessing work is available elsewhere [15,16]. Only the main body of each page remains, which constitutes the focus of interest for readers.

Proper preparation of source ontologies is also essential for ontology reuse automation. Poorly integrated source ontologies create a very complex ontology integration problem during final composition of the output ontology.

Two options exist: either we can directly adopt a single large-scale ontology, or we can manually pre-integrate several small ones. For simplicity, we chose the first option (and will discuss the second option later). Specifically, we adopted the MikroKosmos (μK) ontology [17], a large-scale ontology containing more than 5,000 hierarchically-arranged concepts (excluding instances). These broad-coverage concepts describe various domains, a desideratum for flexible experimentation. The μK ontology has an average of 14 inter-concept links/node, providing rich interpretations for the defined concepts.

We adopted the DTD for the XML version of μK ontology as our standard ontology input XML DTD. Because the μK ontology is represented as a directed graph, mapping any other graph-based knowledge sources, either directly or indirectly, to μK is straightforward.

Still, the basic μK ontology itself is not sufficient for ontology generation. It does not contain extensive lexical content, but rather has an associated lexicon base that supports the ontology matching. Since the original lexicons were not available for this work, we created our own separate lexicon base. This was done by pre-integrating the leaf concepts from the μK ontology with external lexicon dictionaries and declarative data recognizers. Most of these lexicons and data recognizers are collected from the Web. For example, for the ontology concept CAPITAL-CITY we used a web browser to locate lists of all the capital cities of the independent countries in the world. Since we collected information from varied resources, we found that synonym identification became critical for the performance of ontology reuse. We therefore adopted WordNet [2] for our synonym resource in this work; see Lesson 5 below for related work involving a hierarchical terminology resource.

To the extent possible, we rely on available resources for specifying lexical content to be used in concept discovery and matching. However, we have found it useful to develop our own library of data frames (i.e. data recognizers) used for this purpose. These data frames consist of dictionaries of lexical items where enumeration is necessary; otherwise their lexical content is specified via regular expressions were possible. Associated with these definitions are contextual keywords and other informative clues that participate in string matching operations for concept matching.

Although this source ontology preparation process is quite involved, it is a one-time effort. This integrated ontology source thus become static and constant for all downstream ontology reuse applications.

[2] http://wordnet.princeton.edu/

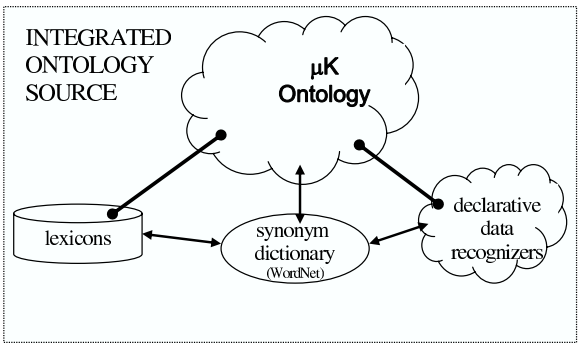


Fig. 2. Pre-integrated source ontology

3.2 Ontology Reuse Process

Figure 1 shows how our system extracts an appropriate sub-domain from a larger, integrated source ontology by executing concept selection, relation retrieval, and constraint discovery. Since any ontology can be viewed as a conceptual graph, our algorithm is implemented to find nodes, edges, and specific constraints in graphs.

3.2.1 Concept Selection

We have implemented the concept selection process as three concept recognition procedures (which could be executed in parallel) followed by a concept disambiguation procedure. In particular, the three recognition procedures involve matching concept names and concept values.

Concept name matching associates content in NL documents with concept names in source ontologies. We have implemented three concept selection strategies:

S1 compare a string with the name of a concept;
S2 compare a string with values belonging to a concept; and
S3 apply data-frame recognizers to recognize a string.

S1 and **S2** are concept recognition procedures (which could be executed in parallel) and **S3** is a concept disambiguation procedure. Concept name matching associates content in NL documents with concept names in source ontologies in the preparation stage described above.

For example, consider the sentence "Afghanistan's capital is Kabul and its population is 17.7 million." Concept matching would associate the word "capital" with the ontology concepts CAPITAL-CITY and FINANCIAL-CAPITAL. It also matches the word "Kabul" with the concept CAPITAL-CITY.

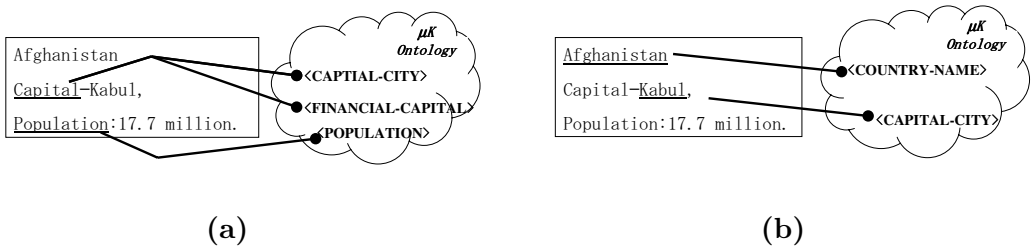


Fig. 3. Matching concepts from the μK ontology via: name selection (a), and value selection (b)

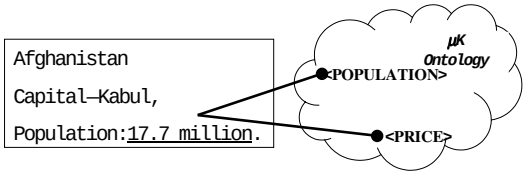


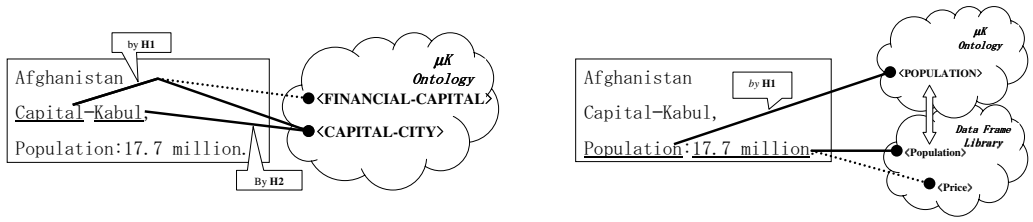
Fig. 4. Matching concepts via data frames

The process is similar for Web page content. In Figure 3(a), **S1** matches the string "Capital" in a Web page with the concepts CAPITAL-CITY and FINANCIAL-CAPITAL in the μK Ontology. Similarly, the string "Population" matches with the μK concept POPULATION. Thus the three object sets, namely CAPITAL-CITY, FINANCIAL-CAPITAL and POPULATION, are potential candidates for selected object sets.

Figure 3(b) shows the kind of matches **S2** provides. "Afghanistan" is an instance of a name of a country and "Kabul" is an instance of a capital city. Hence we select the object sets COUNTRY-NAME and CAPITAL-CITY as potential candidates.

Figure 4 illustrates **S3**, which selects concepts based on regular-expression recognizers in the data-frame library. Two data frame patterns match the "17.7 million" in the training record. One pattern associates with the concept Population, and the other associates with Price. Note that the knowledge sources preprocessing has already integrated the concepts in the data-frame library with the concepts in the μK Ontology. We thus add the associated two μK concepts into the potential candidate list of object sets.

Clearly, matches occasionally entail incorrect conceptual interpretations. For example, in Figure 3(a), **S1** selects both FINANCIAL-CAPITAL and CAPITAL-CITY, but only the latter is correct. The conflict resolution step attempts to resolve these incorrect matches. To resolve conflicts, we have implemented three heuristics, namely: (1) *same string only one meaning*, (2) *favor longer over shorter*, and (3) *context decides meaning*. We sketch each in turn.



(a) **(b)**

Fig. 5. Resolution via *same string only one meaning*, using: **S1** and **S2** (a), and **S1** and **S3** (b)

Same string only one meaning. A given string in a document record may match more than one concept. Such a string, however, can have only one meaning for each occurrence. Therefore, only one concept can be the correct selection. For our purposes if all three heuristics **S1**, **S2**, and **S3** confidently identify a concept to be a correct selection, any other concept by the same string that conflicts with this interpretation is considered an error. In fact, we use the weaker condition that if in a document, both **S1** and **S2**, or both **S1** and **S3** select a concept, we accept it as a correct selection. [3] Any other concept that recognizes the same string then is deemed an error, and the conflict resolution procedure removes it from the list of candidate concepts. Figure 5(a) illustrates this heuristic.

Although both FINANCIAL-CAPITAL and CAPITAL-CITY match "Capital" in the document via **S1**, we know we cannot have both. Since CAPITAL-CITY also matches "Capital" by **S2**, we accept CAPITAL-CITY as a correct selection.

Figure 5(b) shows another example. Although **S3** suggests both the concept POPULATION and PRICE for the string "17.7 million", the system accepts the concept POPULATION as the correct selection based on the additional suggestion from **S1** because of the string "Population".

Favor longer over shorter. If concept A recognizes string A' and concept B recognizes string B', where A' is a proper substring of B', the selection process chooses concept B over A because the meaning of B' overrides the meaning of A'.

In Figure 6(a), **S2** recognize the word "bronze" as an instance of ALLOY, while **S2** matches the whole string, "bronze medal" as an instance of SPORT-ARTIFACT. Because "bronze" is a proper substring of "bronze medal", the system chooses the concept SPORT-ARTIFACT instead of the concept ALLOY.

Context decides meaning. Sometimes we can use the context around a target

[3] Theoretically, exceptional cases may exist, though we have not encountered any during our experiments.

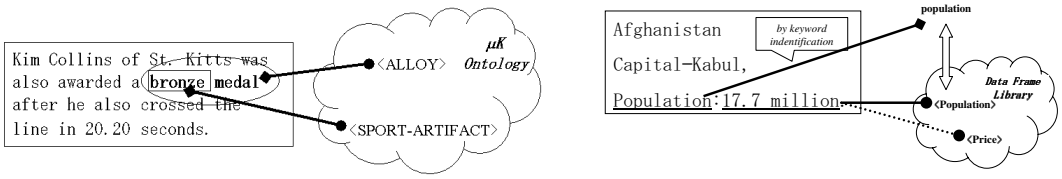


Fig. 6. Resolution via: *favor longer over shorter* (a), and *context decides meaning* (b)

string to identify one of multiple matching concepts for a string. Figure 6(b) shows an example.

In order to decide between the potential candidates POPULATION and PRICE, we use an associated keyword defined in the data-frame library as a context keyword. Since we find the keyword "Population" close to the occurrence of a data value for the concept POPULATION and do not find any context keywords of the concept PRICE, such as the strings "price", "cost", "GNP", "amount", "$", or "dollars", the concept POPULATION has a higher preference for being the correct match. The data frame library provides these contextual keywords.

3.2.2 Relation Retrieval

The crux of the relation retrieval process is about finding appropriate edges between two concept nodes. An obvious resolution is to find all possible paths between any two candidate concepts. It would be easier for users to reject inapplicable edges rather than to add new relations. But this resolution entails serious performance difficulties. Hence we must seek an alternative resolution.

Different paths in an ontology graph refer to relations with different meanings. In addressing the problem of ontology generation, it is almost never necessary to find all possible paths; for a generated ontology we typically only want one. From our studies we have found that in general a shorter path represents a closer or more straightforward relationship between two concepts. An extra-long path often means a very uncommon relation between the two concepts within the domain. Hence it is reasonable to set a threshold length to reduce the search space and thus the complexity of the edge-searching algorithm.

In the implementation, we adapted Dijkstra's algorithm [18]. Although the original algorithm computes only the shortest path, it can be easily extended by repeatedly computing the next shortest path until a threshold length is reached. Since Dijkstra's algorithm has polynomial time complexity and the threshold length is fixed and finite, the time complexity of this updated algorithm is also polynomial.

To illustrate, consider the concepts NATION, COUNTRY-NAME, and CAPITAL-CITY from the previous concept selection discussion, along with identified instances (for the last two), namely Afghanistan and Kabul. Relation retrieval begins by choosing a node (for example NATION) from the μK ontology. We then perform an exhaustive and recursive search to find all reachable nodes and their distance from that node, up to some thresholded distance. When other previously identified concepts are reached, a relationship can be set up between them. For example, both COUNTRY-NAME and CAPITAL-CITY are reachable in this way from NATION, so corresponding relations are established. Occasionally paths of equal length may extend from a concept to more than one ancestor (or descendent) node; in such cases we arbitrarily choose the first one for practical purposes. However, the discussion in Lesson 5 below focuses on why further processing is justified in such cases, and how we have in fact discovered useful information when allowing multiple paths to be pursued.

After this edge-searching procedure, the system performs a subgraph detection procedure to finalize the target domain. Quite often, the edge-searching procedure results in multiple unconnected subgraphs. Normally, two separate subgraphs represent two independent domains. However, since we focus only on narrow-domain applications and assume that the NL documents represent this domain, we expect only one subgraph. Such would be the case with the concepts mentioned in the previous paragraph; only one subgraph in the political entities domain would result. Since we also assume that the NL documents are data-rich, we can assume that the largest subgraph of concepts is the subgraph of interest. This subgraph becomes the conceptual model instance for the ontology we generate. Obviously this also helps boost the accuracy and performance of the system by reducing the possible inventory of concepts to be processed.

We use a standard algorithm to find the largest subgraph, which is the subgraph containing the largest number of concepts. Theoretically there could be two largest subgraphs with the same size. In practice this situation rarely occurs; indeed, usually we obtain one large subgraph and either no other subgraphs or only a few other small subgraphs, each often having just one node and no edges.

3.2.3 Constraint Discovery

Ontologies involve numerous types of constraints; this paper cannot possibly enumerate them all or discuss the methods for reusing constraints. For our purpose in demonstrating automated ontology reuse, we limited our study to cardinality constraints and their discovery. Unlike many other constraints in ontologies, cardinality constraints contain quantitative scales, which render the automatic discovery process particularly interesting. Generating many

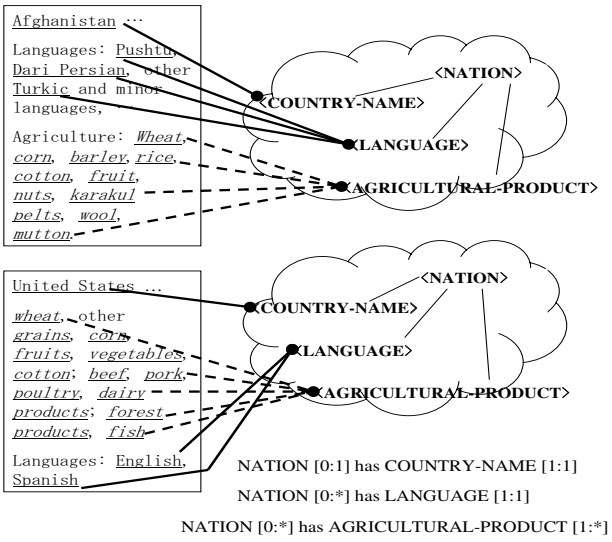


Fig. 7. Generating participation constraints

other constraints, such as hasValue (specifying value instances of particular concepts), is straightforward via our approach. Based on the data recognizers, we can immediately obtain the mapping between extracted instances and their concepts. The generation of these purely descriptive constraints is usually easier than producing constraints without precise quantitative scales such as cardinality. Hence here we address arguably the most difficult constraint generation process: cardinality constraint generation.

We have implemented a cross-counting algorithm to discover cardinality constraints from NL documents.[4] The cardinality constraints we discover are participation constraints [19]. Each participation constraint consists of a pair with a minimum number and a maximum number $[min : max]$. The cross-counting algorithm counts the instantiated numbers of paired concepts, from which the system can decide these minimum and maximum numbers. For example, suppose that in document $D1$ concept A is instantiated by $a1$, and there are no instantiations for concept B in the same document. In another document $D2$, however, concept A is instantiated by the same $a1$ and concept B is instantiated by $b2$. With these two documents, we can determine that the minimum cardinality constraint of concept A to the relation AB is 0 because for an instance $a1$ of A, it may not always have an instance of B appearing at the same time. The details of this algorithm are presented elsewhere [14].

Figure 7 shows an example of discovering a $[1 : 1]$ participation constraint. Suppose that we have two NL documents as this figure shows. For each record, we recognize three concepts: COUNTRY-NAME, LANGUAGE and AGRICULTURAL-

[4] The original μK ontology does not contain information about cardinality constraints.

PRODUCT. By the concept resolution process described earlier we determine that NATION is the central (or most important or primary) concept of this application. We retrieve three binary relationship sets: `NATION has COUNTRY-NAME`, `NATION has LANGUAGE`, and `NATION has AGRICULTURAL-PRODUCT`. Since NATION is the primary concept, the minimum participation constraint for NATION for each relationship set is 0, and each minimum participation constraint of the object sets in each of the three relationship sets is 1. By construction, the primary concept appears only once in each record. The concept COUNTRY-NAME also appears only once in each record, and for a different instance of COUNTRY-NAME appearing in each record. Thus the system sets the maximum participation constraint for both concepts to 1. Hence the system generates the object set `NATION[0:1] has COUNTRY-NAME[1:1]`.

For the relationship set `NATION has AGRICULTURAL-PRODUCT`, the system generates the maximum participation constraint for NATION to be * because each record contains more than one instance of AGRICULTURAL-PRODUCT. However, we can find at least one instance "wheat" in both the records. That is, the same instance of AGRICULTURAL-PRODUCT can be found for different instances of NATION. The system therefore also generates the maximum participation constraint for AGRICULTURAL-PRODUCT to be *. Hence, the system generates `NATION[0:*] has AGRICULTURAL-PRODUCT[1:*]`.

For the relationship set `NATION has LANGUAGE`, Record 1 has three instances of LANGUAGE and Record 2 has two instances of LANGUAGE while none of these instances is the same. Thus, each NATION can have more than one LANGUAGE, and each LANGUAGE belongs to only one NATION. Hence, the system generates `NATION[0:*] has LANGUAGE[1:1]`.

Note that this last example is not totally correct. People know that more than one nation can use the same language. For example, both the United States and the United Kingdom use English as their native language. Significant bias may result from a lack of sufficient examples. Thus, we need a large enough collection of NL documents to generate correct participation constraints. However, even a huge unsupervised selected document base may not totally solve this problem. Supervised selection of NL documents can overcome this problem, but the tradeoff requires more human effort.

3.2.4 Ontology Refinement

After finding concepts, relations, and constraints, composing them together into an ontology is straightforward. The result probably will not precisely describe the target domain, so in a final stage a human revises the resulting ontology manually. There are four basic operations for revision: (1) remove the unexpected content (i.e. object sets or relationship sets), (2) rename the inap-

propriate content, (3) modify the incorrect content, and (4) add the missing content. In general, a preferred ontology reuse procedure will produce outputs requiring less revision operations on (3) and (4), especially the latter. It is in general much easier for users to reject unexpected components than to add something totally new into an ontology on their own initiative. Based on this refinement perspective, our ontology reuse system preserves as much useful information as possible, minimizing the need for addition by users.

4 Experiments and Discussions

This section describes a series of experiments with our ontology reuse system; full details on the experimental methodology, results, and evaluation are available in [1]. We cast our discussion in five lessons that we believe are pertinent to future automated ontology reuse studies.

Lesson 1. Ontology coverage is best specified by the leaf concepts.

For ontology reuse, the coverage of an ontology is the reusable domain described by an ontology. Users for ontology reuse could conceivably believe that one can straightforwardly determine the coverage of an ontology by its root definition. For example, when the root concept of an ontology is BOOK, this ontology should cover the domain of books; when the root concept is FINANCIAL-REPORT, this ontology would ideally cover the domain of financial reports. Since the root of the μK ontology is ALL (i.e. everything), we began our study with the expectation that we could reuse the μK ontology to describe arbitrary domains.

Our initial experiments were not promising: usually we either got no result or the composed ontologies were outside the expected domains. In retrospect the solution was fairly obvious: the real coverage of an ontology is often not necessarily best characterized by its root definition. Instead, often ontology developers do not properly circumscribe the domain, and in such cases a significant portion of the domain is often not reusable.

More precisely, the true reusable domain or coverage of an ontology is primarily determined by the union of its leaf-level concepts, a subset of the root-specified domain. For example, if a NATION ontology contains leaf-level concepts like USA, RUSSIA, CHINA, AUSTRALIA, etc., but lacks MONTENEGRO, the concept MONTENEGRO is not reusable with respect to this ontology. This fairly simple observation, though critical for ontology reuse research, seems not to have been documented in the prior ontology reuse literature.

Lesson 2. Extend ontology coverage with lexicons and data recognizers.

To improve the degree of reusability of existing ontologies, we want to boost the coverage of an ontology so that it is closer to its root definition. We refer to this as the "applicable coverage" of an ontology, where the term "applicable" means the new concepts can be evaluated by an ontology reuse program.

To boost the applicable coverage of our source ontology during the source-ontology preparation stage, we associated lexicons and data recognizers with the leaf-level concepts. We have named the result "instance recognition semantics", or formal specifications that identify instances of a concept C in ordinary text [20]. These are essential to automating ontology reuse.

We further populate the ontology with some upper-level ontology concepts. For example, prior to June 3, 2006 Montenegro was not an independent nation, so the original μK ontology did not have a leaf concept MONTENEGRO under NATION. This portion of the ontology becomes non-reusable for many situations involving Montenegro after June 3, 2006. It is a very complicated issue to obtain permission and then properly modify an ontology that is created by somebody else. For the purpose of automated reuse, however, we developed a simple and effective (though imperfect) alternative. We simply bind a lexicon set to the non-leaf concept NATION, thus adding the name of Montenegro into the lexicon after June 3, 2006. Although we still have not formally specified Montenegro as a country in the ontology, we have rendered the original source ontology reusable for situations involving the new country Montenegro. In the new generated ontology, instead of a specific concept MONTENEGRO as an independent nation, we can correctly generate an upper-level concept—NATION, and thus all the properties of NATION become applicable in this new generated domain ontology. With such a technique, we artificially boost the applicable coverage of the source ontology.

In our experiments we augmented lexicons and data recognizers for leaf-level concepts in the μK ontology and their superclasses up to 2 levels above (on average). The union of these augmented concepts and their relations composes the applicable coverage of the source ontology in our experiments.

Lesson 3. For known target domains, ontology reuse is already possible and even valuable.

After having prepared the source ontology, we started our real experiments. Based on Lesson 1, we decided to focus our experiments on several selected domains rather than on arbitrary domains. We want human inspection to assure that the projecting domains have significant overlap with the applicable coverage of our source ontology. In particular, we chose experiments in three narrow

domains: car advertisements, apartment rentals, and nation descriptions. This paper only briefly summarizes our results; see [14] for details.

First we list some basic settings and statistics of our experiments. Each of the three target domains contains a dozen to twenty concepts. For each domain, we feed four to seven cleaned sample Web pages (NL documents) to the ontology reuse system. The source ontology had been pre-integrated and augmented by its applicable coverage. In order to evaluate the performance of our outputs, we had human experts separately create ontologies for each target domain. We adopted the human-created ontologies as a gold standard to which the automatically generated ontologies were compared for precision and recall.

In general, we obtained low precision results. In the three target domains, the best precision was 48% for concept generation, 14% for relation generation, and 10% for cardinality constraint generation. The news is not all bad. Low precision implies the need for more rejections of corresponding components within a generated ontology. For humans, as mentioned earlier, rejecting inappropriate ontology components is much easier than adding new ontology ones. Hence our strategy is to favor greater recall values (i.e. less addition) over greater precision values (i.e. less rejection).

We updated the traditional recall calculation equation as follows:

```
updated recall = # correctly-reused / # existing-in-source
```

where the numerator is the number of component types (i.e. either concept, relationship, or constraint) correctly reused in a generated ontology; the denominator is the number of component types contained in input sources (both from NL documents and source ontologies). We use this formula because not everything defined in the human-created ontology is also identifiable by the inputs. For example, human experts defined a concept FEATURE in the car-ads ontology, a concept missing from the source μK ontology. Hence it is impossible for a system to reuse an non-pre-existing concept. To be more equitable, our recall calculation must eliminate this type of error.

With the new formula, in the three test domains our worst recall values were 83% (concept generation), 50% (relation generation), and 50% (cardinality constraint generation). All the best recall values were close or equal to 100%. Our ontology reuse system performs quite well even though it still is a prototype. The recall values show that we may reduce at least half of the human effort in ontology construction through ontology reuse when a target ontology is properly contained in the applicable coverage of the source ontology. Considering the expense of training professional ontologists and the time they need to build and tune ontologies, 50% already represents substantial savings. There are many ways to further improve the performance of the system. Already, though, our experiments demonstrate that ontology reuse is no longer

"far from an automated process" [2].

Lesson 4. Ontology modularization facilitates and accelerates automated ontology reuse.

During our experiments, another metric studied was running time. In general the system took about 1,000 seconds to resolve all the ontology components with respect to about 50 to 100 candidate concepts on a Pentium 800 MHz single processor machine. This execution time is rather short compared to the time required for manually creating an ontology of the same scale. Our benchmark showed that almost 90% of the execution time was spent on the relation retrieval process. Though we may further improve this time performance by optimizing our implementation, the problem lies mainly in the magnitude of the source ontology (over 5,000 concepts and over 70,000 relationships to explore).

Reducing the execution time of relation retrieval should be possible by using modular ontologies rather than a single large-scale one. Modular ontologies are usually small and designed to be self-contained. An ontology module is self-contained if all of its defined concepts are specified in terms of other concepts in the module, and do not reference any other concepts outside the module. As soon as several major concepts in a module are selected as candidate concepts, an ontology reuse system may decide to directly reuse the entire module rather than perform a costly relation retrieval algorithm. Hence the execution time for relation retrieval can be significantly reduced.

To pursue this issue, we manually pruned several comparatively independent clusters of ontology components from our source ontology and used them as individual modules. Since these clusters originated from a previously unified ontology, we did not need to further integrate them. The same experiments were re-run with these multiple "modular" ontologies. On average the system took less than 300 seconds—saving more than 70% of run time—to resolve all the ontology components for about 50 to 100 candidate concepts. Because these pruned clusters were not true, self-contained modular ontologies, the performance in terms of precision and recall decreased in this experiment. Saving execution time by replacing a large unified ontology with multiple small modular ontologies is thus a convincing strategy. By using truly well-designed modular ontologies, our ontology reuse system achieves both higher precision and recall values, as well as faster run-time performance.

Lesson 5. Sample documents may help us mine "latent" knowledge from text documents.

We also carefully studied our low-precision experimental results. Many reused

concepts and relations that were generated fell beyond the scope of the expert-created ontologies. Yet they were not all meaningless or useless. On the contrary, we found that useful information—latent in the document but beyond the topic directly at hand—could be gleaned from the results. Such latent information is not really what people cannot find, but is easily overlooked by human readers. We believe that automated ontology generation may provide various industrial and research communities an attractive solution for seeking valuable latent information.

Of course, the generation of reasonable ontologies depends crucially on how well the knowledge base addresses the domain of interest. A perfect fit is not likely, so we rely on extensible knowledge bases which allow ontology builders to generate ontologies even in the presence of only partial coverage. In this section we sketch an experiment that explored extending the default μK ontology with two other knowledge sources and testing it on a novel application domain.

One resource was Eurodicautom, a large-scale terminology bank [5] which consists of over a million concept entries covering a wide range of topics. Each entry is multilingual in character, containing equivalents in any of several languages. For this work we only focused on the English terms, discarding those in other languages. Since Eurodicautom was not in XML format at the time, we first converted entries into the TBX lexicon/termbase [6] exchange framework developed by the SALT project. SALT specifies a data model for interchange among diverse collections of lexicon/termbase data, including a provision for their ontological structure. [7]

The TBX termbase itself is not helpful for this task, though, because it merely defines terms with only minimal hierarchical or relationship information. However, an associated resource called Lenoch places each Eurodicautom term into a more completely specified conceptual hierarchy. Because Lenoch codes define a hierarchy for Eurodicautom terms, we were able to integrate them together and arrange them in the μK XML format. Figure 8(a) shows some examples of Lenoch codes, and figure 8(b) shows corresponding samples from the μK XML encoding. More details on the Eurodicautom, Lenoch, and μK integration are documented elsewhere [21].

We ran our tool on various U.S. Department of Energy (DOE) abstracts [8] . The expert who created a reference ontology in advance was only interested in the generic information about these abstracts, such as the theme of a document,

[5] On 28 June 2007 the Eurodicautom project was subsumed by the new Inter-Active Terminology for Europe (IATE) project.
[6] This is the widely used term for terminology (data)bases.
[7] In progress as ISO 16642 (forthcoming).
[8] A corpus in the ACL/DCI set; see www.ldc.upenn.edu/Catalog/LDC93T1.html.

```
ER Earth Resources - Energy
ER1 general aspects of the subject field
ER4 energy sources
ER41 fossil energy
ER42 nuclear energy
ER43 alternative sources of energy
       (nt: primary sources of energy)
ER431 durable sources of energy
ER4311 wind energy
ER4312 tidal energy
ER4313 solar energy
ER7 fossil raw materials
ER71 coal (sn: energy source; ...)
ER711 coal varieties
```

(a)

```
<RECORD>
  <CONCEPT>energy sources</CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE/FACET>
  <FILLER>alternative sources of energy </FILLER>
  <UID><0></UID>
</RECORD>

<RECORD>
  <CONCEPT>alternative sources of energy </CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE/FACET>
  <FILLER>durable sources of energy</FILLER>
  <UID><0></UID>
</RECORD>

<RECORD>
  <CONCEPT>durable sources of energy</CONCEPT>
  <SLOT>SUBCLASSES</SLOT>
  <FACET>VALUE/FACET>
  <FILLER>wind energy</FILLER>
  <UID><0></UID>
</RECORD>
```

(b)

Fig. 8. Sample Lenoch codes (a) and XML data model from Eurodicautom via SALT (b)

the number of figures and tables, etc. But our ontology reuse tool found much more useful information. For instance, in one sample abstract it generated some concepts and relations indicating that crude oil prices dropped in the year 1986. Although this was not what the human expert originally expected and it was outside the expert-specified domain of interest, we could not deny that this type of information could be very valuable. Figure 9 shows a very small portion of the output ontology generated by the system using the updated knowledge base.

Within the generated ontology, of course, there are several dozen posited relationship sets; some are spurious but in fact several are correct. For example, a novel relationship is appropriately posited between the concept CRUDEOIL (which was a new concept introduced via the integration of Eurodicautom) and the concept NATION (an original concept in μK). The discovery and capture of such interactions between concepts is possible via the procedure outlined in this paper. It should be noted that the focus of this experiment was to test whether the system could properly integrate new sources and generate meaningful relationships between the new added concepts and the original ones. A larger question is whether the DOE abstracts are well suited for treatment by the system, and here we are skeptical since they do not contain extensive data-rich, narrow-domain information that we assume as input.

```
-- DOE Abstract Ontology

Alloy [0:*] MadeOf.SOLIDELEMENT.Subclasses MetallicSolidElement [0:*];
Alloy [0:*] IsA.METAL.StateOfMatter.SOLID.Subclasses CrudeOil [0:*];
Alloy [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Produce [0:*];
AmountAttribute [0:*] IsA.SCALARATTRIBUTE.MeasuredIn.MEASURINGUNIT
Consumption [0:*] IsA.FINANCIALEVENT.Agent Human [0:*];
ControlEvent [0:*] IsA.SOCIALEVENT.Agent Human [0:*];
ControlEvent [0:*] IsA.SOCIALEVENT.Location.PLACE.Subclasses Nation [0:*];
CountryName [0:*] NameOf Nation [0:*];
CountryName [0:*] IsA.REPRESENTATIONALOBJECT.OwnedBy Human [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.OwnedBy Human [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.GROW.Subclasses GrowAnimate [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Increase [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Combine [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Display [0:*];
CrudeOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Produce [0:*];
Custom [0:*] IsA.ABSTRACTOBJECT.ThemeOf.MENTALEVENT.Subclasses AddUp [0:*];
Display [0:*] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.Subclasses Gas [0:*];
Display [0:*] IsA.PHYSICALEVENT.Theme.PHYSICALOBJECT.OwnedBy Human [0:*];
ForProfitCorporation [0:*] OwnedBy Human [0:*];
ForProfitCorporation [0:*] IsA.CORPORATION.HasNationality Nation [0:*];
Gas [0:*] IsA.PHYSICALOBJECT.Location.PLACE.Subclasses Nation [0:*];
Gas [0:*] IsA.PHYSICALOBJECT.ThemeOf.GROW.Subclasses GrowAnimate [0:*];
LinseedOil [0:*] IsA.PHYSICALOBJECT.ThemeOf.PHYSICALEVENT.Subclasses Increase [0:*];
```

Fig. 9. Sample relations in the generated ontology for DOE abstracts

5 Conclusion and Future Work

We have presented an automated ontology reuse approach. Although we only applied our system to reuse the μK ontology, our methodology supports automated ontology reuse in general. Informed by our experiments on real-world examples, we have summarized five lessons that are constructive for future exploration of ontology reuse studies. In essence, we conclude that ontology reuse is no longer "far from an automated process" [2].

In the meantime, a few critical problems remain to be solved. One is to automatically decide whether a target domain is within the reusable coverage of an integrated source ontology. If the majority of a target domain lies outside the source ontology, ontology reuse becomes nothing but extra overhead.

We also need to experiment with applying modular ontologies for ontology reuse. Until now, the research on modular ontologies is still at the stage of theoretical analysis. We need practical study cases to push this research field forward. The study of instance recognition semantics should be paired with modular ontology research to improve the reusability of modular ontologies.

Logic-based approaches have been used with other ontologies; Grau et al. [22] extract particular modules within existing OWL ontologies by detecting the internal logic of the ontology based on derivational proofs involving the notion of *syntactic locality*. Their experiments with large, well-known ontologies such

as the Gene Ontology [9] and the SUMO Upper Ontology [10] yield scores of 85% accuracy for concept reuse in general cases. Though the scoring criteria are somewhat unclear and there is no discussion of relationship and/or constraint reuse in the paper, it may be possible to build a hybrid system integrating this logic approach with our own data extraction ontology reuse system to combine the benefits of both techniques.

Last but not least, mining latent information through ontology reuse is an interesting research topic. More exploration on this topic may bring many benefits to users, especially in the business domain.

So far there are few published studies on automated ontology reuse research. We hope that our results draw more attention to this field of research.

Acknowledgements

This work was funded in part by U.S. National Science Foundation Information and Intelligent Systems grants for the TIDIE (IIS-0083127) and TANGO (IIS-0414644) projects. Part of the work was also supported by the European Commission under the projects DIP (FP6-507483), SUPER (FP6-026850), and MUSING (FP6-027097), by the Austrian BMVIT/FFG under the FIT-IT Semantic Systems project myOntology (grant no. 812515/9284), and by a Young Researcher's Grant from the University of Innsbruck. We are also grateful to Sergei Nirenburg for providing a copy of μK for this work.

References

[1] Y. Ding, D. Fensel, Ontology library systems: The key for successful ontology reuse, in: Proceedings of the First Semantic Web Working Symposium (SWWS'01), Stanford, CA, 2001, pp. 93–112.

[2] M. Uschold, M. Healy, K. Williamson, P. Clark, S. Woods, Ontology reuse and application, in: Proceedings of the International Conference on Formal Ontology and Information Systems (FOIS'98), Trento, Italy, 1998, pp. 179–192.

[3] J. Bao, D. Caragea, V. Honavar, Modular ontology – a formal investigation of semantics and expressivity, in: Proceedings of the First Asian Semantic Web Conference (ASWC 2006), Beijing, China, 2006, pp. 616–631.

[4] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, H. Stuckenschmidt, Contextualizing ontologies, Journal of Web Semantics 1 (4) (2004) 325–343.

[9] http://www.geneontology.org
[10] http://ontology.teknowledge.com/

[5] B. Grau, B. Parsia, E. Sirin, A. Kalyanpur, Modularizing OWL ontologies, in: Proceedings of the Workshop on Ontology Management, 3rd International Conference on Knowledge Capture (K-CAP 2005), Banff, Canada, 2005.

[6] H. Alani, S. Harris, B. O'Neil, Ontology winnowing: A case study on the AKT reference ontology, in: Proceedings of the International Conference on Intelligent Agents, Web Technology and Internet Commerce (IAWTIC'2005), Vienna, Austria, 2005, pp. 710–715.

[7] N. Noy, M. Musen, Specifying ontology views by traversal, in: Proceedings of the Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, 2004, pp. 713–725.

[8] H. Stuckenschmidt, M. Klein, Structure-based partitioning of large class hierarchies, in: Proceedings of the Third International Semantic Web Conference (ISWC 2004), Hiroshima, Japan, 2004, pp. 289–303.

[9] H. Pinto, J. Martins, Reusing ontologies, in: AAAI Spring Symposium on Bringing Knowledge to Business Processes, 2000, pp. 77–84, Stanford University, California, March 20-22.

[10] H. Alani, Ontology construction from online ontologies, in: Proceedings of the 15th World Wide Web Conference, 2006, pp. 491–495.

[11] E. Bontas, M. Mochol, R. Tolksdorf, Case studies on ontology reuse, in: Proceedings of the 5th International Conference on Knowledge Management (I-Know05), Graz, Austria, 2005, pp. 345–353.

[12] W. Qin, Y. Suo, Y. Shi, CAMPS: A Middleware for Providing Context-Aware Services for Smart Space, Vol. 3947 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2006, pp. 644–653.

[13] M. Uschold, M. King, Towards a methodology for building ontologies, in: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing in conjunction with IJCAI-95, Montreal, Canada, 1995.

[14] Y. Ding, Semi-automatic generation of resilient data-extraction ontologies, Master's thesis, Brigham Young University, Provo, UT (June 2003).

[15] D. Embley, Y. Jiang, Y.-K. Ng, Record-boundary discovery in Web documents, in: Proceedings of the 1999 ACM International Conference on Management of Data (SIGMOD'99), Philadelphia, Pennsylvania, 1999, pp. 467–478.

[16] D. Embley, L. Xu, Record location and reconfiguration in unstructured multiple-record Web documents, in: Proceedings of the Third International Workshop on the Web and Databases (WebDB2000), Dallas, TX, 2000, pp. 123–128.

[17] K. Mahesh, Ontology development for machine translation: Ideology and methodology, Tech. Rep. MCCS-96-292, Computer Research Laboratory, New Mexico State Univeristy (1996).

[18] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.

[19] S. Liddle, D. Embley, S. Woodfield, Cardinality constraints in semantic data models, Data & Knowledge Engineering 11 (3) (1993) 235–270.

[20] Y. Ding, D. Embley, S. Liddle, Automatic Creation and Simplified Querying of Semantic Web Content: An Approach Based on Information-Extraction Ontologies, Vol. 4185 of Lecture Notes in Computer Science, Springer, Berlin/Heidelberg, 2006, pp. 400–414.

[21] D. Lonsdale, Y. Ding, D. W. Embley, A. Melby, Peppering knowledge sources with SALT: Boosting conceptual content for ontology generation, in: Semantic Web meets Language Resources: Papers from the AAAI Workshop, AAAI Press, Menlo Park, CA, 2002, pp. 30–36, Technical Report WS-02-16.

[22] B. C. Grau, I. Horrocks, Y. Kazakov, U. Sattler, Extracting modules from ontologies: A logic-based approach, in: OWL: Experiences and Directions: Third International Workshop (OWLED), 2007, June 6-7, Innsbruck, Austria.