Annotating Documents for the Semantic Web Using Data-Extraction Ontologies

A Dissertation Proposal
Presented to the
Department of Computer Science
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

by
Yihong Ding
September 14, 2005

1 Introduction

The gigantic volume of web content forces people to rely on machines to help search for information. Search engines help but not enough. The limitations, however, are caused more by current web content representations than by the search technology itself. It is now time to improve web representations so that they are machine understandable. Representing web content by *ontologies*, which are commonly shared, explicitly defined, generic conceptualizations [Gru93], makes the content machine understandable. The web with machine understandable content is known as the *semantic web* [BLHL01].

We can see the problem in more detail by considering the limits of current search engines. Current generic search engines, like *Google* [goo], do a phenomenal job of ranking billions of web pages and identifying potentially useful candidates, often presenting the page a user wants within the first few search results. Some domain-specific search engines can work even better on their specialized domains. For example, NEC's *CiteSeer* [cit] helps people find scientific literature. The problem, however, is not what they do but what they cannot do. Keyword-based search technologies restrict the types of questions people can ask. For example, they cannot make a request like, "Give me the lowest price flight from Salt Lake City to Tucson during the next four weeks." The required information is on the web, but search engines are unable to answer this type of question because they do not know how to match the specified concepts in the request to data instances in web content. In general, machines do not know how to match instances because current web representations do not provide any explanation of their content. The current web is designed for humans, not for machines.

A straightforward solution to this problem is to design a new type of machine understandable web representation and develop web pages based on the new format. A dilemma arises, however, about how to deal with current web pages. There are billions of pages in the current web, and not very many web developers are willing to rewrite their pages according to some new standard, especially if it would require tedious manual labeling of documents.

Research on automatically or semiautomatically annotating documents for the semantic web is an attempt to resolve this dilemma. The goal of web semantic annotation research is to add metadata to web content so that it becomes machine understandable. Unlike an annotation in the normal sense, which is an unrestricted note, a semantic annotation must be explicit, formal, and unambiguous: *explicit* makes a semantic annotation publicly accessible, *formal* makes a semantic annotation publicly agreeable, *unambiguous* makes a semantic annotation publicly identifiable. These three properties enable machine understanding.

The history of semantic annotation can be traced back to ancient Greece when the great Greek philosopher Aristotle described ontology as "the science of being qua being" [Ari93], i.e., the study of "that which is" in terms of "that which is." For many years after Aristotle, philosophers studied how ontologies can specify facts in the world in an unambiguous way. In this sense, the history of semantic annotation is the history of ontologies. A new age of semantic annotation started in the year 1999 when Tim Berners-Lee proposed the idea of the semantic web [BL99, W3Cc]. Semantic annotation becomes an important way to enable the semantic web by annotating current web content with formal specifications of conceptualizations, which are ontologies.

A typical semantic annotation process includes three components. First, an ontology describes a domain of interest. Second, a data instance recognition process discovers all instances of interest in target web documents based on the defined ontology. Third, an annotation generation process creates a semantic meaning disclosure file for each annotated document. Through the semantic meaning disclosure file, any ontology-aware machine agent can understand the target document.

Currently, there are three ways people have tried to do web semantic annotation. First, people have tried to manually annotate web pages with the help of visual interfaces. These systems allow users to make arbitrary notes manually and share them through a network. When the notes are restricted to be from an ontology, we can use the system to semantically annotate web documents. The main research focus of this type of approach, however, has been on optimized annotation storage and sharing mechanisms instead of on annotation methods. Annotea is a representative web annotator of this category. Annotea uses an RDF-based [W3Cb] annotation schema for describing annotations and XPointer [xpt] for locating the annotations in the annotated document [KKPS01]. The RDF-based annotation schema is a simple version of an ontology. [HLO99] has surveyed many other manual annotation systems [cri, GSM99, Hol96, OAM99, RMW97, thi], which could potentially be used for semantically annotating the web. Because of the huge volume of web pages, however, it is very unlikely that people can use manual semantic annotation tools to convert pages on the current web to be pages for the semantic web. Automated semantic annotation tools are a necessity.

The second approach is to do automatic semantic annotation through an automatic ontology generation process. Automatic ontology generation processes usually use machine learning techniques to classify data instances [DF02]. This paradigm, if it would work, would directly lead to semantic annotation results by assigning the generated semantic categories to the corresponding data instances. Although researchers have studied many different ways to do automatic ontology generation [DF02], SCORE (Semantic Content Organization and Retrieval Engine) is the only system to have been tried for semantic annotation. SCORE employs multiple metadata extractor agents to extract entities using regular-expression-based crawling rules from both semi-structured and unstructured text [SBA$^+$02]. It then applies both classification-based methods and knowledge-base methods to identify entity hierarchies and relationships. Its advantage is that it does not require pre-defined ontologies for target web pages, but because of the difficulties of ontology generation research [DF02], its practicality has not been proven to be successful and is still under study.

The third way to do semantic annotation, and the way which is currently the most popular, is to annotate web pages using pre-defined ontologies [DEG$^+$03, HSC02, KPT$^+$04, VVMD$^+$02]. This approach assumes that ontologies already exist so that it avoids the very difficult ontology generation problem. These annotation tools depend on automatic information extraction (IE) tools to help locate concept instances. Once the information is extracted, these annotation tools then assign semantic categories to the extracted concept instances based on pre-defined domain ontologies.

Although results are encouraging for this third type of semantic annotator, there are still problems that need exploration. There are three procedures and four main issues that we must study carefully in this paradigm. Figure 1 shows the three basic procedures: (1) extraction, (2) alignment, and (3) annotation. First, an annotation tool uses its adopted/adapted IE wrapper to extract

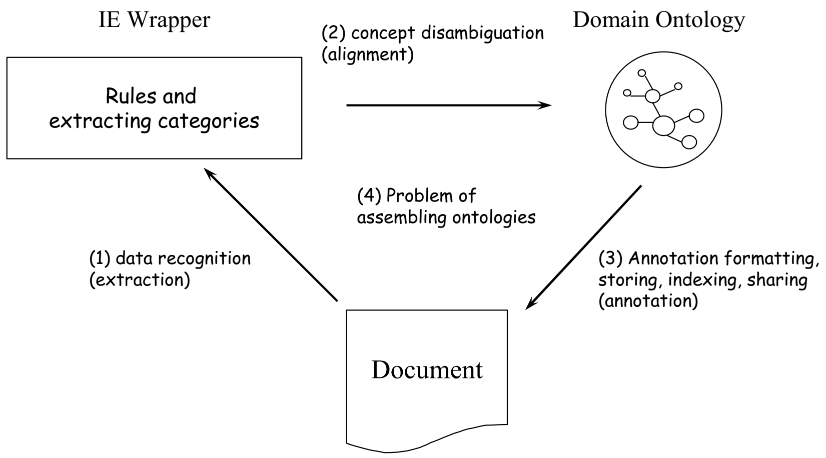

Figure 1: Paradigm and Four Main Issues for The Third Type of Semantic Annotator.

data instances from a target document. This procedure relates to the traditional problem of data recognition in data extraction research. Although many researchers have worked on this problem [LRNdST02], it is still largely unsolved. After doing data extraction, the annotation tool needs to align the extraction categories in the IE wrapper with the concepts defined in the domain ontologies. Both [SR03] and [KPT$^+$04] discuss this problem. This is called "the problem of concept disambiguation [SR03]." In [KPT$^+$04], the authors suggest that we need to integrate domain ontologies with extraction engines to avoid the problem altogether and proposes this as a future direction for study. As the final step, the annotation tool takes care of the problems of annotation formatting, storing, indexing, and sharing. There is no standard agreement on any of these procedures, and they are open to exploration. Thus, these three procedures also represent three research issues we need to study. We also need to study the problem of creating a domain ontology in the first place; otherwise, there can be no semantic annotation since a machine can only understand annotations with respect to a domain ontology.

For my dissertation work I plan to address all these research issues and build a prototype of a new semantic annotator. The approach I plan to pursue is to use data-extraction ontologies [ECJ+99]. Since data-extraction ontologies use ontologies to represent extraction categories, as Figure 2 shows, we can combine[1] the problem of data recognition and concept disambiguation into one problem and simplify the structure of the semantic annotation problem.

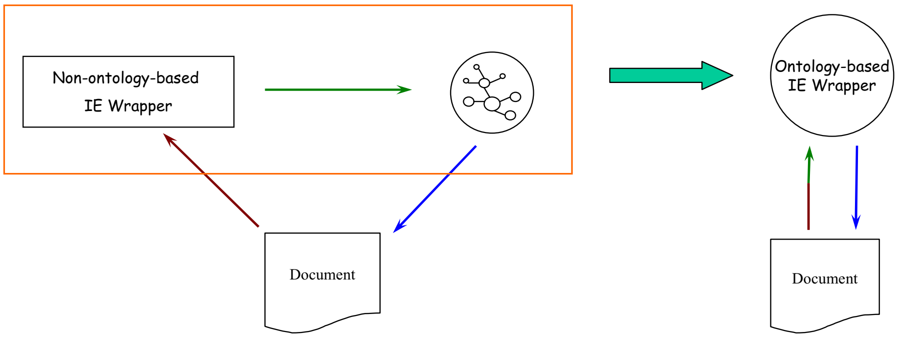

Figure 2: New Paradigm of Semantic Annotators Using Ontology-based IE Engine.

The basic semantic annotation prototype using data-extraction ontologies requires three major augmentations to make it work for the semantic web: (1) representation compatibility, (2) execution speed, and (3) a streamlined way of assembling an annotation domain ontology. The first major augmentation is to convert data-extraction ontologies to semantic web ontologies in OWL (Web Ontology Language) [W3Ca]. This is an important step to make our work compatible with the rest of the world. With this augmentation, not only will we be able to provide annotation in a standard language, but we will also be able to represent data-extraction ontologies in OWL so that others can easily adopt them for their own purposes. In addition, we will provide a tool so that other people can use their own OWL ontologies to do semantic annotation without knowing the original data-extraction ontology language, which is called OSMX (Object-oriented Systems Model in XML) [WLE05].

Another major augmentation is to provide a two-layer annotation model to achieve both resiliency with respect to document layout and a high speed of execution. The lower layer conceptual

[1] [KPT+04] suggests that we may avoid the problem of concept disambiguation by integrating ontologies with IE engines. Based on our experiences with data-extraction ontologies, however, we believe that we cannot totally eliminate the concept disambiguation problem, though it is indeed simplified. Instead, we consider the problem of concept disambiguation with the traditional problem of data recognition to be a new problem of concept and instance identification.

annotator will use the original ontology-based IE wrapper to do annotation so that the system is resilient to document layout (see [ECJ+99] for a full explanation of this resiliency problem). The upper layer annotator is to be dynamically generated from the output of the lower layer conceptual annotator. It will make use of fixed structural information so that it can execute with high speed to annotate a large number of similar documents. This will enable the annotation to scale up with respect to document size.

The third and last major augmentation is to dynamically assemble an annotation domain ontology when there is no pre-constructed ontology available for an annotation task. This mechanism depends on a pre-constructed collection of knowledge that contains existing ontologies, ontology snippets, and individual data frames. When users do not specify a domain ontology for an annotation task, the system will try to match the semantics within the collection of knowledge to the content of a presented web page. The system then assembles the matched pieces from the collection of knowledge to create a new domain ontology for a user's task.

2 Thesis Statement

Given the impetus to develop the semantic web and provide semantic annotations as described in the introduction and given the current state of the art in semantic annotation, we believe that we can build a semantic annotation system with the following properties. It will be resilient to variety in web page layouts, will have higher execution speed, and possibly better accuracy on a large number of web pages comparing with the current data-extraction tool, and it will be compatible with current semantic web standards. When there is no existing ontology for an annotation task, the system will also be able to suggest a set of ontology components that have been previously used and that are applicable to the domain of interest. Users (possibly with some help from the system) may assemble these components to create a new ontology.

3 Research Description

According to the thesis statement, we discuss four issues to be resolved in this dissertation work: (1) automatic semantic annotation, (2) ontology language conversion and OWL extensions, (3) performance, and (4) ontology construction.

3.1 Automatic Semantic Annotation

As mentioned in Section 1, there are two types of automatic semantic annotation tools: either accompanied with automatic ontology generation or accompanied with automatic information extraction. Because of the difficulty of ontology generation, and also because of our previous experiences with data-extraction ontologies, we only plan to focus on creating semantic annotation tools using automatic information extraction (IE) techniques.

Figure 2 shows the generic architecture of semantic annotation tools that use automated IE engines. There are two basic processes in the paradigm—extraction and annotation. Given a document, the extraction process finds information of interest according to the defined domain ontology, and the annotation process creates explicit annotations for the document based on the results from the extraction process.

3.1.1 Extraction

Information extraction (IE) engines are the base platform of the type of semantic annotators that use automated information extraction techniques. Researchers have studied automatic information extraction for over a decade. In the most recent survey of automatic IE tools [LRNdST02], the authors present six different categories of IE engines. We can study the advisability of using these six types of IE tools in automatic semantic annotation and how suitable they are for annotating web content if we adopt them.

1. Wrapper languages provide a formal interpretation for defining extraction patterns [AM98, CM98, HMGM97, HFAN98, LHL$^+$98]. Wrapper languages are hard to design. Extraction patterns require manual specification. Moreover, different web pages usually require the specification of individual extraction patterns. It is thus obvious that this IE approach is not suitable for automatic semantic annotation. Indeed, no current semantic annotation researchers have tried to adapt this type of IE tool in their system.

2. HTML-aware IE tools extract data of interest based on pre-defined HTML layout descriptions [AGM03, BFG01, CL01, CMM01, LPH00, SA01]. Because these tools only need one pass of a document to extract data in pre-described positions inside documents, they perform with high execution speed. These tools may achieve accuracy as high as nearly 100% if a target document fits the described HTML layout. If a target document does not fit the described HTML layout, these tools cannot extract data correctly. If the layout changes, the wrappers need to be regenerated to fit the new layout. Another characteristic of this type of IE tool is that they are constructed so that they find the location of data of interest, not the semantic meaning of the data with respect to an ontology. Thus, this approach requires a step to label or align the data extracted with an ontology. [ACMM03] describes an attempt to do semantic annotation with this type of IE tool. The authors, however, point out that automatic labeling is hard, and based on their results we can see that it is not very successful.

3. NLP- (natural language processing)-based IE tools analyze text using NLP techniques, such as filtering, part-of-speech tagging, and lexical semantic tagging [CM99, Fre00, Sod99]. They work well for unstructured, grammatical web documents, for instance, the news domain. NLP-based IE tools, however, do not work well for fully structured documents, for semi-structured

documents, or for telegraphic text documents. This is because these documents do not contain enough grammatical sentences for NLP techniques to work. There are three major current semantic annotation approaches adapt this type of IE tool for their system [HSC02, VVMD+02, KPT+04]. Like HTML-aware tools, however, these annotation tools need to align extracted information with whatever domain ontology they use for semantic annotation. [KPT+04] explains, saying, "The main drawback ... is that none of these approaches expects an input or produces output with respect to ontologies. [Thus,] ... a set of heuristics for post-processing and mapping of the IE results to an ontology ... [is needed.] However, such heuristics are not sufficient for large-scale, domain-independent semantic annotation." The authors then argue that "IE and wrapper induction techniques need to use the ontology more directly during the process of extraction."

4. ILP- (Inductive Learning Processing)-based tools do data extraction from fully structured and semi-structured documents [HD98, Kus00, MMK01]. Instead of linguistic constraints, ILP-based tools learn the formatting features that implicitly delineate the structure of data found on a page. Therefore, pure ILP-based tools do not work well for unstructured documents because of the lack of formatting information. Another problem is that machine learning processes usually need a large set of training documents, which must be labeled, usually manually. There is one semantic annotation approach that uses an ILP-based IE technique [DEG+03]. Similar to HTML-aware and NLP-based IE tools, an ontology is not part of the input. Thus, the same alignment issues exist as for HTML-aware and NLP-based IE tools.

5. Modeling-based tools adapt supervised machine learning approaches to do data extraction [Ade98, LRND02]. After manually or semi-automatically selecting training documents, users guide the wrapper generation process through an interactive user-interface to create a data-extraction model. Because this IE technique requires human-guided supervised learning, it is hard to scale this type of tool for large applications. Probably because of this limitation, no researchers have tried to use this type of IE tool to annotate textual web pages.

6. Ontology-based IE tools apply pre-defined data-extraction ontologies to do data extraction [AKM+03, DYKR00, ECJ+99, MNS02]. Since this type of tool extracts information with respect to an ontology, no alignment of concepts is necessary. No previous approach for semantic annotation has attempted to use ontology-based IE tools. This dissertation is the first attempt. The following paragraphs explain the arguments for and against using ontology-based IE tools to do semantic annotation.

Besides the benefit of eliminating the alignment between the concepts in domain ontologies and extraction categories in IE engines, there are two other major advantages for using ontology-based IE tools to do semantic annotation.

- First, an ontology-based IE tool is resilient to the layouts of web pages. A data-extraction ontology defines conceptual recognizers of domain concepts and also conceptual relationships along with participation constraints among the concepts. Because all the information is defined at the conceptual level, not at the page layout level, we can apply a data-extraction ontology to any web page that contains information for the target domain. Note that, except for NLP-based IE tools, no other IE tools have this resiliency property; thus, when the structure of web pages change or new pages are encountered, wrappers have to be rebuilt [LMK03].
- Second, an ontology-based IE tool is adaptive to changes in domain information. When people want, for example, to add a new extraction category to an existing wrapper, they can directly add a concept into a data-extraction ontology without changing existing concepts. This is, however, not true for many other IE tools. When the content of target domain changes, most of the time, users have to regenerate the whole wrapper to reflect the changes.

There are two arguments against ontology-based IE tools. Critics of this work have complained about the dependence on ontologies and have thought that the creation of ontologies is an extra unnecessary burden to do data extraction. Even though these criticisms make some sense for pure data extraction tasks, they definitely no longer hold for semantic annotation applications because domain ontologies are required in any case. Another argument against using ontology-based IE tools is that they execute comparatively slower than many other IE tools because they take a lot of computational time to enumerate every possibility of matching concept recognizers to data instances within documents. One of my proposed augmentations, the two-layer annotation model, is designed to solve this problem.

3.1.2 Annotation

How should we record and store annotations for the semantic web? Although it is likely to be straightforward to adapt our work proposed here to any set of guidelines provided by the semantic web community, we know of no current, agreed-upon guidelines. Some researchers, however, have considered this question and have provided suggestions. In [vHF99] the authors describe several characteristics a practical knowledge representation standard should have: syntactically, it must (1) be supported by current web technologies, (2) avoid duplication, and (3) allow nesting; and semantically, it must (1) be able to represent facts within a data model, (2) be able to refer to formal terminologies (i.e. ontologies), and (3) have a sound theoretical basis for inference. Agreeing with these suggestions, we have adapted OWL [W3Ca] as our annotation language. OWL is supported by current web technologies, allows nesting since it is based on XML, allows developers to avoid duplication so long as their specifications satisfy database-like normal forms established for XNF (e.g. [AL04, EM01, VLL04]), is designed to represent a formal ontology, and has a sound theoretical basis (OWL-DL, for example, is the description logic version of OWL). This leaves, as the only remaining issue, the representation of facts, which we discuss in this section.

For representing facts, we emphasize that the goal of an annotation process is (1) to explicitly bind each data instance of interest in a web page to its correct semantic category within an ontology and (2) to correctly capture the ontological relationships among these data instances. Generally speaking, we can see two ways to achieve this goal: *explicit annotation*, which adds special tags that bind tagged instances in a web page to an externally specified ontology, and *implicit annotation*, which adds nothing explicit to the document, but instead extracts instance position information as well as the data instances and stores them in an externally specified ontology. [HH00] gives an early example of explicit annotation (most others have followed this lead), and [HNS$^+$04] gives an example of implicit annotation.

Using explicit annotation, we have created an online demo [DEG] of our semantic annotation tool. Within the demo, we assign a four-tuple in a SPAN tag for every recognized data instance x. This four-tuple uniquely identifies (1) the ontology used for annotation (in case there are several for the same document), (2) the concept within the ontology to which x belongs, (3) the record number for x so that the system knows which values relate together to form a record, and (4) a value number within the record in case more than one instance of the concept can appear within a record. The SPAN tag is a standard HTML tag, which browsers ignore when there is no style sheet; it provides a generic container for text that can be used for the purpose of adding semantics, as we do. For example, if we were doing an application for car advertisements, an explicit annotation <SPAN class=(CarAds,Mileage,13,0)>117K</SPAN> means the following: *117K* is the annotated content, *CarAds* is the domain ontology, *Mileage* is the concept, *13* is the record number, and *0* is the value number. In addition to the SPAN tags surrounding all data instances of interest to an ontology, we also add a META tag in the header that links an ontology name, *CarAds* for our example, with its URL. SPAN annotations along with the URL specifying the ontology allow the system to create the equivalent of a populated semantic ontology for each annotated page.

Using implicit annotation, we directly augment ontologies with instances. Within the augmented ontologies, we record each data instance with its local offset within the document. At the same time, we cache a copy of the web page so that we can guarantee that the instance position information is correct. Also, we give a unique identifier to each data instance and group the instances together when they are within one record.

Each of these two annotation techniques has its own benefits and drawbacks. Explicit annotation favors situations in which web pages frequently change but only with minor modifications—for example, daily price updates of items for sale. Thus, if updates are done correctly, neither anyone nor any annotation system needs to re-annotate the changed document because the updated content is still within proper explicit tags. This advantage does not hold for implicit annotation because updates may change the length of strings and thus the offset positions of data instances. For explicit annotation to work, of course, the annotation system must be able to write explicit tags in web pages owned by others; thus, issues about who controls what and who has which permissions must be addressed and resolved.

Implicit annotation favors situations in which web pages change infrequently or change in ways that would destroy explicit annotation and also favors situations in which we may wish to annotate a page using more than one ontology. Because we cache web pages when we use implicit annotation, the information may not be current. We can, of course, obtain the current page and re-annotate it, but this can be costly if re-annotation must be frequent to keep the page up to date. If we annotate a web page with several ontologies, implicit annotation is likely to be better because the multiple explicit tags can cause confusion and may even require illegal tag nesting: $<tag_1>$ $data_1$ $<tag_2>$ *common data* $</tag_1>$ $data_2$ $</tag_2>$.

The last thing we want to point out is that although it is possible to query both explicit and implicit annotations, it is more convenient to query implicit annotations because all values are in the same XML document. Thus, for example, we can issue an XQuery statement to query implicit annotations but not to query explicit annotations.

3.2 Ontology Conversion and OWL Extensions

When adapting ontology-based IE tools to do semantic annotation, we use ontologies to both recognize data instances and represent semantic annotations. Historically, these two problems have been studied by different groups of researchers. This has led to the problem that ontologies designed for information extraction purposes are neither syntactically nor semantically the same as ontologies designed for representing knowledge for the semantic web; although, they do have many common features since both of them are explicit specifications of conceptualizations. Our purpose is to unite these two types of ontologies as one so that we have uniform ontology representations to both do annotation (through extraction) and represent annotation results (through annotation disclosure files).

The two ontology languages we wish to work with are OSMX (Object-oriented Systems Model in XML) and OWL. OSMX ontologies support data extraction [ECJ+99], and OWL is a W3C-recommended ontology language specifically for representing knowledge for the semantic web [W3Ca]. For most of the rest of the world, OWL is an example of a “standard” ontology language. Within an OWL ontology are concepts, relationships among concepts, and cardinality constraints over relationships. With its basis of description logics [BCM+03], it is possible to properly reason using OWL ontologies. Similarly, OSMX has concepts, relationships among concepts, and cardinality constraints over relationships, and it is also possible to build reasoning mechanisms on top of OSMX because of its logic foundation, which is a subset of first-order logic. Beyond these similarities, OSMX allows users to describe formal semantics about data recognition for individual concepts. OWL has no corresponding data recognition capability, and thus, cannot do information extraction.

Because of the popular acceptance of OWL, we need to choose OWL ontologies to represent annotation results, and we see it as advantageous to give OWL the capability to do information extraction so that the entire annotation process can be represented in OWL. Therefore, we face two challenges. First, we need to convert OSMX to OWL ontologies so that we can represent the results of OSMX information extraction as OWL annotation disclosure files. Second, we need to convert OWL augmented with information extraction features to OSMX ontologies so that users need not to learn two ontology languages. Also, at least for the prototype implementation, we see this as the most practical way to create an ontology-based wrapper generator for OWL.

Many transformations between OSMX and OWL are syntactic only. Both languages model domains using the subject-object-predicate data model with only name differences. For example, an *object set* in OSMX is a *class* in OWL; a *relationship set* in OSMX is an *ObjectProperty* in OWL; a *participation constraint* in OSMX is a *Cardinality* in OWL; an *isa* hierarchical relationship in OSMX is a *subClassOf* specialization in OWL.

On the other hand, each ontology language has unique specifications on some semantics that are not straightforward to convert to the other language. For example, OWL allows users to specify subproperties so that one property may subsume another. In OSMX, although we can still model this type of *is-a* relationship among relationship sets, we have no built-in way to represent it. Instead, we have to use a general constraint in OSMX to accomplish this task. As another example, OSMX allows users to directly declare arbitrary n-ary relationship sets, whereas OWL permits only binary relationships, represented as properties. To resolve this discrepancy, we need to decompose an n-ary relationship set, when n is greater than 2, to multiple binary properties when converting OSMX to OWL. As the third example, OWL supports multiple namespaces within one ontology so that users can refer to concepts inside another ontology. Currently, OSMX does not support namespace declarations, and we need to augment OSMX so that it does, probably by adopting the same style namespace declaration as in OWL.

A special requirement of the conversion from OSMX to OWL is how to handle the data recognition semantics. Within an OSMX ontology, users can specify recognition patterns, context patterns, indicative keywords, and canonicalization functions to describe data recognition semantics. Since standard OWL does not support these features, we must extend it to represent this information. Figure 3 is an example of how we describe these semantics. To augment OWL, we plan to create an *osmx* namespace so that we can directly import data recognition semantics into OWL.

3.3 Performance

There are three performance issues we need to consider: accuracy, speed of execution, and resiliency of the annotation tool. High accuracy is a performance requirement for any system, and so is speed. Because there are so many web pages to annotate, slow execution, even with high accuracy, is problematic. Resiliency is another important issue for semantic annotation tools: we want the

```
<owl:Class rdf:about=“Year”>
    <osmx:DataFrame>
        <osmx:ValuePhraseList>
            <osmx:ValuePhrase />
                <rdfs:comment>2 digit year</rdfs:comment>
                <osmx:ValueExpression>
                    <osmx:ExpressionText>\d{2}</osmx:ExpressionText>
                </osmx:ValueExpression>
                <osmx:LeftContextExpression>
                    <osmx:ExpressionText>\'</osmx:ExpressionText>
                </osmx:LeftContextExpression>
                <osmx:RightContextExpression>
                    <osmx:ExpressionText>\b</osmx:ExpressionText>
                </osmx:RightContextExpression>
            </osmx:ValuePhrase />
            ...
        </osmx:ValuePhraseList>
    </osmx:DataFrame>
</owl:Class>
```

Figure 3: Example of a Representation of Data Frames in OWL.

annotation process to work continuously without the overhead of regenerating extraction patterns according to the change of web page layouts, and we want the same annotation process to work immediately with new pages within the domain when they come online.

With this motivation, we are building an improved semantic annotation tool that is resilient, executes fast, and has high accuracy. Figure 4 shows a two-layer annotation model that can accomplish these goals. The lower layer conceptual annotator uses an ontology-based IE tool. Hence the conceptual annotator will be resilient to changes of web pages and will be able to work immediately on web pages within the domain when they come online. The upper layer structural annotator uses a layout-based IE tool. Since layout-based IE tools execute quickly, and, when properly constructed, have high accuracy, the structural annotator will also execute quickly and have high accuracy.

According to our previous research on data-extraction ontologies [ECJ+99], we know that the conceptual annotator can be resilient and can have fairly good accuracy for recognizing data instances for its defined semantic categories. The main problems are its execution speed and sometimes its accuracy. As a trade-off to achieve resiliency, the conceptual annotator requires a large number of computation cycles to enumerate all possible candidate instances and resolve the ambiguities. Hence, its execution speed cannot be guaranteed to be fast. Also, although it is designed to achieve good accuracy in general cases, it does not take into account local structural patterns, which can lead to higher extraction accuracy.

Unlike the conceptual annotator, a structural annotator performs its task quickly because it

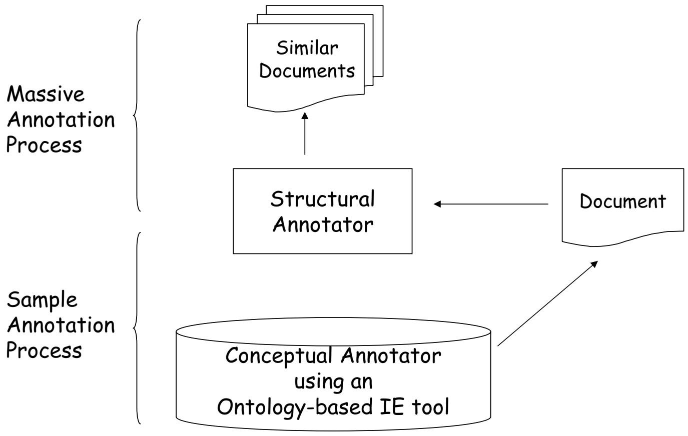

Figure 4: Two-Layer Annotation Model.

requires only a single pass through a document. A structural annotator also assures high accuracy because it processes web pages that match a known layout structure. As a trade-off, structural annotators are not resilient because they fail to annotate correctly when layouts are unknown or change.

In the system we are proposing we wish to dynamically construct a structural annotator when the conceptual annotator annotates a web page. Using the results that the conceptual annotator generates, the system can determine where ontological concepts are located on the page. To generate layout patterns, the system uses the identified locations of concepts to do a structural analysis to create layout patterns that can uniquely lead to the concept locations. In principle, a layout pattern may contain any useful information to help uniquely decide the locations of concepts. In practice, there are three typical layout patterns that are most helpful: (1) an HTML hierarchical path that leads to concept locations, (2) local context around locations, and (3) dependencies (e.g., sequence and co-occurrence) among multiple semantic categories.

The system combines the identified semantic categories, their locations and layout patterns to be a structural annotator. When the results are imprecise or incomplete, the system needs to provide a user interface to let users correct errors and add new semantic categories and layout patterns as may be needed. After users make corrections (if any), the system regenerates the structure annotator according to the modifications.

Almost all the HTML-aware IE tools referenced in Section 3.1.1 contain a layout-structure generation algorithm. The most significant difference between our approach and theirs is that our system not only finds the extraction categories and layout patterns, but can also correctly identify the semantic meanings of the extraction categories with respect to an ontology. After construction, HTML-aware IE tools rely on users to specify the meanings of their extraction categories with respect to an ontology. The researchers who built RoadRunner [CMM01], one of the HTML-aware IE tools, have tried to adapt their system to do semantic annotation. They apply an image recognition algorithm to find the closest categoric name of an extraction category from the graphic view of a web page [AGM03]. Even though their algorithm was reported to be successful in finding the labels, they still could not automatically match the semantic meanings of the labels to an ontology. In the conclusion of their paper, they suggest that the combination of their image recognition algorithm with the data-extraction ontologies studied at BYU could be a proper solution to their problem. Our approach proposed here follows up on this suggestion and shows how the approaches can be successfully combined.

3.4 Ontology Construction

Until now, we have assumed that domain ontologies are pre-constructed. Manual construction of ontologies is tedious and error-prone. To overcome this obstacle, some researchers have attempted to find ways to automatically or semi-automatically construct ontologies [DF02]. This research has demonstrated that automatic generation of ontologies from scratch is difficult. Rather than generate ontologies from scratch, we propose a mechanism that can maximize the reuse of existing ontologies and minimize the work of constructing new ontologies.

Our proposal is to provide a way to assemble a domain ontology for a web page when there is no input ontology for an annotation task. The pre-assumption for assembling ontologies is that there exists a collection of knowledge that contains ontologies, snippets of ontologies, and single data frames. Our hope is that if some part of (or in the best case all of) the required formal semantics has already been built, we want to adopt the existing semantics to avoid the work of reconstructing it from scratch. Therefore, the work turns out to be determining how the system can find useful knowledge components and assemble them together to become the domain ontology that describes the information in a web page presented by the user. Figure 5 illustrates our solution.

The collection of knowledge plays the role of a known knowledge base in our paradigm. It not only consists of ontologies, snippets of ontologies, and data frames, but it also stores any previously determined mapping information among these components. The difference between this proposed collection and an integrated knowledge base is that we do not plan to enforce any integration until it is demanded. Furthermore, we want to make the collection flexible, able to handle different scenarios, because enforcing one way of integrating may prohibit other ways of integrating.

Our observation about reusing existing knowledge is based on the fact that many "new" domains are not so new that they have never been described before. "What has been will be again, what has been done will be done again; there is nothing new under the sun."[2] An ontology for a "new" domain may be a combination of unions and projections of several existing ontologies. For example,

[2] The Holy Bible, Ecclesiastes 1:9.

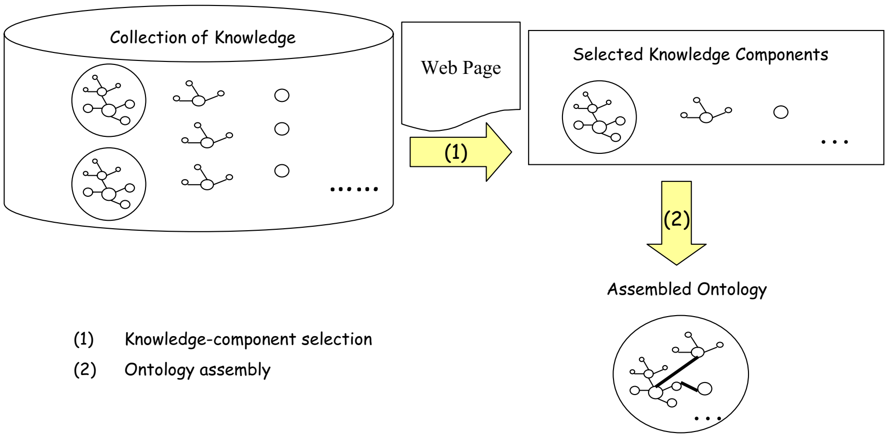

Figure 5: Architecture on Dynamically Assembling Domain of Interest.

if the collection of knowledge contains an ontology about professors and another ontology about company organizations, a new ontology about university employees is basically a subset of the union of these ontologies where part of the information, like *Terminal Degree*, *University of Terminal Degree*, *Research Specialities*, and *Teaching Specialities*, can be directly adopted from the domain of professors, and other information, like *Supervisor*, *Position*, and *Department*, can be retrieved from the ontology of company organizations. The problem then revolves around two questions. (1) How does the system retrieve components of knowledge with respect to the web page presented and then assemble these components to be the target domain ontology? (2) How does a user create new knowledge and update the collection with the newly created knowledge when some part of or even the whole target domain is beyond the collection of knowledge?

When a user presents a web page without an accompanying domain ontology, the system needs to select components of existing formal semantics that describe the web page content. Since each semantic component is either an extraction ontology, a snippet of an extraction ontology, or a single data frame, the system can invoke their data recognizers to decide whether the component or part of the component is contained in the target domain. Once a knowledge component is selected, the system projects on the relevant part of the component (if necessary) and tries to combine it with the other selected components. During the assembling process, the system may use stored mapping information to help. The system may also be able to apply a mapping algorithm [EXD04] to help generate additional needed mappings among selected components.

Although this selecting and assembling paradigm seems straightforward, there are many problems we need to explore. For example, how do we solve ambiguities when multiple mappings exist? How can we find and eliminate semantic conflicts? How do we provide tools for users to connect components when no mapping information is available? We plan to do our best to propose solutions to these questions, but this research work is not likely to be able to resolve all of them. Instead, our main focus on this part of the research is to build a framework system and use it to discover as many problems and resolutions as we can.

When users find information within the presented web page that cannot be covered by the existing formal semantics in the collection of knowledge, they may have to build it from scratch. In these cases, the system refers users to our ontology development tool [WLE05]. It is always the case that someone must be the first to create the formal semantics for some "new" knowledge that has never before been created. When users build new ontologies, ontology snippets, or data frames, they may add them into the collection of knowledge. Similarly, when users specify mappings among components or verify system-generated mappings among components, the system also adds this new mapping information to the collection of knowledge. The philosophy for updating the collection of knowledge is that the system stores all previously used, user-provided information. Thus, the collection of knowledge grows naturally as it is used.

3.5 Generic Semantic Annotation: An Overall Picture

Having discussed all the proposed work, we can finally obtain a clear picture of the full semantic annotation paradigm we are proposing. When a new web page is presented for annotation, the system either finds an appropriate domain ontology to do the annotation for the page, or the system semi-automatically assembles a domain of interest for the annotation work. An assembled domain ontology will work not only for the web page presented, but also for other web pages within the same application domain.

Once the system has both domain ontologies and target web pages as inputs, it can annotate pages using the conceptual annotator. When there are enough pages from the same site to make it worthwhile, the system semi-automatically builds structural annotators based on layout patterns. It then can use these created structural annotators to annotate web pages faster and more accurately.

The system represents annotation results in OWL. Our proposed OWL-to-OSMX converter lets users directly input OWL domain ontologies to the annotation tool; and the OSMX-to-OWL converter lets the system convert any OSMX input to OWL semantic annotations. The augmentation of OWL with data recognition semantics provides a standard way to represent information extraction in OWL.

4 Research Plan

I plan to build a semantic annotation prototype system that illustrates my proposed research. This prototype system will be able to automatically annotate web pages with given domain ontologies. I will run experiments against real web sites that contain rich information to evaluate the performance of this system. Specifically, I plan to process my research in three stages: (1) semantic annotation prototype system set-up, (2) system augmentations, and (3) dynamic ontology assembler plug-in.

4.1 Prototype System Set-up

The data-extraction ontology approach is a unique ontology-based IE approach developed at BYU. We use its extraction engine, which is named *Ontos* [WLE05], as the base extraction platform for the new semantic annotation prototype system.

The prototype system set-up stage includes converting the *Ontos* extraction engine to output semantic meaning disclosure files, which are ontologies, instead of the original relational tables. We have already done much of this work and currently the prototype system generates both implicit annotations (in which extracted values are stored in a file along with pointers into a cached web page) and explicit annotations (in which semantic tags are added to a cached web page). We have also created an online demo of the semantic annotation prototype system to illustrate how pages are annotated and how semantic annotations may appear. The demo runs online and is also downloadable at the DEG web site [DEG].

4.2 Prototype System Augmentations

After we set up the basic semantic annotation platform, we plan to add two major augmentations to improve system performance: the ontology converter and the two-layer annotator.

First, we plan to do an exhaustive comparison of syntax and semantics between OWL and OSMX. The comparison should show the strength and weakness of the two languages with respect to their knowledge specification, knowledge reasoning, and knowledge instantiation capabilities. Based on the comparison, we will propose an augmentation of OWL to better satisfy the requirements for semantic annotation and implement an ontology converter that transforms OSMX ontologies and OWL ontologies from one language to the other and vice versa.

We would like to guarantee that our transformation to and from OWL and OSMX are correct. One expectation might be that a transformation followed by a reverse transformation would yield the original. When we convert an ontology from one language to the other and then transform it back, however, we are not likely to syntactically get the original one back after the two transformations. Thus, we would still have to check semantic equivalence. A complete and sound algorithm to check semantic equivalence of two syntactically different ontologies is costly and difficult to implement (yes, "implement" because the proof would need to be produced mechanically). Since in our case we are only concerned about semantic annotation tasks, we can perform a simple test to check the correctness of the ontology converter, even though it is not sound and complete.

The simple test is as follows. Suppose we start with an OSMX ontology A. First we convert it to OWL and then transform it back to OSMX yielding another OSMX ontology B. Instead of directly checking whether all the constructs in A and B are equivalent, we apply both A and B to annotate a common set of web pages. If the two ontologies are semantically equivalent, they must yield the same annotations for every individual web page. When the set is large enough, say 30 – 50 pages, we can be reasonably confident that the converters are good enough.

When we implement the two-layer annotation model, *Ontos* becomes the lower layer conceptual annotator. Based on the recognized annotations within sample web pages, we plan to write (or adapt one of [BFG01] or [CMM01] to be) a structural similarity detector that generates common patterns for instances among the annotated sample documents. Using the detected common patterns, the system can create a structural annotator that handles the layout that fits the sample documents. This structural annotator becomes the upper layer of the annotation model.

To test the performance, we plan to evaluate both the accuracy of the annotation results and the execution speed. In particular, we plan to annotate apporximately five large web sites with machine-generated web pages, each of which contains dozens if not hundreds of web pages, and compare the precision, recall, and speed of the dynamically generated structural annotators with the precision, recall, and speed of the conceptual annotator. Both the accuracy and the speed should be better than that of using only the lower layer annotator based on the original *Ontos* engine. Further, we would like to compare both speed and accuracy issues with the results of the other published annotators using their studied web sites. However, this comparison depends on the information available in their published papers or on whether we can obtain permission to use their annotation tools.

4.3 Dynamic Ontology Assembler

To build the dynamic ontology assembler, we need to construct a collection of knowledge. The collection should contain at least twenty domain ontologies and some ontology snippets and individual data frames so that there is a large enough base to make assembling possible and reasonable. Beyond the collection of knowledge, we need to implement an algorithm to select and sort out individual knowledge components with respect to arbitrary web pages. *Ontos* or the extraction part of the two-layer annotation model will provide the base for this algorithm. We also need to implement an algorithm to assemble selected knowledge components to be a unified ontology. We may adapt part of our previous work on attribute match discovery [EJX02] to help with this part of the implementation.

We plan to do two types of experiments to evaluate this part of the work. First, we plan to do a regular precision and recall study to examine how precise and how complete the knowledge component selection algorithm is for retrieving knowledge components out of the collection of knowledge with respect to a given web page. Secondly, our previous study [Din03] shows that under certain conditions, it may be better to write an ontology from scratch rather than automatically generate it from a knowledge base. Therefore, we plan to do a pilot study on how much effort a user must expend to improve an automatically assembled ontology until it becomes a final version and how we can improve the chances that favor automatic generation by improving the organization of the knowledge base. We will record the time it takes to create an ontology from scratch and the time it takes to assemble an equivalent ontology using our tool. We can then observe which conditions favor assembling which favor manually writing from scratch. We also hope that we can get some initial empirical evidence to show that when conditions are right, assembling an ontology takes much less effort than building from scratch. Furthermore, from [Din03] we know that a highly integrated, broad-coverage, non-IE-oriented knowledge base (i.e. Mikrokosmos [Mah96]) is not helpful for automatically generating a domain ontology. On the other hand, a knowledge base with too many highly decomposed ontology components requires a tremendous amount of integration. There must be a balance between these two extremes that favors automatic ontology assembly, and we want to use our pilot study to help find this balance.

4.4 Deliverable Artifacts

When the dissertation is finished, the following artifacts are expected to be available:

- a prototype system for annotating web documents based on data-extraction ontologies, including
  - a web page annotator,
  - an implementation of our proposed 2-layer annotation model, and
  - a tool to help assemble and create extraction ontologies
- a collection of knowledge that contains existing ontologies, ontology snippets, and data frames;
- a transformation tool that converts OSMX ontologies and OWL ontologies from one to the other; and
- a proposed augmentation of OWL about knowledge recognition semantics.

4.5 Delimitations of the Dissertation

In this dissertation, I assume that the following are not part of this research study:

- automatic ontology creation from scratch;
- annotation storing, indexing, and sharing infrastructure;
- semantic annotation for multimedia content; and
- parallel or distributional computational techniques to further scale the semantic annotation system to a large number of web pages.

5 Research Papers

The following are proposed titles of papers.

- Semantic Web Annotation—Using Ontology-based Information Extraction
- Highly Accurate and Resilient Semantic Annotation—A Combination of Data-Extraction Ontologies and Layout-based Wrappers
- Dynamically Assembling Domain Ontologies Using a Collection of Knowledge
- The Comparison between OSMX and OWL—A Theoretic Study
- OWL2OSMX: Directly Using OWL Ontologies to Do Semantic Annotation

6 Contribution to Computer Science

My vision is to develop an annotation method to convert current web pages into machine-processable semantic web pages. The new method uses an ontology-based IE engine to avoid the extra overhead of aligning extraction categories in IE wrappers with concepts in domain ontologies. This research also combines the resiliency of ontology-based IE techniques with the fast and highly accurate layout-based IE techniques using a novel two-layer annotation model. The dynamic domain ontology assembler helps maximize the reuse of existing knowledge and minimize the load of manual ontology creation. Finally, the conversions between OSMX and OWL link this semantic annotation work to the rest of the semantic web society.

7 Dissertation Schedule

The following schedule shows the expected milestone date for each step of my dissertation work.

- Basic ontology-based web semantic annotation tool (transformed from original Ontos tool) (April 2005)
- The OWL-OSMX converter that converts one to the other (including the theoretic proof of the equivalence between the two) (August 2005)
- Implementation of two-layer annotation model (December 2005)
- Implementation of dynamic ontology assembler (April 2006)

References

[ACMM03] Luigi Arlotta, Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Automatic annotation of data extracted from large web sites. In *Proceedings of the Sixth International Workshop on the Web and Databases (WebDb 2003)*, pages 7–12, San Diego, California, June 2003.

[Ade98] B. Adelberg. Nodose – a tool for semi-automatically extracting structured and semistructured data from text documents. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 283–294, Seattle, Washington, 1998.

[AGM03] Arvind Arasu and Hector Garcia-Molina. Extraction structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 337–348, San Diego, California, June 2003.

[AKM+03] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21, January/February 2003.

[AL04] M. Arenas and L. Libkin. A normal form for XML documents. *ACM Transactions on Database Systems*, 29(1):195–232, March 2004.

[AM98] G. O. Arocena and A. O. Mendelzon. WebOQL: Restructuring documents, databases, and webs. In *Proceedings of the 14th International Conference on Data Engineering*, pages 24–33, Orlando, Florida, 1998.

[Ari93] Aristotle. *Metaphysics* (about 350 BC). Oxford University Press, New York, 1993.

[BCM+03] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[BFG01] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 119–128, Rome, Italy, 2001.

[BL99] Tim Berners-Lee. *Weaving the Web*. Harper SanFrancisco, San Francisco, California, 1999.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 284(5):34–43, May 2001.

[cit] Citeseer paper search. http://citeseer.ist.psu.edu.

[CL01] Chia-Hui Chang and Shao-Chen Lui. IEPAD: Information extraction based on pattern discovery. In *Proceedings of the Tenth International Conference on World Wide Web*, pages 681–688, Hong Kong, China, May 2001.

[CM98] Valter Crescenzi and Giansalvatore Mecca. Grammars have exceptions. *Informaion Systems*, 23(8):539–565, 1998.

[CM99] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence*, pages 328–334, Orlando, Florida, 1999.

[CMM01] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB)*, pages 109–118, Rome, Italy, 2001.

[cri] The critlink mediator. http://www.doc.ic.ac.uk/~ids/dotdot/misc/research_related/AnnotatingTheWeb/CritSuite/crit-0.9.2-ids-0.0.1-as-cgi-script/web/.

[DEG] Homepage for BYU Data Extraction Group. http://www.deg.byu.edu.

[DEG+03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Kevin S. McCurley, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. A case for automated large scale semantic annotations. *Journal of Web Semantics*, 1(1):115–132, December 2003.

[DF02] Ying Ding and Schubert Foo. Ontology research and development: Part 1 a review of ontology generation. *Journal of Information Science*, 28(2):123–136, February 2002.

[Din03] Yihong Ding. Semiautomatic generation of resilient data-extraction ontologies. Master's thesis, Computer Science Department, Brigham Young University, 2003.

[DYKR00] Hasan Davulcu, Guizhen Yang, Michael Kifer, and I. V. Ramakrishnan. Computational aspects of resilient data extraction from semistructured sources. In *Proceedings of the Nineteenth ACM International Symposium on Principles of Database Systems (PODS)*, pages 136–144, Dallas, Texas, May 2000.

[ECJ+99] David W. Embley, Douglas M. Campbell, Yuan S. Jiang, Stephen W. Liddle, Yiu-Kai Ng, Randy D. Smith, and Deryle Lonsdale. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowledge Engineering*, 31(3):227–251, 1999.

[EJX02] David W. Embley, David Jackman, and Li Xu. Attribute match discovery in information integration: Exploiting multiple facets of metadata. *Journal of the Brazilian Computing Society*, 8(2):32–43, November 2002.

[EM01] David W. Embley and Wai Y. Mok. Developing XML documents with guaranteed 'good' propoerties. In *Proceedings of The 20th International Conference on Conceptual Modeling (ER2001)*, pages 426–441, Yokohama, Japan, November 2001.

[EXD04] David W. Embley, Li Xu, and Yihong Ding. Automatic direct and indirect schema mapping: Experiences and lessons learned. *SIGMOD Record*, 33(4):14–19, December 2004.

[Fre00] D. Freitag. Machine learning for information extraction in information domains. *Machine Learning*, 39(2/3):169–202, May 2000.

[goo] Google Search Engine. http://www.google.com.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[GSM99] Geri Gay, Amanda Sturgill, and Wendy Martin. Document-centered peer collaborations: An exploration of the educational uses of networked communication technologies. *Journal of Computer-Mediated Communication*, 4(3), March 1999. online Journal at http://www.ascusc.org/jcmc/vol4/issue3/gay.html.

[HD98] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[HFAN98] G. Huck, P. Fankhauser, K. Aberer, and E. J. Neuhold. Jedi: Extracting and synthesizing information from the web. In *Proceedings of the 3rd International Conference on Cooperative Information Systems (IFCIS)*, pages 32–43, New York City, New York, 1998.

[HH00] J. Heflin and J. Hendler. Dynamic ontologies on the web. In *Proceedings of Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449, Menlo Park, California, July-August 2000.

[HLO99] Rachel M. Heck, Sarah M. Luebke, and Chad H. Obermark. A survey of web annotation systems. Technical report, Grinnell College, Grinnell, Iowa, 1999.

[HMGM97] J. Hammer, J. McHugh, and H. Garcia-Molina. Semistructured data: The tsimmis experience. In *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems*, pages 1–8, St. Petersburg, Russia, 1997.

[HNS+04] L. Hollink, G. Nguyen, G. Schreiber, J. Wielemaker, B. Wielinga, and M. Worring. Adding spatial semantics to image annotations. In *Proceedings of the Fourth International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2004)*, Hiroshima, Japan, November 2004. in conjunction with ISWC 2004.

[Hol96] Koen Holtman. The futplex system. In *Proceedings of the ERCIM workshop on CSCW and the Web*, Sankt Augustin, Germany, February 7-9 1996.

[HSC02] Siegfried Handschuh, Steffen Staab, and Fabio Ciravegna. S-CREAM — Semi-automatic CREAtion of Metadata. In *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW-2002)*, pages 358–372, Madrid, Spain, October 2002.

[KKPS01] Jos Kahan, Marja-Riitta Koivunen, Eric Prud'Hommeaux, and Ralph R. Swickd. Annotea: An open RDF infrastructure for shared web annotations. In *Proceedings of The Tenth International World Wide Web Conference*, pages 623–632, Hong Kong, China, May 1-5 2001.

[KPT+04] Atanas Kiryakov, Borislav Popov, Ivan Terziev, Dimitar Manov, and Damyan Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Sematics*, 2(1):49–79, December 2004.

[Kus00] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal*, 118(1-2):15–68, 2000.

[LHL+98] B. Ludascher, R. Himmeroder, G. Lausen, W. May, and C. Schlepphorst. Managing semistructured data with FLORID: A deductive object-oriented perspective. *Information Systems*, 23(8):589–613, 1998.

[LMK03] Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. 18:149–181, Feburary 2003.

[LPH00] L. Liu, C. Pu, and W. Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Proceedings of the 16th International Conference on Data Engineering (ICDE)*, pages 611–621, San Diego, California, 2000.

[LRND02] A. H. F. Laender, B. Ribeiro-Neto, and A. S. DaSilva. Debye – data extraction by example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[LRNdST02] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, June 2002.

[Mah96] K. Mahesh. Ontology development for machine translation: Ideology and methodology. Technical Report MCCS-96-292, Computer Research Laboratory, New Mexico State Univeristy, 1996.

[MMK01] I. Muslea, S. Minton, and C. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[MNS02] Alexander Maedche, Gunter Neumann, and Steffen Staab. Bootstrapping an ontology-based information extraction system. In *Szczepaniak, Piotr S.; Segovia, Javier; Kacprzyk, Janusz; Zadeh, Lotfi A. (eds), Intelligent Exploration of the Web*, pages 345–360. Springer, 2002.

[OAM99] Ilia A. Ovsiannikov, Michael A. Arabib, and Thomas H. McNeill. Annotation technology. *International Journal of Human-Computer Studies*, 50(4):329–362, April 1999.

[RMW97] Martin Roscheisen, Christian Mogensen, and Terry Winograd. Shared web annotations as a platform for third-party value-added information providers: Architecture, protocols, and usage examples. Technical report, Stanford University, Stanford, California, 1997.

[SA01] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data and Knowledge Engineering*, 36(3):283–516, 2001.

[SBA+02] Amit Sheth, Clemens Bertram, David Avant, Brian Hammond, Krysztof Kochut, and Yashodhan Warke. Managing semantic content for the web. *IEEE Internet Computing*, 6(4):80–87, July/August 2002.

[Sod99] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, February 1999.

[SR03] Amit Sheth and Cartic Ramakrishnan. Semantic (Web) technology in action: Ontology driven information systems for search, integration and analysis. *IEEE Data Engineering Bulletin*, 26(4):40–48, December 2003.

[thi] Thirdvoice web annotator. http://c2.com/cgi/wiki?ThirdVoice.

[vHF99] Frank van Harmelen and Dieter Fensel. Practical knowledge representation for the web. In *Proceedings of the IJCAI-99 Workshop on Intelligent Information Integration*, Stockholm, Sweden, July 31, 1999.

[VLL04] M. W. Vincent, J. Liu, and C. Liu. Strong functional dependencies and their application to normal forms in XML. *ACM Transactions on Database Systems*, 29(3):445–462, September 2004.

[VVMD+02] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt, and Fabio Ciravegna. Mnm: Ontology driven tool for semantic markup. In *Proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002) with ECAI 2002*, pages 43–47, Lyon, France, July 2002.

[W3Ca] W3C (World Wide Web Consortium) *OWL Web Ontology Language Reference*. http://www.w3.org/TR/owl-ref/.

[W3Cb] Resource Description Framework. http://www.w3.org/RDF/.

[W3Cc] W3C (World Wide Web Consortium) *Semantic Web Activity Page*. http://www.w3.org/2001/sw/.

[WLE05] A. Wessman, S.W. Liddle, and D.W. Embley. A generalized framework for an ontology-based data-extraction system. In *Proceedings of the 4th International Conference on Information Systems Technology and its Applications (ISTA'05)*, pages 239–253, Palmerston North, New Zealand, May 2005.

[xpt] XML Pointer Language (XPointer). http://www.w3.org/TR/WD-xptr.

This dissertation proposal by Yihong Ding is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the dissertation proposal requirement in the Department of Computer Science for the degree of Doctor of Philosophy.

David W. Embley, Committee Chair

Deryle W. Lonsdale, Committee Member

Stephen W. Liddle, Committee Member

William A. Barrett, Committee Member

Eric G. Mercer, Committee Member

Parris Egbert, Graduate Coordinator