Semantic Annotation Based on Extraction Ontologies

Yihong Ding¹, David W. Embley¹, and Stephen W. Liddle²

¹Department of Computer Science
²Information Systems Department
Brigham Young University, Provo, Utah 84602, U.S.A.
ding@cs.byu.edu, embley@cs.byu.edu, liddle@byu.edu

Abstract
The “semantic web” represents a major advance in web utility, but it is currently difficult to create semantic-web content because pages must be semantically annotated through processes that are mostly manual and require a high degree of engineering skill. We must therefore devise means for transforming existing, non-semantic web pages into semantic web pages. We propose using information extraction ontologies to handle this challenge. In this paper we show how a successful ontology-based data-extraction technique can automatically generate semantic annotations for ordinary, data-rich web pages. A unique characteristic of this approach is the use of extensional semantics inside ontologies to help specify annotation domains and perform data recognition. We have implemented a prototype of our approach to demonstrate that our proposal works.

1 Introduction

The sheer volume of web content forces people to rely on machines to help search for information. Search engines help, but by themselves are not enough. Search engines like Google, for example, do a good job ranking billions of web pages and identifying useful candidates, often presenting the page a user wants within the first few search results. The problem, however, is not what search engines do, but what they cannot do. Keyword-based searching restricts the types of questions people can ask. For example, users cannot make requests like, “Find me a red Nissan for under $5000 – it should be a 1990 or newer and have less than 120K miles on it.” The required information is out there on the web, but search engines cannot answer this type of question because they do not know how to match the specified concepts in the request to data instances on the web.

A solution to this problem is to design a new type of machine-understandable web representation and develop web pages based on the new format, or in other words develop the semantic web
[7]. Semantic-web proponents propose making web content machine understandable through the use of ontologies, which are commonly shared, explicitly defined, generic conceptualizations [28]. But then one of the immediate problems we face is how to deal with current web pages. There are billions of pages on the current web, and it is impractical to ask web developers to rewrite their pages according to some new, semantic-web standard, especially if this would require tedious manual labeling of documents.

Web semantic annotation research attempts to resolve this problem. The goal of web semantic annotation is to add comments 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, and unambiguous makes a semantic annotation publicly identifiable. These three properties enable machine understanding, and annotating with respect to an ontology makes this possible. In this paper we show how to automatically annotate existing data-rich web pages with respect to an ontology.
To clarify our intentions, we give an example. Figure 1 shows two ordinary, human-readable web pages for selling cars. Our system can annotate these pages automatically with respect to a given ontology about car advertisements and thus can convert them to semantic web pages so that these pages also exist in machine-readable form. We store these annotations in such a way that we can directly query them using an available query language (XQuery [51] for our particular implementation). This entire process allows us to query the content of web pages not originally designed for the semantic web, thus, a request equivalent to “Find me a red Nissan for under $5000 – it should be a 1990 or newer and have less than 120K miles on it” over the pages in Figure 1 would yield results such as those in Figure 2. The results in Figure 2 are actual answers to the query in a table whose header attributes are the concept names from the given car-ads ontology, restricted to those concepts mentioned in the query. In addition, there is always one additional attribute, Source, whose values are links back into the original documents at the location where the information is provided. When a user clicks on Car01 (the link in the first row in Figure 2), for example, the document in Figure 1 from the Athens site appears, except it would be scrolled to the right place and the information requested in the query would be highlighted.

We give the details of our contribution of automatically creating semantic web content so that we can directly query it as follows. Section 2 describes information-extraction (IE) ontologies, which are the basis for our automated semantic-web annotation tool. We emphasize the role of extensional semantics in addition to intensional semantics as a means to declaratively direct information extraction. Section 3 argues that IE ontologies may well provide the best alternative for automatically annotating much of the data-rich content of the current web. We make this argument by surveying related work and showing how the approach based on IE ontologies resolves

<table>
<thead>
<tr>
<th>Color</th>
<th>Make</th>
<th>Price</th>
<th>Year</th>
<th>Mileage</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>......</td>
<td>NISSAN</td>
<td>$4,500</td>
<td>1993</td>
<td>117K</td>
<td>Car01</td>
</tr>
<tr>
<td>......</td>
<td>NISSAN</td>
<td>$900</td>
<td>'93</td>
<td></td>
<td>Car13</td>
</tr>
</tbody>
</table>

Figure 2: Query Results.
problems others have faced. Section 4 describes our prototype work on automatically annotating existing web pages so that they can be used for the semantic web, and Section 5 shows how we can directly query pages annotated for the semantic web. Section 6 provides experimental evidence about the accuracy of our annotation system. We also critique its current limitations and describe the extensions needed to make it practical. We conclude in Section 7.

2 Ontologies for Semantic Annotation

In semantic web applications, ontologies describe formal semantics for applications, and thus make information sharable and machine-understandable. The work of semantic annotation is, however, more than just knowledge representation. Semantic annotation applications must also establish mappings between ontology concepts and data instances within documents so that these data instances become sharable and machine-understandable. In this section, we introduce information-extraction ontologies and show that they are useful both for representing knowledge and for establishing mappings between ontology concepts and document data instances.

Guarino explains that the ontology space is characterized by a hierarchy [29]. There are top-level ontologies, which can be specialized into domain and task ontologies, and application ontologies, which are specialized versions of domain and/or task ontologies. Information-extraction ontologies are more specialized than the typically broad top-level ontologies that many people associate with the term “ontology”.

2.1 Information Extraction Ontologies

We have described information-extraction ontologies elsewhere [22], but to make our paper self-contained, we briefly reintroduce them here. We have used extraction ontologies in a number of applications, including information extraction [22], high-precision classification [25], and schema mapping for ontology alignment [50]. In this paper we show how to use extraction ontologies for semantic web annotation.

An extraction ontology specifies named sets of objects, which we call object sets or concepts, and named sets of relationships among object sets, which we call relationship sets. Figure 3 shows
a graphical rendition of an extraction ontology for car advertisements. The extraction ontology has two types of concepts: lexical concepts (enclosed in dashed rectangles) and nonlexical concepts (enclosed in solid rectangles). A concept is lexical if its instances are indistinguishable from their representations. Mileage is an example of a lexical concept because its instances (e.g., “117K” and “5,700”) represent themselves. A concept is nonlexical if its instances are object identifiers, which represent real-world objects. Car is an example of a nonlexical concept because its instances are identifiers such as, say, “Car01”, which represents a particular car in the real world. An extraction ontology also provides for explicit concept instances (denoted as large black dots). We designate the main concept in an extraction ontology by marking it with “-►” in the upper right corner, which denotes that the object set Car becomes (“-►”) an object instance (“•”) for a single car ad.

Figure 3 also shows relationship sets among concepts, represented by connecting lines, such as the connecting line between Car and Year. The numbers near the connections between relationship sets and object sets are participation constraints. Participation constraints give the minimum and maximum participation of an object in an object set with respect to the connected relationship set. For example, the 0:1 participation constraint on Car in the Car-Mileage relationship set
denotes that a car need not have a mileage in a car ad, but if it does, it has only one. A white triangle defines a generalization/specialization relationship, with the generalization concept connected to the apex of the triangle and one or more specialization concepts connected to its base. In Figure 3, for example, Feature is a generalization of Engine and BodyType, among others. The white triangle can, of course, appear repeatedly, and thus we can have large ISA hierarchies in an extraction ontology. A black triangle defines an aggregation with the super-part concept connected to the apex of the triangle and the component-part concepts connected to its base. In Figure 3, for example, ModelTrim is an aggregation of Model and Trim. Like ISA hierarchies, large PartOf hierarchies are also possible.

As a key feature of extraction ontologies, the concepts each have an associated data frame. A data frame describes information about a concept—its external and internal representations, its contextual keywords or phrases that may indicate the presence of an instance of the concept, operations that convert between internal and external representations, and other manipulation operations that can apply to instances of the concept along with contextual keywords or phrases that indicate the applicability of an operation [19]. Figure 4 shows sample (partial) data frames for the concepts Price and Make in our ontology for car advertisements. As Figure 4 shows, we use regular expressions to capture external representations. The Price data frame, for example, captures instances of this concept such as “$4500” and “17,900”. A data frame’s context keywords are also regular expressions. The Price data frame in Figure 4, for example, includes context keywords such as “asking” and “negotiable”. In the context of one of these keywords in a car ad, if a number appears, it is likely that this number is a price. The operations of a data frame can manipulate a concept’s instances. For example, the Price data frame includes the operation LessThan that takes two instances of Price and returns a Boolean. The context keywords of an operation indicate an operation’s applicability; context keywords such as “less than” and “<”, for example, apply to the LessThan operation. Sometimes external representations are best described by lexicons or other reference sets. These lexicons or reference sets are also regular expressions, often simple lists of possible external representations, and can be used in place of or in combination
Price
  
  **internal representation:** Real
  
  **external representation:** \$?(\d+ | \d\d\d\d)\d{0,2}n
  
  **context keywords:** price | asking | obo | neg(\.,otiable) | ...
  
  LessThan(p1: Price, p2: Price) **returns** (Boolean)
  
  **context keywords:** less than | < | or less | fewer | ...
  
end

Make
  
  **external representation:** CarMake.lexicon
  
end

Figure 4: Sample data frames for car ads ontology.

with regular expressions. In Figure 4, *CarMake.lexicon* is a lexicon of car makes, which would include, for example, “Toyota”, “Honda”, and “Nissan” and potentially also misspellings (e.g. “Volkswagon”) and abbreviations (e.g. “Chev” and “Chevy”).

We can apply an extraction ontology to obtain a structured representation of the unstructured information in a relevant document. For example, given the car-ads extraction ontology and one of the Nissan ads in Figure 1:

  '93 NISSAN Model XE, $900, Air Conditioning, new tires, for listings, call 1-800-749-8104 ext. V896.

we can extract “93” as the Year, “NISSAN” as the Make, “XE” as the Model, “$900” as the Price, both “Air Conditioning” and “new tires” as Features with “Air Conditioning” also being an Accessory, and “1-800-749-8104” as the PhoneNr. As part of the extraction, the conversion routines in the data frames convert these extracted values to a canonical internal representation, so that, for example, “93” becomes the integer 1993 and “$900” becomes the real number 900.

Although often simple, as illustrated here, there are sometimes subtle problems that can make extraction ontologies fail to extract correctly.

- Ambiguities can occur. For example, when there are two prices (e.g., list price and selling price).
price). In this case, we use heuristic rules to sort out the ambiguities [22].

- Correct identification of each individual record in a document can be difficult. This requires work on record separation [23], and sometimes requires the system to recognize that it must distribute factored information, such as a dealer telephone number that applies to many records, or that it must split records, such as when multiple cars sold by one dealer appear as a single ad, or that it must access off-page information under a link [24].

- The data is often behind forms. In this case the system must first fill in the form and obtain the information from the hidden web [10, 39].

- The hidden web—and sometimes the visible web too—presents data in a relatively structured form, such as a table. Surprisingly, certain types of structure cause more difficulties for our extraction ontologies than might be expected, and special techniques are needed to handle these types of structured information [27].

### 2.2 Extensional Semantics in Annotation Ontologies

We claim that information-extraction ontologies are well positioned to satisfy the requirements of semantic annotation. Not only do they provide the intensional-level semantics found in typical ontologies, but they also provide the extensional-level semantics needed to connect individual data items found in ordinary web pages with the typical intensional-level semantics.

A typical ontology for the semantic web, e.g. an OWL (Web Ontology Language) ontology [43], defines concepts, properties, and restrictions on concepts and properties to model a conceptual domain. So do our information-extraction ontologies. In our information-extraction ontologies, object sets are concepts, relationship sets are properties, and participation constraints are typical restrictions on object sets within specific relationship sets. Therefore, our information-extraction ontologies satisfy the basics of a typical ontology. Figure 3 shows our intensional perspective in graphical form.

Typically, current ontologies for the semantic web focus on specifying intensional semantics. But we also need the extensional perspective, which is often missing in these ontologies. Since the
goal of semantic annotation work is to determine correct bindings of data instances to ontology concepts, we also need extensional semantics. These extensional semantics describe the denotation of a lexical concept, i.e., the set of lexical objects characterized by the concept. We provide this extensional perspective with our data frames.

Figure 4 exemplifies the fundamental idea. A data frame describes the external representations of lexical strings that belong to a concept. We write these descriptions as instantiation patterns using regular expressions. Added to these instantiation patterns, we provide regular expressions for context and keyword phrases, which aid in correctly classifying instantiation patterns that may be similar in several different data frames.

There are several benefits of using extensional semantics in annotation ontologies.

- Extensional semantics let developers provide a complete and precise semantic description of their understanding of an annotation domain. By using extensional semantics in annotation ontologies, we deliver not only abstract concepts and their logical relationships, but also the possible concrete instantiations of these concepts when they are lexical.

- With built-in extensional semantics, we no longer need to rely on separate programs to assert the binding of a data instance to its ontological concept. Indeed, for the data frames in our ontologies, since we provide extensional semantics declaratively, we only need one program to assert bindings. This same program works for all lexical concepts in all ontologies.

- Extensional semantics is the key to resiliency. We are careful to ontologically declare the actual semantics of a lexical concept, as opposed to using page layout or chance appearances of surrounding symbols to identify a concept. Therefore, our instance recognizers continue to work when page layouts change and when surrounding symbols change. And, perhaps even more important, these instance recognizers work on new or not-yet-considered pages within the same domain and sometimes even in different domains. This helps make ontology-driven annotation scalable.

- Extensional semantics inside ontologies help provide a degree of integrity. We can use data
frames to “type-check” values for a concept. Thus, for example, we would not mistakenly annotate “Taurus” as a *CarMake*, but rather correctly annotate it as a *CarModel*. Humans, as well as ontologies armed with extensional semantics, can catch these kinds of errors, but they are likely to be hard, if not impossible, for independent machine agents to catch and resolve.

### 3 Semantic Annotation

Having explained what an extraction ontology is, we now turn our attention to applying extraction ontologies to annotating pages for the semantic web. We begin by first arguing that extraction ontologies are likely to be the best approach among current proposals for converting HTML web pages into semantic web pages. We make our argument by surveying the current literature and explaining why extraction ontologies are likely to be superior.

The history of semantic annotation can be traced back to ancient Greece when the philosopher Aristotle described ontology as “the science of being *qua* being” [3], 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. With the emergence of the semantic web [5, 48], semantic annotation has become an enabling technology—a way to annotate current web content with formal ontologies.

A typical semantic annotation process includes three components. First, an ontology must be created to describe a domain of interest. Second, a data instance recognition process discovers 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.

Researchers have suggested three different approaches to annotating pages for the semantic web. The first approach is to manually annotate web pages with the help of visual interfaces. An-
notea is a representative annotator of this category [35], and [32] has surveyed many other manual annotation systems that could be used for semantically annotating the web. Although useful for small numbers of pages, it is highly unlikely that we could use manual semantic annotation tools to convert the current web to be the semantic web—the web is just too large. Thus, we require automated semantic annotation tools to do the job. The second approach is to do automatic semantic annotation through an automatic ontology generation process. Automatic ontology generation processes typically use machine learning techniques to classify data instances [17]. This paradigm, if it could work, would lead directly to semantic annotation results by assigning the generated semantic categories to the corresponding data instances. Although researchers have studied many different ways to generate ontologies automatically [17], SCORE (Semantic Content Organization and Retrieval Engine) is the only system to have been tried for semantic annotation [44]. Because of the difficulties of ontology generation [17], however, the practicality of this approach 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 [4, 16, 31, 36, 42, 46]. 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 with the basic procedures. Figure 5 shows the three basic procedures: (1) extraction, (2) alignment, and (3) annotation. Although researchers have neither fully resolved the issues with the first procedure nor decided on the best solution for the third procedure, it is the second procedure that has become the most critical for those attempting to adapt IE tools to annotate current web pages for the semantic web. It is nontrivial to align the extraction categories in an IE wrapper with the concepts defined in semantic-web ontologies. Sheth et al. [45] and Kirykov et al. [36] both discuss this problem. Sheth names this “the problem of concept disambiguation” [45]. Kiryakov suggests that we need to integrate domain ontologies with extraction engines to avoid the problem altogether and proposes this as a direction for future study [36]. Indeed, this is the approach we propose here. Since data-extraction ontologies use ontologies to represent extraction categories, as Figure 6 shows, we can combine\(^1\) the problems of data recognition and concept disambiguation into one problem and simplify the structure of the semantic annotation problem.

Before turning our attention to the issue of creating a semantic meaning disclosure file for annotated web documents, we briefly continue our discussion of adapting IE tools as the base platform for semantic annotation tools. We do so to (1) recognize the work of several who have

\(^1\)Although Kiryakov et al. [36] suggest that we may avoid the problem of concept disambiguation by integrating ontologies with IE engines, based on our experiences with data-extraction ontologies, 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 simply be the problem of concept and instance identification.
attempted to adapt IE tools for semantic annotation and (2) to further discuss the issues with the
question in mind of using the suggestion of [36] to directly extract into semantic-web ontologies.
In so doing, we consider IE tools as categorized in a recent survey [37].

1. *Wrapper Languages.* These IE tools require manual specification, and thus will not scale.
No current semantic annotation researchers have tried to use this type of IE tool in their
system.

2. *HTML-aware IE tools.* These IE tools extract data of interest based on pre-defined HTML
layout descriptions. Recently, Arlotta et al. [4] described their attempt to do semantic
annotation with this type of IE tool. The base tool for this adaptation is RoadRunner [11],
but RoadRunner only finds the location of data of interest, not the semantic meaning of the
data with respect to an ontology. This necessitates labeling or aligning the data extracted
with an ontology, a task which the authors point out is not easy [4].

3. *NLP-based IE tools.* These IE tools analyze text using NLP (Natural Language Processing)
techniques such as filtering, part-of-speech tagging, and lexical semantic tagging. There are
several current efforts to adapt these IE tools for semantic annotation [31, 36, 46]. Unfortu-
nately, the current construction of these tools also suffers from the need to align extracted
information with whatever domain ontology is used for semantic annotation. Kiryakov et al.
[36] explain, 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].” Another limitation
of NLP-based IE tools is that they only work well with sentential text; they do not work well
with telegraphic text or with structured or semi-structured data, which constitutes much of
the data-rich web.

4. *ILP-based tools.* Instead of linguistic constraints, ILP (Inductive Learning Processing) based
tools learn the formatting features that implicitly delineate the structure of data found in a
page. Dill et al. [16] showed how to use this type of IE tool for semantic annotation. Similar
to HTML-aware and NLP-based IE tools, an ontology is not part of the input, and thus ILP-based IE tools have the same postprocessing alignment issues. Furthermore, machine learning processes usually need a large set of training documents, which must be labeled, usually manually.

5. **Modeling-based IE tools.** These IE tools adapt supervised machine learning approaches to do data extraction. 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 likely to be hard to scale for large applications. Although some researchers have tried to use this technique to annotate multimedia web content (e.g. [34]), no researchers have tried to use this type of tool to annotate textual web pages of the type we are considering here.

6. **Ontology-based IE tools** (e.g. [1, 14, 22, 41]). These IE tools apply pre-defined extraction ontologies to perform data extraction. Since this type of tool extracts information with respect to an ontology, no alignment of concepts is necessary. Besides the benefit of eliminating the alignment between concepts in domain ontologies and extraction categories in IE engines, ontology-based IE tools also have the benefit of being resilient to the layouts of web pages and immediately work with new web pages in the same domain. These features allow this technique to scale up because extraction ontologies neither need to be rewritten nor regenerated for pages that change or for new pages within the domain. Finally, we point out that although critics of ontology-based IE tools have complained (and perhaps rightly so) about the dependence on expert-generated ontologies, and have thought that the creation of ontologies is an unnecessary extra burden to do data extraction, this complaint no longer holds for semantic annotation applications because domain ontologies are required in any case.
4 IE-Based Semantic Web 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. To record and store annotations, we face two representation issues: (1) representation of intensional knowledge and (2) representation of extensional information.

4.1 Intensional Representation

Some researchers have considered the question of representing intensional knowledge for purposes of semantic annotation and have provided suggestions. In [30] 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 adopted OWL 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. [2, 26, 47]), is designed to represent a formal ontology, and has a sound theoretical basis (description logics). This leaves, as the only remaining issue, the representation of facts, which we discuss below.

Before explaining how we represent facts, we show a small snippet of the OWL ontology generated from the extraction ontology in Figure 3. Figure 7 shows some of the components corresponding to the Mileage and Engine concepts declared in Figure 3. Observe in Figure 7 that hasMileage is a property for Car, which corresponds to the relationship set between Car and Mileage in Figure 3. Further observe that we capture the participation constraint 0:1 for this relationship set by the minCardinality and maxCardinality declarations in Figure 7. Declaring Engine to be a subClassOf Feature in Figure 7 is how we capture the Engine-ISA-Feature part
of the generalization/specialization in Figure 3. For the aggregation in Figure 3, we simply decompose it into ordinary binary relationship sets and model it as we do other binary relationship sets, adding transitivity as a distinguishing feature when there are two or more levels in the aggregation hierarchy. As indicated by these examples, the algorithm to convert an extraction ontology in our system to an OWL ontology is reasonably straightforward.

4.2 Extensional Representation

Generally speaking, we can see two ways to represent annotated data instances: *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 ex-
tracts instance position information as well as the data instances and stores them in an externally specified knowledge base. As examples, [33] shows how to do explicit annotation (most others have followed this lead), and [36] shows how to do implicit annotation. In our prototype, we have implemented both explicit and implicit annotation.

Using explicit annotation, we have created an online demonstration [15] of our semantic annotation tool. Figure 8 is a screen shot showing that our system has extracted specific information from a web site containing car ads and has, in addition, annotated the web page so that we can highlight extracted information with the hover feature of CSS. The hover feature is only for the demonstration. For the annotation itself, we include a four-tuple in each 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, as happens in our ontology for car ads, for example, with Feature, which can have multiple values in a single record. Thus, for example, we annotate the value 117K in Figure 8 by \(<\text{span class="CarAds,Mileage,13,0"}>117K</span>\). Here CarAds is the ontology, Mileage is the concept, 13 is the record number, and 0 is the value number. Span annotations along with a URL specifying an OWL ontology allow the system to create the equivalent of a populated semantic ontology for each annotated page.

For implicit annotation, we start by generating an OWL ontology from an extraction ontology. Then we augment the OWL ontology with instance data. Figure 9 shows a portion of an implicit annotation for the Athens web page in Figure 1. When we do implicit annotation, we also cache a copy of the web page so that we can guarantee that the instance position information is correct. Figure 9 shows a URL indicating that for our implementation we have cached the web page on our web site. Following the URL in Figure 9, we show the beginning of our CarAds ontology. Next, we add the instances. Figure 9 shows the mileage instance 117K, its canonical value 117000, and its character offset 37733 in the cached web page. Observe that we give each instance a unique
identifier, \textit{MileageIns13} for the \textit{117K} in our example. We then collect all the instances together as a record. The OWL \textit{Thing} in Figure 9 is the record about \textit{CarIns13}, which includes the unique identifiers of its price, year, make, etc. This semantic meaning disclosure file fully annotates the Athens web page in Figure 1.

Both annotation techniques have their own benefits and drawbacks. Explicit annotation favors situations in which web pages frequently change but only with minor modifications (e.g., 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, 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 (1) web pages change infrequently or change in
ways that would destroy explicit annotation, or (2) 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 be outdated. 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: \(<tag1>data1<tag2>common data</tag1>data2</tag2>\).

With respect to query processing, which we discuss in the next section, we point out that it is
possible to query both explicit and implicit annotations. It is more convenient, however, 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. For explicit annotations, we would first need to transform the annotations into XML format before issuing the XQuery statement.

5 Querying Annotated Semantic Web Pages

Given an implicitly annotated semantic web disclosure file (i.e., externally stored annotation information), we can query the file and thus query the annotated web page. Since the semantic web disclosure file is in XML, we can use XQuery to directly query the information, as we explain in Section 5.1. Ordinary users, however, will not be able to write queries in XQuery. We therefore argue in Section 5.2 that a more user-friendly mechanism is needed and further suggest that information-extraction ontologies may provide us with a reasonable way to provide the needed user-friendly mechanism.

5.1 XQuery for Implicitly Annotated Semantic Web Pages

Figure 10 shows an XQuery for our sample query, “Find me a red Nissan for under $5000 – it should be a 1990 or newer and have less than 120K miles on it,” written over the external semantic web annotation disclosure file in Figure 9. Each let clause looks up the corresponding extracted value, and a phrase within the where clause tests the given condition. An element in the return clause generates XML that contains the extracted value. In our running example, all the concepts happen to be optional; for required concepts we drop the “or empty(...)” phrase. To perform semantic web searches, we apply this query to all documents that are applicable to the given domain, collect the results, and display them to the user in tabular format as Figure 2 shows.

Note that optional elements might not be present in some of the records, and thus—as is the case with the ordinary web—our semantic web queries may return irrelevant results. For example, suppose a car ad does not list the car’s color, but otherwise satisfies the user’s constraints. Rather than miss a potential object of interest, we allow optional elements to be missing, and we return the partial record with the query results. It would not be hard to allow users to override this behavior and require the presence of all concepts in each of the query results.
for $doc in document(\$URL)/rdf:RDF return
<QueryResult> {
for $Record in $doc/owl:Thing
let $id := substring-after(xs:string($Record/@rdf:about), “CarIns”) 
let $Color := $doc/car:Color[@rdf:ID=concat(“ColorIns”, $id)]/car:ColorValue/text() 
let $Make := $doc/car:Make[@rdf:ID=concat(“MakeIns”, $id)]/car:MakeValue/text() 
let $Year := $doc/car:Year[@rdf:ID=concat(“YearIns”, $id)]/car:YearValue/text() 
let $Mileage := $doc/car:Mileage[@rdf:ID=concat(“MileageIns”, $id)]/car:MileageValue/text() 
where ($Color = “red” or empty($Color)) and ($Make = “Nissan” or empty($Make)) and ($Price < 5000 or empty($Price)) and ($Year >= 1990 or empty($Year)) and ($Mileage < 120000 or empty($Mileage))
return <Record ID="{$id}" > <Color>{$Color}</Color><Make>{$Make}</Make> 
        <Price>{$Price}</Price><Year>{$Year}</Year><Mileage>{$Mileage}</Mileage> </Record> 
} </QueryResult>

Figure 10: XQuery to Search an Annotated Web Page.

5.2 IE-Based Semantic Web Queries

For researchers and developers, XQuery is a fine choice as a query language for the semantic web. On the other hand, few end users will have the ability, patience, or interest to write XQuery expressions. A practical semantic web query solution must be sufficiently expressive while also being easy to use. Indeed, successful search engines respond to mostly unstructured, free-form text, with a small set of operators such as quoted phrases and plus signs in front of required words. Even these simple operators are ignored by most end users. Google also provides an “Advanced Search” page that delivers a query form to help users access the various search semantics, including phrase search, require all, require one, and negation, among others, but relatively few end users take advantage of it.

We see several general strategies for how to address the needs of end users: graphical query languages, forms-based queries (effectively syntax-directed or by-example queries), and free-form textual queries. This is an important area for future research, so we briefly explore the possibilities. We have experimented previously with graphical query languages [12, 13] and with forms-based approaches [21]. However, we have good reason to believe that, like current web search engines, semantic web searches will migrate to free-form text.

The authors of [8] discuss principles and desirable features for web query languages. With their
“verbalizing” principle, they advocate “some form of controlled natural language processing” and point out that the success of web search engines demonstrates the importance of “a seemingly free-form, ‘natural’ interface.” We agree, and we further advocate an even stronger position. For many ordinary users of the semantic web, we believe that queries should be totally free-form, natural-language text, with no restrictions. The problem then becomes how to interpret the query. It is easy to suppose that natural language processing (NLP) can solve this problem, but this is easier said than done. Furthermore, free-form natural text may be telegraphic (may be short, sometimes non-grammatical, and often having incomplete phrases), on which NLP techniques typically do not work well. The authors of [6] suggest the use of controlled natural language. Perhaps this may be necessary and potentially could be successful. Users, however, must learn and adjust to artificial requirements imposed over the syntax and semantics of the language. Many ordinary users may neither have the patience to learn the nuances of the language nor the skill necessary to make the required adjustments.

In light of these requirements (the need for free-form, natural-language-like text) and difficulties (the problems of traditional NLP and controlled natural languages), we propose a different approach. Our approach may be characterized as an information-extraction, ontology-based, natural-language-like approach. The essence of the idea is to (1) extract constants, keywords, and keyword phrases in a natural-language or telegraphic query; (2) find the best-match ontology; and (3) embed the query in the ontology yielding (3a) a join over the relationship-set paths connecting identified concepts, (3b) a selection on identified constants modified by identified operators, and (3c) a projection on mentioned concepts. Our expectation for success is based on arguments in [9] suggesting the possibility of using ontologies to help build better question/answer systems and on our experience long ago experimenting with a natural-language-like approach to query phrases over conceptual models [20].

Consider our running example, where the user specifies, “Find me a red Nissan for under $5000 – it should be a 1990 or newer and have less than 120K miles on it.” The best-matching extraction ontology from our library is the car-ads ontology. When we apply our car-ads extraction ontology
to this sentence, we discover that the desired object has restrictions on five concepts: color, make, price, year, and mileage. For string-valued concepts (color and make), we can test equality (either equal or not equal). Since there are no keyword phrases in the query that indicate negation, in this case we search for objects where Color = red and Make = Nissan. For numeric concepts (price, year, and mileage), we can test ranges. Associated with each operator in a data frame are keywords or phrases that indicate when the operator applies. In this case, “under” indicates < (a less-than comparison), “or newer” indicates ≥, and “less than” indicates <. So in our example, we must search for Price < 5000, Year ≥ 1990, and Mileage < 120000. Recall, from our discussion in Section 2, that our data frames specify operators that convert a string to a canonical internal representation. Thus, for example, “120K” becomes 120000 when we invoke the canonicalization operator. The result of this extraction over the user-specified query is a set of filters that indicate concept name, operator, value, and optionality (whether a value may or must appear according to the participation constraints in the extraction ontology). In the search process, we look for objects that satisfy all the filter expressions. Figure 11 shows the particular concept filters for our example. Given a set of concept filters, we can readily generate rather than write the XQuery in Figure 10.

This is an important area for future research, and we are not yet prepared to claim that this proposed query scheme is definitely the best path forward. But we have illustrated how the use of extensional semantics not only helps recognize data instances, but it can also help convert natural-language queries into machine-processable queries. When both annotations and machine queries are generated through the same interface, we expect to see improved practical performance in answering these queries.

<table>
<thead>
<tr>
<th>Name</th>
<th>Operator</th>
<th>Value</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>=</td>
<td>red</td>
<td>true</td>
</tr>
<tr>
<td>Make</td>
<td>=</td>
<td>Nissan</td>
<td>true</td>
</tr>
<tr>
<td>Price</td>
<td>&lt;</td>
<td>5000</td>
<td>true</td>
</tr>
<tr>
<td>Year</td>
<td>≥</td>
<td>1990</td>
<td>true</td>
</tr>
<tr>
<td>Mileage</td>
<td>&lt;</td>
<td>120000</td>
<td>true</td>
</tr>
</tbody>
</table>

Figure 11: Filters Extracted from Natural-Language User Query.
6 Evaluation

We provide two types of evaluation—an objective evaluation of annotation accuracy and a subjective evaluation giving our view of what it would take to make our prototype system viable and practical.

6.1 Accuracy Evaluation

We are interested, of course, in how accurately an annotation system binds real-world data to the concepts defined in annotation ontologies. Since our annotation results depend, and only depend, on our ability to correctly extract information, we can apply the traditional IE evaluation metrics, precision and recall, to evaluate performance accuracy. We point out, however, that this is not the case for a traditional non-ontology-based IE process. For non-ontology-based IE annotators, calculations of precision and recall are according to either self-defined or machine-learned extraction categories. But for semantic annotation, we need to compute precision and recall with respect to the concepts defined in a domain ontology. Therefore, for systems that use a non-ontology-based IE engine, there are two precision and recall metrics. One evaluates the performance of IE processes itself, and the other evaluates how well the system maps these extraction categories to the concepts defined in an ontology. The final precision and recall values are the products of the two respective precision and recall values. This is not required for annotation systems that use ontology-based IE engines, such as ours. Because of the integration of ontologies into the extraction process itself, the evaluation of precision and recall for the semantic annotation system is the same as the evaluation of precision and recall for the original ontology-based IE tool.

Although the study of semantic annotation is still a new research topic, researchers have studied information extraction for more than a decade, and so have we. Over the course of many years, we have developed our ontology-based IE tool and have tested it on various domains, each with dozens of real-world web pages. Among them, there are some simple, unified domains like automobile sales and apartment rentals, and there are complicated or loosely unified domains like genealogy and obituaries.
Based on approximately 20 domains with which we have experimented we summarize our experience as follows. In simple, unified domains we typically achieve close to 100% precision and recall in almost all fields, while in more complicated or loosely unified domains, the precision and recall for some fields falls off dramatically. For obituaries, for example, we were only able to achieve about 74% precision for relatives of the deceased and only about 82% recall for recognizing funeral addresses. In general, within nearly 20 domains that contain in total over 200 different object sets, our extraction engine typically achieves at least 80% accuracy for both precision and recall values on most fields. For over half of the domains, the precision and recall values were above 90%. Further details about these experiments and results can be found in our previous publications (e.g. [22, 24, 27]). Many of these domain ontologies and some cached real-world web pages are also available in our online demos at our web site [15], which contains both our original ontology-based IE tool demo and our current semantic annotation prototype demo.

6.2 Practical Considerations

Beyond accuracy, there are several criteria that a practical semantic annotation system should satisfy, such as generality, resiliency, conformance to standards, and ease of development and maintenance of complex ontologies. In contrast with precision and recall measures, it is harder to establish objective metrics for these practical considerations. We cannot, however, ignore these important criteria, since the success of a semantic annotation system depends on them.

Our first practical consideration is generality of the semantic annotation approach. In other words, what is the range of pages for which the annotation system is effective? Because we use an ontology-based IE engine, our prototype system targets data-rich web pages that each have a relatively narrow domain [22]. There is no particular restriction that limits applicability, but as the domain of a page broadens, our approach becomes less accurate because the extensional-level semantics overlap more and become harder to segment. This issue is not unique with our approach (see, for example, [42]). Fortunately, data-rich pages are quite common on the web (consider shopping, news, and product portals, for example).

Within an application domain, our semantic annotation approach works best on semi-struc-
tured web pages containing multiple records that are laid out in straightforward fashion. A multi-record collection lets our system cross-validate the correctness of recognized data instances. We have also shown that our approach works well on single-record web pages and complex web pages with complicated table structures [24, 27]. Although our method is also applicable to fully unstructured natural-language text, our experiments show that performance is usually lower in these scenarios. Unlike other semantic annotators (such as [31] and [36]), there are no particular NLP methods encoded in our ontology-based data-recognition program. A question we expect to explore in the future is whether a hybrid system that also uses NLP techniques will increase the generality of our approach.

Another practical consideration is the resiliency of an annotation system. Web pages change often, both in terms of current content and physical layout. If such changes break the underlying automatic annotator, someone will have to work to maintain the annotation system, and such an approach will ultimately fail to scale to cover the enormous web. As we explained earlier, our approach is resilient to web page layout changes, and thus we minimize the need for wrapper maintenance in the information-extraction layer of the system [38]. A trade-off for resiliency is that our current system sacrifices some execution speed (and possibly even some accuracy). To address this problem, we have proposed—and are working to develop—a two-layer semantic annotation architecture that will divide the work more efficiently into an upper-layer set of structural annotators and base-layer conceptual annotators [18]. Each layer will be optimized to its particular task.

Another practical consideration is adherence to accepted standards. The reason we annotate pages in the semantic web is so we can use them. Any system that does not conform to semantic web standards will not be interoperable, and thus will not get used. Thus, we convert our proprietary OSMX ontologies to standard OWL ontologies when we generate annotations. Most recent semantic annotation approaches adopt a similar strategy. Researchers using implicit annotation (where annotations are stored separately from source pages) typically use either RDF [36] or DAML+OIL [31].
Although almost all of the current main-stream automatic semantic annotation approaches assume the separation of ontology generation and semantic annotation, the problem of ontology generation is an unavoidable issue for a workable semantic annotation system. In the absence of high-quality ontologies it does not matter how well the data recognition process works. Without good ontologies we cannot get high-quality annotations. Since ontology creation is an inherently difficult task requiring expert developers, we have developed several different ontology construction tools to support building high-quality information extraction ontologies. These tools include a manual ontology construction tool [49], a semi-automatic ontology construction tool that can extract a smaller domain ontology out of a large, unified knowledge base [40], and a by-example ontology generation tool that can semi-automatically create an information extraction ontology using manually marked web pages [52].

7 Concluding Remarks

We have presented an approach to semantic web-page annotation that is based on the use of data-extraction ontologies. We have argued that ontology-based information extraction engines can provide a solid foundation for an automated semantic web annotation tool. Ontology-based IE engines provide two fundamental advantages: (1) they include declared extensional-level semantics, and (2) they extract information directly into an annotation ontology. In our experiments, both precision and recall are running at roughly 85% to 90% for each of the individual lexical concepts in an extraction ontology. Our prototype implementation supports both internal and external annotation. We can directly query our external annotation with XQuery.

Although we have accomplished much, there is much more to do to make the ontology-based IE approach to semantic web annotation practical. We have argued that unconstrained natural-language query processing may be necessary, and we have proposed as future work an ontology-based IE approach to unconstrained natural-language query processing. We have further suggested that additional work on incorporating NLP-based annotation techniques into our system and implementing an efficient two-layer semantic annotation model may be necessary.
The future of the semantic web is bright, but delivering on its vision will not be easy. Effective deployment of the semantic web requires some way to automatically accommodate the huge quantity of existing data-rich web pages on the ordinary web, and some way to handle ordinary user requests. Our approach addresses these challenges.

Acknowledgments

This work was supported by NSF under grants 0083127 and 0414644 and by the Rollins Center for eBusiness at BYU under grant EB-05046.

References


555-570, Yokohama, Japan, November 2001.

[26] D.W. Embley and W.Y. Mok, “Developing XML Documents with Guaranteed Good Propo-
erties,” Proc. 20th International Conference on Conceptual Modeling (ER2001), pp. 426-441,

[27] D.W. Embley, C. Tao, and S.W. Liddle, “Automating the extraction of data from HTML
tables with unknown structure,” Data & Knowledge Engineering, vol. 54, no. 1, pp. 3–28,
July 2005.


IJCAI-99 Workshop on Intelligent Information Integration, Stockholm, Sweden, July 31,
1999.

data,” Proc. European Conference on Knowledge Acquisition and Management (EKAW-


ficial Intelligence for Web Search, pp. 35–40, Menlo Park, California, July-August 2000.


