Automatic Creation and Simplified Querying of Semantic Web Content: An Approach Based on Information-Extraction Ontologies

Yihong Ding[1⋆], David W. Embley[1⋆], and Stephen W. Liddle[2⋆⋆]

[1] Department of Computer Science,
[2] Department of Information Systems,
Brigham Young Univeristy, Provo, Utah 84602, U.S.A.
{ding, 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. Furthermore, users need an effective way to query the semantic web, but any burden placed on users to learn a query language is unlikely to garner sufficient user support and interest. Unfortunately, both the creation and use of semantic-web pages are difficult, and these are precisely the processes that must be made simple in order for the semantic web to truly succeed. We propose using information-extraction ontologies to handle both of these challenges. In this paper we show how a successful ontology-based data-extraction technique can (1) automatically generate semantic annotations for ordinary web pages, and (2) support free-form, textual queries that will be relatively simple for end users to write.

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 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 traditional search engines cannot answer this type of request because they do not know how to match the specified concepts in the request to data instances on the web.

⋆ Supported by NSF grant #0414644.
⋆⋆ Supported by the Kevin and Debra Rollins Center for eBusiness at Brigham Young University under grant EB-05046.

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* [2]. Semantic-web proponents propose making web content machine understandable through the use of *ontologies*, which are commonly shared, explicitly defined, generic conceptualizations [7]. 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 semantic web query language (SPARQL [15] 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 *Car1* (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.

Our automated semantic annotation approach employs a unique ontology-based data recognizer that uses information-extraction (IE) ontologies. A unique characteristic of this approach is the use of instance recognition semantics inside ontologies to help specify annotation domains and perform data recognition. Our approach solves a common annotation problem of requiring "a set of heuristics for post-processing and mapping of the IE results to an ontology" [9].

Fig. 1. Sample Car Ads from Salt Lake City Weekly and Athens Banner-Herald Sites.

We give the details[3] of our contribution of automatically creating semantic web content so that we can directly query it as follows. Section 2 describes information-extraction ontologies, which are the basis for our automated semantic-web annotation tool. Section 3 describes our prototype work on automatically annotating existing web pages so that they can be used for the semantic web, and Section 4 shows how we can directly query pages annotated for the semantic web. Section 5 provides experimental evidence about the accuracy of our annotation system as well as pragmatic consideration. We conclude in Section 6.

[3] Since this paper gives a full, broad vision of our approach to both the creation and use of semantic-web pages, our presentation is necessarily high level. We provide as much detail as space allows and refer the interested reader to additional papers that augment ideas and results presented here.

Color	Make	Price	Year	Mileage	Source
	Nissan	$4,500	1993	117,000	Car1
...	...	...	...	...	...
red	Nissan	$900	1993		Car13
...	...	...	...	...	...

Fig. 2. Query Results.

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.

2.1 Information Extraction Ontologies

We have described information-extraction ontologies elsewhere [6], but to make our paper self-contained, we briefly reintroduce them here.[4] 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, "Car1", 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

[4] We mention, in passing, that the ontological basis for our extraction ontologies has been fully formalized in terms of predicate calculus. (See Appendix A of [5].)

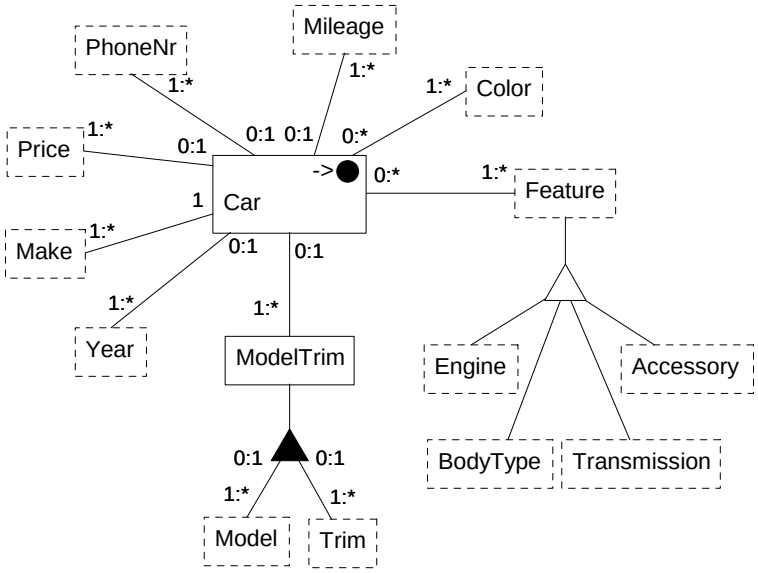


Fig. 3. Graphical Component of an Extraction Ontology

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. 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

```
Price
    internal representation: Integer
    external representation: \$?(\d+ | \d?\d?\d,\d\d\d)
    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
```

Fig. 4. Sample data frames for car ads ontology.

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 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, sweet cherry red. 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*, "red" as the *Color*, both "Air Conditioning" and "new tires" as *Feature*s 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 canonical internal representations, so that, for example, "**'93**" becomes the integer *1993* and "$900" becomes the integer *900*.

2.2 Annotation through Instance Recognition Semantics

Information-extraction ontologies are well positioned to satisfy the requirements of semantic annotation. Not only do they provide the intentional-level semantics found in typical ontologies, but they also provide the instance recognition semantics needed to connect individual data items found in ordinary web pages with the typical intentional-level semantics.

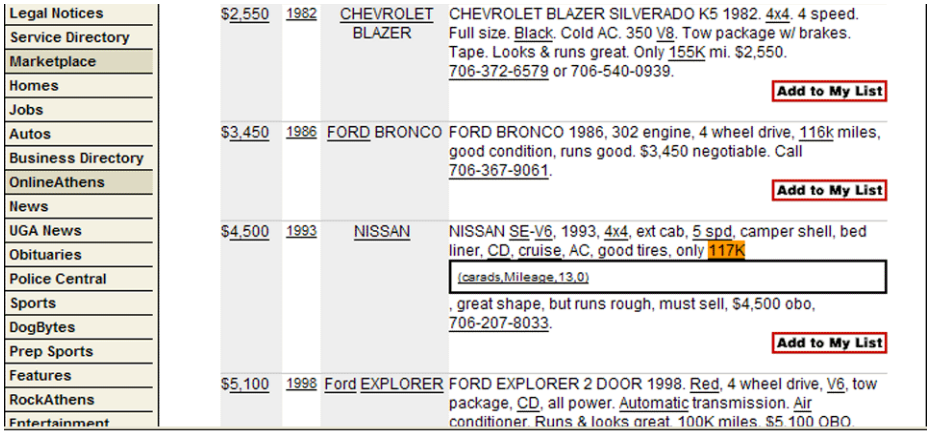
Figure 4 exemplifies the fundamental idea. The external representations describe textual instantiation patterns of a concept. Added to these instantiation patterns, we provide regular expressions for context and keyword phrases, which aid in correctly classifying instantiation patterns that may be similar in several different data frames.

This approach stands in stark contrast to a typical automated semantic annotation paradigm (e.g., the approaches in [1], [4], [8], [9], [12], and [16]), which do not use extraction ontologies. Although results are encouraging for these automated semantic annotators, there are problems in these annotation paradigms. A complete annotation process using typical non-ontology-based IE tools contains 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 extraction categories in an IE wrapper with concepts defined in semantic-web ontologies. Sheth and Ramakrishnan believe this "concept disambiguation" problem is a major issue for the semantic annotation [14]. Furthermore, Kiryakov et al. think that this requirement of post-extraction alignment is the "main drawback" of current automated annotation approaches [9]. They suggest that we need to integrate domain ontologies with extraction engines to solve the problem and proposes this as a direction for future work [9]. Indeed, this is the approach we take. Since information-extraction ontologies represent extraction categories with ontologies, we can combine the problems of data recognition and concept disambiguation and simplify the structure of the semantic annotation problem.

3 IE-Based Semantic Web Annotation

Generally speaking, there are 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 extracts instance position information as well as the data instances and stores them in an externally specified knowledge base. In our prototype, we have implemented both explicit and implicit annotation.

Using explicit annotation, we have created an online demonstration [3] of our semantic annotation tool. Figure 5 is a screen shot showing that our system has extracted specific information from a web site containing car ads and



Make	Model	Trim	Year	Mileage	PhoneNr	Price	Color	Transmisson	Engine	BodyType	Accessory	OtherFeature	Source
Dodge			1984		706-769-4466	2,000				Show			5
TOYOTA			2002	100k	706-769-4323	19,800			Show	Show			7
CHEVROLET			1982	155K	706-372-6579	2,550	Show		Show	Show			9
FORD			1986	116k	706-367-9061	3,450							11
NISSAN		SE	1993	117K	706-207-8033	4,500		Show	Show	Show	Show		13
Ford	EXPLORER		1998	100K	706-769-3060	5,100	Show	Show	Show	Show	Show		15
CHEVROLET			1971	18K	706-357-5145	16,000	Show						17
FORD	F150		1999	103k	706-318-7730	7,250	Show	Show		Show	Show		19

Fig. 5. Page Annotation Demo: Car Ads from Athens Site (Hovering on 117K).

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 5 by 117K. 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 [13] 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 create an RDF data file to store annotated data instances based on the domain declaration defined in the OWL ontology. Figure 6 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. In

```
<rdf:RDF ... xmlns:ontos="http://www.deg.byu.edu/ontology/ontosBasic#"
             xmlns:carad="http://www.deg.byu.edu/ontology/carad#"
             xmlns:webpage="http://www.deg.byu.edu/demos/..." ... >
...
  <rdf:Description rdf:about="&webpage;CarIns13">
    <carad:Mileage>117000</carad:Mileage>
    <carad:Price>4500</carad:Price>
    <carad:Make>Nissan</carad:Make>
    <carad:Year>1993</carad:Year>
    ...
  </rdf:Description>
  <rdf:Description rdf:about="&webpage;Mileage13">
    <ontos:ValueInText>117K</ontos:ValueInText>
    <ontos:CanonicalValue>117000</ontos:CanonicalValue>
    <ontos:CanonicalDataType>xsd:integer</ontos:CanonicalDataype>
    <ontos:CanonicalDisplayValue>117,000</ontos:CanonicalDisplayValue>
    <ontos:Offset> 37733 </ontos:Offset>
  </rdf:Description>
...
</rdf:RDF>
```

Fig. 6. Implicit Annotation for Car Ads Web Page.

Figure 6, we first declare several namespaces of referenced ontologies and web pages. Specifically, we include an *ontos* namespace, which provides general system information, a namespace referencing the ontology we use for annotation (here *carad*), and a *webpage* namespace for the annotated web pages. For each car, we store its canonical data values with their respective attribute names. For a lexical concept, such as mileage in Figure 6, we store its original value in the source text (*117K*), its canonical internal value (*117000*) and type (*integer*), and its canonical display value (*117,000*). We use canonical internal values (together with type information) in SPARQL queries and use canonical display values when returning results to users (as in Figure 2). We also store offset values in the cached web page (e.g. *37733* is the actual offset of the extracted instance "117K"). The RDF document in Figure 6 fully annotates the Athens web page in Figure 1.

4 Querying Annotated Semantic Web Pages

Given an implicit annotation in an RDF file, we can query the file and thus query the annotated web page. Because we store information in an RDF file, we can use SPARQL to query the information directly, as we explain in Section 4.1. Ordinary users, however, will not be able to write queries in SPARQL. We therefore argue in Section 4.2 that a more user-friendly mechanism is needed and further show that information-extraction ontologies may give us a reasonable way to provide the needed user-friendly mechanism.

```
PREFIX carad: <http://www.deg.byu.edu/ontology/carad#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
SELECT ?make ?color ?price ?year ?mileage
WHERE { ?x carad:Make ?make . FILTER (?make = "Nissan") .
    OPTIONAL {?x carad:Color ?color} . FILTER (?color = "red") .
    OPTIONAL {?x carad:Price ?price} . FILTER (xsd:integer(?price) < 5000) .
    OPTIONAL {?x carad:Year ?year} . FILTER (xsd:integer(?year) >= 1990) .
    OPTIONAL {?x carad:Mileage ?mileage} .
                                    FILTER (xsd:integer(?mileage) < 120000) }
```

Fig. 7. SPARQL Query to Search an Annotated Web Page.

4.1 SPARQL for Implicitly Annotated Semantic Web Pages

Figure 7 shows a SPARQL rendition of 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." The query is written over the RDF file in Figure 6 that annotates the web page. The *PREFIX* clause associates a prefix label with an IRI (a generalization of URIs that is fully compatible with URIs and URLs). The prefix label becomes a local namespace abbreviation for the address specified by the IRI. The *SELECT* clause names the result variables. The first clause in the *WHERE* clause requires the car bound to x to have *Make* equal to *NISSAN*. Each *OPTIONAL* clause checks whether a corresponding extracted value satisfies certain constraints. The keyword *OPTIONAL* allows the content to be unbound. Otherwise, however, any bound value must satisfy the constraints in the corresponding *FILTER* clause. 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 a tabular format as Figure 2 shows.

The reason we make our conditions *OPTIONAL* is that optional elements might not be present in some of the records. 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.

4.2 IE-Based Semantic Web Queries

For researchers and developers, SPARQL 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 learn to write SPARQL queries. A practical semantic web query solution must be sufficiently expressive while also being easy to use. We believe that, like current web search engines, semantic web searches will migrate to free-form text. Because it is impossible to execute free-form queries directly, mapping from free-form queries to executable queries is necessary.

Our approach can 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 free-form query; (2) find the ontology that matches best; 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.[5,6]

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 extraction ontology from our library that best matches this query 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, 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 $\geq$, and "less than" indicates $<$. So in our example, we must search for *Price* $<$ *5000*, *Year* $\geq$ *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 and to a canonical representation for display. Thus, for example, *"120K"* becomes the integer *120000* as its canonical internal representation and the string "120,000" as its canonical display value. We therefore are able to apply standard conditions and *FILTER* clauses to compose a SPARQL query. Because web data is stored using an open world assumption, we should not reject an answer when a data value is not present. Hence, by default we add *OPTIONAL* before each generated condition. There is, however, another factor that decides the *OPTIONAL* before a generated condition. When a minimum participation constraint in the extraction ontology is "1", the corresponding generated condition becomes mandatory instead of optional. For example, in Figure 3, each *Car* must have one and only one *Make*. We therefore remove the default *OPTIONAL* from the generated condition of *Make*. Figure 8 shows the particular concept conditions for our example. Given a set of concept conditions, we can readily generate, rather than manually write, the SPARQL query in Figure 7.

[5] See [17] for a full explanation.

[6] The theoretical underpinnings of this approach are found in the "window functions" explained in [11].

Name	Operator	Value	Optional
Color	=	*red*	*true*
Make	=	*Nissan*	*false*
Price	<	*5000*	*true*
Year	≥	*1990*	*true*
Mileage	<	*120000*	*true*

Fig. 8. Filters Extracted from Natural-Language User Query.

5 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.

5.1 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 information extraction (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 the IE process 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%.

We have recently been able to obtain some initial results for IE-based query conversion [17]. Four subjects each provided five queries on five domains (car ads, real estate, countries, movies, and diamonds) for a total of 100 queries. The recall for identifying concept values to be returned was 89% and for correctly generating conditions was 75% while the corresponding precision values were respectively 89% and 88%. Overall, the system interpreted 47% of the queries with perfect accuracy while interpreting an additional 49% with partial accuracy.

5.2 Practical Considerations

Beyond accuracy, there are several criteria that a practical semantic annotation system should satisfy, such as generality, resiliency, and conformance to standards. 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 [6]. There is no particular restriction that limits applicability, but as the domain of a page broadens, our approach becomes less accurate because the instance recognition semantics overlap more and become harder to segment. This issue is not unique with our approach (see, for example, [12]). Fortunately, narrow-domain, 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-structured web pages containing multiple records that are laid out in a straightforward way. A multi-record collection lets our system cross-validate the correctness of recognized data instances. The approach, however, also works on single-record web pages and complex web pages with complicated table structures. Although our method is also applicable to fully unstructured natural-language text, our experiments show that performance is usually lower for these types of pages. Unlike other semantic annotators (such as [8] and [9]), there are no typical natural-language-processing (NLP) techniques 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. 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 [10]. 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. Each layer will be optimized for 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 be used. Thus, we convert our proprietary OSMX ontologies to standard OWL ontologies [13] 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 [9] or DAML+OIL [8].

6 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 instance recognition 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 SPARQL. We can also generate SPARQL queries from free-form text input, and we therefore provide a way for ordinary users to query annotated semantic web pages. In initial experiments with the query generator, 47% of the queries submitted by subjects returned fully correct results, and all but 4% returned some useful results. We are currently working to improve the quality of these generated queries.

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.

References

1. L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo, "Automatic annotation of data extracted from large web sites," *Proc. Sixth International Workshop on the Web and Databases (WebDB 2003)*, pp. 7-12, San Diego, California, June 2003.
2. T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 36, no. 25, pp. 34-43, May 2001.
3. Homepage, *BYU Data Extraction Group*, URL: http://www.deg.byu.edu.
4. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K.S. McCurley, S. Rajagopalan, A. Tomkins, J.A. Tomlin, and J.Y. Zien, "A Case for Automated Large Scale Semantic Annotations," *Journal of Web Semantics*, vol. 1, no. 1, pp. 115–132, December 2003.
5. D.W. Embley and B.D. Kurtz and S.N. Woodfield, *Object-oriented Systems Analysis: A Model-Driven Approach*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.
6. D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data & Knowledge Engineering*, vol. 31, no. 3, pp. 227-251, November 1999.
7. T.R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199-220, 1993.
8. S. Handschuh, S. Staab, and F. Ciravegna, "S-CREAM Semi-automatic CREAtion of Metadata," *Proc. European Conference on Knowledge Acquisition and Management (EKAW-2002)*, pp. 358–372, Madrid, Spain, October, 2002.
9. A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff, "Semantic Annotation, Indexing, and Retrieval," *Journal of Web Semantics*, vol. 2, no. 1, pp. 49–79, December 2004.
10. K. Lerman, S. N. Minton, and C. A. Knoblock, "Wrapper maintenance: A machine learning approach," *Journal of Artificial Intelligence Research*, vol. 18, pp.149–181, 2003.
11. D. Maier, *The Theory of Relational Databases*, Computer Science Press, Inc., Rockville, Maryland, 1983.
12. S. Mukherjee, G. Yang, and I.V. Ramakrishnan, "Automatic Annotation of Content-Rich HTML Documents: Structural and Semantic Analysis," *Proc. Second International Semantic Web Conference (ISWC 2003)*, pp. 533–549, Sanibel Island, Florida, October, 2003.
13. W3C (World Wide Web Consortium) *OWL Web Ontology Language Reference*, http://www.w3.org/TR/owl-ref/.
14. A. Sheth and C. Ramakrishnan, "Semantic (Web) technology in action: Ontology driven information systems for search, integration and analysis," *IEEE Data Engineering Bulletin*, vol. 26, no. 4, pp. 40-48, December 2003.
15. W3C (World Wide Web Consortium), *SPARQL Query Language for RDF*, February 2006. URL: http://www.w3.org/TR/rdf-sparql-query/.
16. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, "MnM: Ontology Driven Tool for Semantic Markup," *Proc. Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, pp. 43–47, Lyon, France, July, 2002.
17. M. Vickers, "Ontology-Based Free-Form Query Processing for the Semantic Web," Masters Thesis, Brigham Young University, Provo, Utah, June 2006.