Automatic Creation and Simplified Querying of Semantic Web Content*

David W. Embley, Yihong Ding, Stephen W. Liddle, and Mark Vickers

Brigham Young University, Provo, Utah 84602, U.S.A.
embley@cs.byu.edu, ding@cs.byu.edu, liddle@byu.edu, msv6@email.byu.edu

Abstract. The semantic web represents a major advance in web utility, but it is 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 we place on users to learn a query language is unlikely to garner sufficient user support and interest. If we want users to take advantage of the semantic web, we must devise a means for transforming existing (non-semantic) web pages into semantic web pages, and we must provide a simple and unrestricted interface for processing user queries. We propose using information extraction ontologies to handle both of these challenges. 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. Our approach demonstrates how the semantic web can be created for and used by ordinary people. We have created an initial prototype to demonstrate that our proposal works.

Keywords: Semantic Web Annotation, Natural Language Semantic Web Queries, Information-Extraction Ontologies.

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. *Google*, for example, does a phenomenal 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 in 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, i.e. develop

* Supported in part by the National Science Foundation under grant No. IIS-0083127 and the Rollins Center for eBusiness at BYU under grant No. EB-05046.

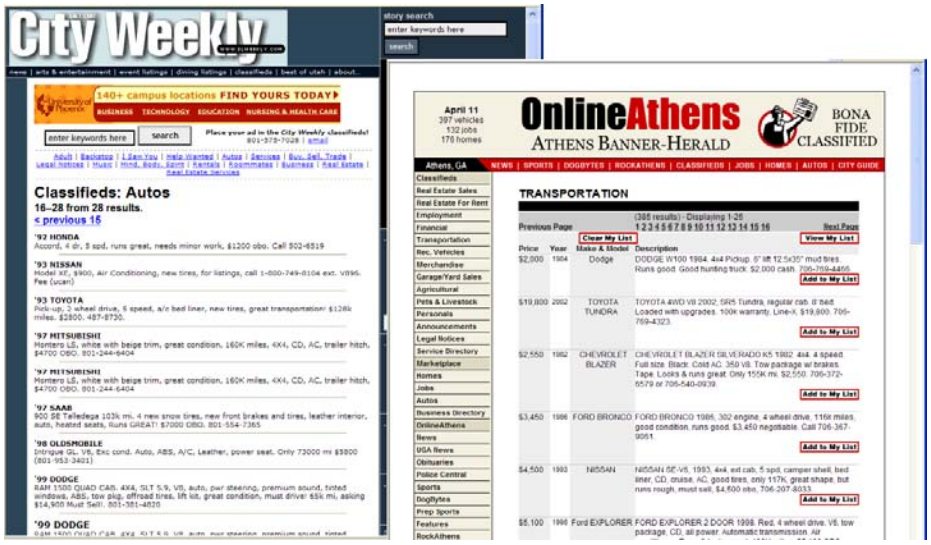

Fig. 1. Sample Car Ads from Salt Lake City Weekly and Athens Banner-Herald Sites.

the *semantic web* [3]. We make semantic-web content machine understandable through *ontologies*, which are commonly shared, explicitly defined, generic conceptualizations [19]. But then the dilemma is how to deal with current web pages. There are billions of pages in the current web, and few web developers would be willing 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 dilemma. The goal of web semantic annotation research 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 web pages with respect to an ontology.

Even if we annotate pages with respect to an ontology, however, we have still not completely solved our problem because we must provide a way for users to query these semantic web pages. We argue in this paper that any learning burden we place on users, such as learning how to use a query system, is not likely to be successful, and thus we must provide users with the ability to query the semantic web using free-form text queries that have no requirements other than providing a statement of what is wanted. We then show how to enable this possibility for domains of information on the semantic web, which we assume to be ontology-based, annotated, human-readable web pages.

Color	Make	Price	Year	Mileage	Source
	NISSAN	$4,500	1993	117K	Car01
....					
	NISSAN	$900	'93		Car13
....					

Fig. 2. Query Results.

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. Furthermore, our system can answer ordinary, natural-language text queries over semantic web pages. Thus, assuming our annotation system had already created semantic web pages for Figure 1, the query, "Find me a red Nissan for under $5000 – it should be a 1990 or newer and have less than 120K miles on it," 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, 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 contributions of both automatically creating semantic web content and providing a simple way to query semantic web content as follows. Section 2 describes information-extraction (IE) ontologies, which are the basis for both our automated semantic-web annotation tool and our simple-to-use semantic-web query tool. Section 3 argues that IE ontologies may well provide the best alternative for automatically annotating much of the current web. It makes this argument by surveying related work and showing how the approach based on IE ontologies resolves 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. Section 5 describes our prototype work on semantic web queries and shows how we can use extraction ontologies to simplify query requirements for semantic-web users. Finally, in Section 6 we make concluding remarks and consider future work, including both limitations and extensions of the work proposed here.

2 Information-Extraction Ontologies

We have described information-extraction ontologies elsewhere [13], but to make our paper self-contained, we briefly reintroduce them here. We have used them

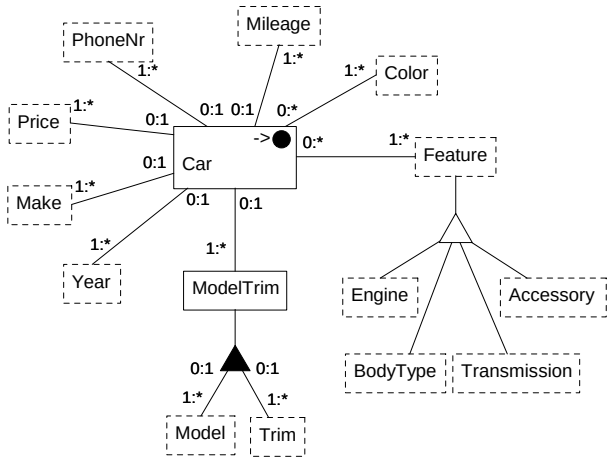

Fig. 3. Graphical Component of an Extraction Ontology.

in a number of applications, including information extraction itself [13], high-precision classification [16], and schema mapping for ontology alignment [34]. In this paper we show how to use them for semantic web annotation (Section 4) and semantic web queries (Section 5).

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

with the generalization connected to the apex of the triangle and one or more specializations 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 aggregation connected to the apex of the triangle and the component parts connected to its base. In Figure 3, for example, *ModelTrim* is an aggregation of the *Model* and the *Trim*. Like *ISA* hierarchies, large aggregation 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 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, 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, 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 *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 a canonical internal representation, so that, for example, "**'93**" becomes the integer *1993* and "$900" becomes the real number

```
Price
   internal representation: Real
   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.

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). In this case, we use heuristic rules to sort out the ambiguities [13]. We assume that we can correctly identify each individual record in a document. This requires work on record separation [14], 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 [18]. We assume also that the data is accessible, but often it is behind forms, in which case the system must first fill in the form and obtain the information from the hidden web [7]. Further, often the hidden web—and sometimes the visible web too—presents data in a relatively structured form, such as a table. Surprisingly, structure causes more difficulties for extraction ontologies than might be expected, and special techniques are needed to handle structured information [17].

3 Semantic Annotation

Having explained what an extraction ontology is, we now discuss how to apply extraction ontologies for annotating pages for the semantic web. A typical semantic annotation process includes three components. First, an ontology must be created to describe the 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.

For ontology creation, we assume either that an extraction ontology is hand crafted or semiautomatically generated. Using tools we have developed [32] for hand-crafting extraction ontologies of the size needed for car ads, we have found that a few dozen person hours is sufficient for building extraction ontologies that can extract with precision and recall both nearing 90% [13]. Others involved in semantic annotation research [11, 20, 25, 31] also assume ontologies already exist. Although [28] reports having tried to automatically generate ontologies for use in semantic annotation, they also report problems with this approach, particularly the "concept disambiguation problem" [29]. Generally, the problem of automatic ontology generation is considered to be difficult and not currently very successful [12], but research continues, including our own [30], and we certainly welcome a more automated way to generate ontologies.

For data instance recognition, manual recognition and annotation aided by tools is possible (e.g. [21, 24]). Although useful for small numbers of pages, it is unlikely that we can use manual semantic annotation tools to convert the current web to the semantic web. Automated semantic annotation tools are a necessity.

To develop automated annotation tools, researchers have turned to information-extraction (IE) tools to find answers. Based on the categories in a recent survey of IE tools [26], we can see how various IE tools do semantic annotation.

– *Wrapper languages* require manual specification, and thus will not scale.

– *HTML-aware IE tools* extract data of interest based on pre-defined HTML layout descriptions. [2] describes an attempt to do semantic annotation with the HTML-aware RoadRunner tool [8], but RoadRunner only finds the location of data of interest, not the semantic meaning of the data with respect to an ontology. This necessitates manual labeling or aligning the data extracted with an ontology, a task which the authors of [2] point out is not easy.

– *NLP-based IE tools* analyze text using NLP (Natural Language Processing) techniques. There are several current efforts to adapt these IE tools for semantic annotation [20, 25, 31]. Unfortunately, as currently constructed, these tools also suffers from the need to align extracted information with whatever domain ontology they use for semantic annotation. [25] explains, saying, "The main drawback ... is that none of these approaches expects an input or produces output with respect to ontologies. [Thus,] ... a set of heuristics for post-processing and mapping of the IE results to an ontology ... [is needed]." Further, NLP-based IE tools 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 web.

– *ILP-based tools* learn the formatting features that implicitly delineate the structure of data found in a page. [11] uses 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. Further, machine learning processes usually need a large set of training documents, which must be labeled, usually manually.

– *Modeling-based IE tools* adapt supervised machine learning approaches to do data extraction. Because this IE technique requires human-guided supervised learning, it is likely to be hard to scale this type of tool for large applications.

– *Ontology-based IE tools* (e.g. [1, 9, 13, 27]) apply pre-defined data-extraction ontologies to perform data extraction. Since this type of tool extracts information with respect to an ontology, no separate alignment of concepts is necessary. We argue that these types of tools are likely to be the best for developing automated semantic web annotation tools.

Previous to the work proposed here, no attempt has been made to use ontology-based IE tools for semantic annotation. Besides the benefit of eliminating the additional 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 and for new pages within the domain that come online.

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. Generally speaking, we can see two ways to achieve this goal: *explicit annotation*, which adds special tags that bind tagged instances in a web page to an externally specified ontology, and *implicit annotation*, which adds nothing explicit to the document, but instead extracts instance position information as well as the data instances and stores them in an externally specified ontology. As examples, [22] shows how to do explicit annotation (most others have followed this lead), and [23] shows how to do implicit annotation.

Using explicit annotation, we have created an online demo [10] 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 has, in addition, annotated the web page so that we can highlight extracted information with the hover feature of HTML. The hover feature is only for the demo. 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. 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 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.

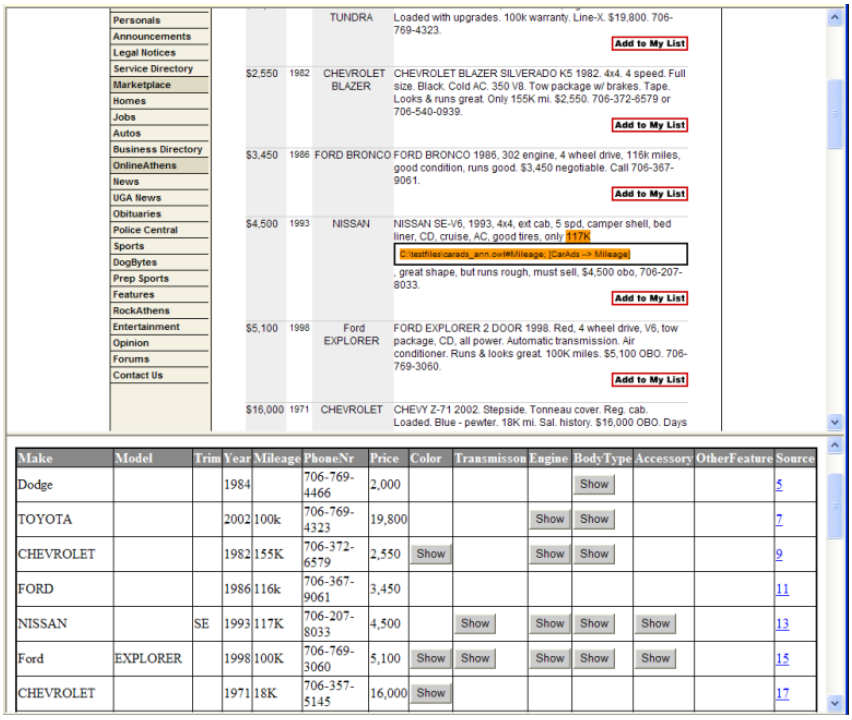

Fig. 5. Sample Annotated Page from Demo: Car Ads from Athens Site.

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. Figure 6 shows that for our implementation we have cached the web page on our web site. Following the URL in Figure 6, we show the beginning of our *CarAds* ontology. Next, we add the instances. Figure 6 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, *MileageIns13* for the *117K* in our example. We then collect all the instances together as a record. The OWL *Thing* in Figure 6 is the record about *CarAdsIns13*, 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

```
<?xml version="1.0"?>
<rdf:RDF
...
  xmlns:webpage="http://www.deg.byu.edu/demos/..."
...
  <owl:Class rdf:ID="CarAds">
...
  <carads:Mileage rdf:ID="MileageIns13">
    <carads:MileageValue rdf:datatype="xs:string">117K</carads:MileageValue>
    <carads:MileageCanonicalValue rdf:datatype="xs:nonNegativeInteger">117000
      </carads:MileageCanonicalValue>
    <offset rdf:datatype="xs:nonNegativeInteger">37733</offset>
  </carads:Mileage>
...
  <owl:Thing rdf:about="#CarAdsIns13">
    <hasPrice rdf:resource="#PriceIns13" />
    <hasYear rdf:resource="#YearIns13" />
    <hasMake rdf:resource="#MakeIns13" />
...
  </owl:Thing>
...
</rdf:RDF>
```

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

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 and (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: $<tag_1> data_1 <tag_2> common\ data </tag_1> data_2 </tag_2>$.

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.

5 IE-Based Semantic Web Queries

The authors of [5] discuss principles and desirable features for web query languages. In 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 pure NLP techniques have not proven to be very successful for question/answer systems (as witnessed by the flurry of early NLP work for database querying, which has since died out). Furthermore, free-form natural text may be telegraphic (may be short, sometimes non-grammatical, and often incomplete phrases), on which NLP techniques typically do not work well. [4] suggests 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 [6] suggesting the possibility of using ontologies to help build better question/answer systems and on our experience long ago with attempting to do "NLP" over conceptual models [15].

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 $\geq$, 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. 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. The particular concept filters for our example include: $\langle Color, =, red, true\rangle$, $\langle Make,$

```
for $doc in document(URL)/rdf:RDF return
<QueryResult> {
for $Record in $doc/owl:Thing
let $id := substring-after(xs:string($Record/@rdf:about), "CarAdsIns")
let $Color := $doc/carads:Color[@rdf:ID=concat("ColorIns", $id)]/carads:ColorValue/text()
let $Make := $doc/carads:Make[@rdf:ID=concat("MakeIns", $id)]/carads:MakeValue/text()
let $Price := $doc/carads:Price[@rdf:ID=concat("PriceIns", $id)]/carads:PriceValue/text()
let $Year := $doc/carads:Year[@rdf:ID=concat("YearIns", $id)]/carads:YearValue/text()
let $Mileage := $doc/carads:Mileage[@rdf:ID=concat("MileageIns", $id)]/carads:MileageValue/text()
where  ($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>
```

Fig. 7. XQuery to Search an Annotated Web Page.

=, *Nissan*, *true*⟩, ⟨*Price*, <, *5000*, *true*⟩, ⟨*Year*, ≥, *1990*, *true*⟩, ⟨*Mileage*, <, *120000*, *true*⟩.

Given a set of concept filters, we can readily search a web page by generating an XQuery (e.g. Figure 7) over an external semantic web annotation disclosure file (e.g. Figure 6). Each filter expression generates a *let* clause to look up the corresponding extracted value, a phrase within the *where* clause to test the given condition, and an element in the *return* clause to generate 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 it 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 (imagine a checkbox that allows the user to require all filters to match). Another aspect not demonstrated by the running example is what happens when a concept is mentioned in the query, but there is no operator indicated for comparison. For example, suppose the user added "I'm okay with any body style" to the query. In this case, "body" is a keyword for the *BodyType* concept in our ontology. Since a relevant keyword is present, the system extracts a filter expression ⟨*BodyType*, ∅, ∅, *true*⟩. The generated XQuery then includes a *let* clause and output tags in the *return* clause for *BodyStyle*, similar to those for the other concepts in Figure 7. Nothing is added to the *where* clause because no additional selection is appropriate. Thus, the query result returns any *BodyType* it finds (e.g., Sedan, Hatchback, Convertible) as part of the result.

6 Concluding Remarks

We have presented an approach to semantic web-page annotation and query processing that is based on the use of data-extraction ontologies. This paper presents two major contributions:

1. A mechanism for automatically generating semantic annotations for ordinary web pages based on an extraction ontology, and
2. A means for querying these annotated web pages in a simple, free-form, natural-language manner.

Our prototype tool is in the initial stages. We do have a first version running, but queries are limited to conjunctive queries. Besides supporting only implicit AND operators, it would be useful to consider how to structure more-complex queries that involve explicit AND, OR, and NOT operators. However, we must be careful because we want users to be able to write queries in ordinary natural language, not controlled natural language. Nonetheless, there may be NLP techniques that could help us refine our query-processing accuracy. Full NLP parsing of queries might be useful sometimes, but since users are often telegraphic in their queries, we are considering "shallow parsing" as a means of identifying phrase chunks that could help us understand the query intent and map the query more accurately into the ontology. We also need to augment the current prototype to handle richer ontology structures that, for example, require more joins than do the applications we have tested thus far.

We intend to perform empirical studies to validate our approach over a broader range of queries. In prior experiments, we achieved a fairly high measure of precision and recall for data extraction [13]. We expect that our annotation and query approach will have similar performance characteristics, but we need to check this formally. Our approach is likely to be useful in those situations where rigidly precise data is not required. We assume that a human will interpret our query results much as a human typically interprets search-engine results. As we continue our work, we expect to learn more about the range of applications for which this approach is well suited.

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 web pages on the ordinary web, and some way to handle ordinary user requests. Our approach addresses these challenges.

References

1. H. Alani, S. Kim, D. Millard, M. Weal, W. Hall, P. Lewis, and N. Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21, January/February 2003.
2. L. Arlotta, V. Crescenzi, G. Mecca, and P. Merialdo. Automatic annotation of data extracted from large web sites. In *Proceedings of the Sixth International Workshop*

on the Web and Databases (WebDB 2003), pages 7–12, San Diego, California, June 2003.

3. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 36(25):34–43, May 2001.
4. A. Bernstein, E. Kaufmann, and N. Fuchs. Talking to the semantic web – a controlled english query interface for ontologies. *AIS SIGSEMIS Bulletin*, 2(1):42–47, January-March 2005.
5. F. Bry, C. Koch, T. Furche, S. Schaffert, L. Badea, and S. Berger. Querying the web reconsidered: Design principles for versatile web query languages. *International Journal on Semantic Web and Information Systems*, 1(2), April–June 2005.
6. W. Ceusters, B. Smith, and M. van Mol. Using ontology in query answering systems: Scenarios, requirements and challenges. In *Proceedings of the 2nd Network of Excellence in Computational Logic (CoLogNET) and Network of Excellence in Human Language Technologies (ElsNET) Symposium*, Amsterdam, The Netherlands, December 2003.
7. X. Chen, D. Embley, and S. Liddle. Query rewriting for extracting data behind HTML forms. In *Proceedings of the International Workshop on Conceptual Model-directed Web Information Integration and Mining*, pages 335–345, Shanghai, China, November 2004.
8. V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 109–118, Rome, Italy, September 2001.
9. H. Davulcu, G. Yang, M. Kifer, and I. Ramakrishnan. Computational aspects of resilient data extraction from semistructured sources. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Sumposium on Principles of Database Systems*, pages 136–144, Dallas, Texas, May 2000.
10. Home Page for BYU Data Extraction Group. URL: http://www.deg.byu.edu.
11. S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, K. McCurley, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Zien. A case for automated large scale semantic annotations. *Journal of Web Semantics*, 1(1):115–132, December 2003.
12. Y. Ding and S. Foo. Ontology research and development. part 1: A review of ontology generation. *Journal of Information Science*, 28(2):123–136, February 2002.
13. D. Embley, D. Campbell, Y. Jiang, S. Liddle, D. Lonsdale, Y.-K. Ng, and R. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.
14. D. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 467–478, Philadelphia, Pennsylvania, May/June 1999.
15. D. Embley and R. Kimbrell. A scheme-driven natural language query translator. In *Proceedings of the 1985 ACM Computer Science Conference*, pages 292–297, New Orleans, Louisiana, March 1985.
16. D. Embley, Y.-K. Ng, and L. Xu. Recognizing ontology-applicable multiple-record web documents. In *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001)*, pages 555–570, Yokohama, Japan, November 2001.
17. D. Embley, C. Tao, and S. Liddle. Automating the extraction of data from HTML tables with unknown structure. *Data & Knowledge Engineering*. (to appear).

18. D. Embley and L. Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *Proceedings of the Third International Workshop on the Web and Databases (WebDB2000)*, pages 123–128, Dallas, Texas, May 2000.
19. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
20. S. Handschuh, S. Staab, and F. Ciravegna. S-CREAM—Semi-automatic CREAtion of Metadata. In *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW-2002)*, Madrid, Spain, October 2002.
21. R. Heck, S. Luebke, and C. Obermark. A survey of web annotation systems. Technical report, Grinnell College, Grinnell, Iowa, 1999.
22. J. Heflin and J. Hendler. Dynamic ontologies on the web. In *Proceedings of AAAI-2000*, 2000.
23. L. Hollink, G. Nguyen, G. Schreiber, J. Wielemaker, B. Wielinga, and M. Worring. Adding spatial semantics to image annotations. In *Proceedings of the Fourth International Workshop on Knowledge Markup and Semantic Annotation (SemAnnot 2004)*, Hiroshima, Japan, November 2004. in conjunction with ISWC 2004.
24. J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swickd. Annotea: An open rdf infrastructure for shared web annotations. In *Proceedings of The Tenth International World Wide Web Conference*, pages 623–632, Hong Kong, China, May 2001.
25. A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *Journal of Web Sematics*, 2(1):49–79, December 2004.
26. A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira. A brief survey of web data extraction tools. *SIGMOD Record*, 31(2):84–93, June 2002.
27. A. Maedche, G. Neumann, and S. Staab. Bootstrapping an ontology-based information extraction system. In *Intelligent Exploration of the Web*, 2002.
28. A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. Managing semantic content for the web. *IEEE Internet Computing*, 6(4):80–87, July/August 2002.
29. A. Sheth and C. Ramakrishnan. Semantic (Web) technology in action: Ontology driven information systems for search, integration and analysis. *IEEE Data Engineering Bulletin*, 26(4):40–48, December 2003.
30. Y. Tijerino, D. Embley, D. Lonsdale, Y. Ding, and G. Nagy. Toward ontology generation from tables. *World Wide Web: Internet and Web Information Systems*. (to appear).
31. M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna. MnM: Ontology driven tool for semantic markup. In *Proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, Lyon, France, July 2002.
32. A. Wessman, S. Liddle, and D. Embley. A generalized framework for an ontology-based data-extraction system. In *Proceedings of the 4th International Conference on Information Systems Technology and its Applications (ISTA'05)*, Palmerston North, New Zealand, May 2005. (to appear).
33. XQuery 1.0: An XML Query Language, November 2003. URL: http://www.w3.org/TR/xquery/.
34. L. Xu and D. Embley. A composite approach to automating direct and indirect schema mappings. *Information Systems*. (to appear).