Toward Semantic Understanding—An Approach Based on Information Extraction Ontologies

David W. Embley

Department of Computer Science
Brigham Young University
Provo, UT 84602, USA
Email: embley@cs.byu.edu

Abstract

Information is ubiquitous, and we are flooded with more than we can process. Somehow, we must rely less on visual processing, point-and-click navigation, and manual decision making and more on computer sifting and organization of information and automated negotiation and decision making. A resolution of these problems requires software with semantic understanding—a grand challenge of our time.

More particularly, we must solve problems of automated interoperability, integration, and knowledge sharing, and we must build information agents and process agents that we can trust to give us the information we want and need and to negotiate on our behalf in harmony with our beliefs and goals.

This paper proffers the use of information-extraction ontologies as an approach that may lead to semantic understanding.

Keywords: Semantics, information extraction, high-precision classification, schema mapping, data integration, Semantic Web, agent communication, ontology, ontology generation.

1 Introduction

Semantics is a grand challenge for the current generation of computer technology. It is the key for unlocking the door, for example, to personal agents that can roam the Semantic Web and carry out sophisticated tasks for their masters, to information exchange and negotiation in e-business, and to automated, large-scale, in-silico experiments in e-science.

The *American Heritage Dictionary* (AmH 2003) defines *semantics* as "the meaning or the interpretation of a word, sentence, or other language form." The keyword here is "meaning," but meaning requires understanding, and as Berners-Lee *et al.* state in their famous Semantic Web paper, "The computer doesn't truly 'understand' [anything]." They go on to say, however, that computers can manipulate terms "in ways that are useful and meaningful to the human user." This is a key point for semantic research in computing—we only have to manipulate symbols in ways that are meaningful and useful for human users. The illusion of understanding is sufficient if the symbol manipulation is good enough to obtain meaningful and useful results and good enough to allow us to trust the results at the level required for the application.

This paper takes a tiny peck at the grand challenge of semantics by motivating a particular approach to dealing with semantics (Section 2); by giving some practical, real-world applications of the approach (Section 3); and then by showing how to build on the approach to automatically attain an automated measure of semantic agreement (Section 4). Section 5 provides a short summary and gives future directions and challenges.

Supported in part by the National Science Foundation under grant number IIS-0083127.

2 Motivation

Since computers do not truly "understand" what symbols mean, computer science researchers have the responsibility and opportunity to creatively endow computers with the ability to perform useful tasks—indeed, to perform increasingly sophisticated useful tasks. How can we succeed in raising the level of sophistication required for tomorrow's applications?

Directing our discussion particularly to semantics, we first give some foundational material by defining data, information, knowledge, and meaning. Based on this foundation, we then state our central theme, which leads us to information extraction ontologies—the basis for the particular approach to "semantic understanding" discussed here.

2.1 Foundations

As a foundation, we give a variation of the definitions for data, information, knowledge, and meaning provided originally by Meadow (1992).

- *Data*: isolated attribute-value pairs.
- *Information*: data in a conceptual framework.
- *Knowledge*: information with a degree of certainty or community agreement.
- *Meaning*: data, information, or knowledge that is relevant or actuates.

These definitions help because the give us a working basis for "meaning," which we can take to be the results we want from "semantic understanding." Although meaning may well be "in the eye of the beholder," if we can, as Jim Gray said in a recent SIGMOD interview (Winslett 2003), "[take] data and [analyze] it and [simplify] it and [tell] people exactly the information they want, rather than all the information they could have," we will succeed in truly managing information. This lets us focus on something we can—with effort—succeed in doing.

Let's assume, as many do, that "meaning" for an individual is to be handled by personal software agents, which have access to knowledge both about their masters and about the world of interest to their masters, and concentrate in this paper on the a foundational ideas needed to enable and actuate this assumption. Turning to "knowledge," which is next lower on the list, we observe that Meadow's definition of knowledge coincides with what most researchers call *ontologies*—agreed upon logical theories for an application domain, independent of any particular application (Spyns, Meersman & Jarrar 2002). Drilling further down in Meadow's definitions, we observe that logical theories are commonly conceptualized in a data model or conceptual framework, or, in Meadow's words, as information. Drilling to the bottom in Meadow's definitions, we arrive at isolated attribute-value pairs—data, the most fundamental concept for meaning.

In most of computer science, this foundation—the notion of data—is extremely weak. Our declarations of data typically only weakly classify data as *integer*, *real*, and *string* and provide only highly general operations over these classifications. To strengthen our foundation, we must have a much stronger notion of an attribute-value pair. We provide this stronger foundation through the use of data frames (Embley 1980). A *data frame*[1] "[encapsulates] the essential properties of everyday data items" such as currency, dates, weights, and measures. A data frame extends an abstract data type to include not only an internal data representation and applicable operations but also highly sophisticated representational and contextual information that allows a string that appears in a text document to be classified as belonging to the data frame. Thus, for example, a data frame for a birth date has regular expressions that recognize all forms of dates and regular expression recognizers for keywords such as "born," "born on," and "birth date" to distinguish a birth date from other dates such as the date of a meeting or a purchase date.

Our hypothesis is this: ontological conceptualization over data frames can increase shared understanding. The stronger foundation provided by data frames leads to richer, more understandable information, which, in turn, leads to a more solid bases for knowledge and the potential for increased understanding and more meaningful semantics.

2.2 Information Extraction Ontologies

We formalize ontological conceptualizations over data frames as *extraction ontologies*. In this conceptualization we fundamentally base an extraction ontology on its ability to recognize and classify value strings especially in semistructured and telegraphic text.

An *extraction ontology* is an augmented conceptual-model instance that serves as a wrapper for a narrow domain of interest such as car ads. The conceptual-model instance includes objects, relationships, constraints, and data-frame descriptions of strings for lexical objects. When we apply an extraction ontology to a document such as a web page, the ontology identifies objects and relationships and associates them with named object sets and relationship sets in the ontology's conceptual-model instance and thus wraps the page so that it is understandable in terms of the schema implicitly specified in the conceptual-model instance.

[1]The name "data frame" was coined because of the similarities to abstract data types (Liskov & Zilles 1974) and Minsky frames (1975). Minsky's theory of frames is a theory of rich symbolic structure where a frame represents a particular situation. Data frames represent data items instead of situations, but the information included and its purpose are quite similar.

```
1.  Car [-> object];
2.  Car [0:1] has Year [1:*];
3.  Car [0:1] has Make [1:*];
4.  Car [0:1] has Model [1:*];
5.  Car [0:1] has Mileage [1:*];
6.  Car [0:*] has Feature [1:*];
7.  Car [0:1] has Price [1:*];
9.  PhoneNr [1:*] is for Car [0:1];
10. Year matches [4]
11.       constant {extract "\d{2}";
12.           context "\b'[4-9]\d\b";
13.           ...
14. Mileage matches [8]
15.       keyword "\bmiles\b",
16.           "\bmi\.",
17.           "\bmi\b",
10.           "\bmileage\b",
19.           "\bodometer\b";
20. ...
```

Figure 1: Partial Car-Ads Extraction Ontology

The ontological approach to writing wrappers directly addresses the hardest part of wrapper creation, which is to make a wrapper robust so that it works for all sites, including sites not in existence at the time the wrapper is written and sites that change their layout and content after the wrapper is written.[2] Wrappers based on extraction ontologies are robust. Robust wrappers are critical: without them, we have to create by hand, or at best semiautomatically, a wrapper for every new web site encountered; with them, extracting information from new or changed web pages can be fully automatic. Ontology-based wrappers are an example of the kind of "intelligent" symbol manipulation that both gives the "illusion of understanding" and obtains meaningful and useful results.

3 Applications

We now give several applications to show the power of extraction ontologies. Our first application is not surprising—it directly uses extraction ontologies to obtain useful information from the web. It is surprising, however, at least at first glance that extraction ontologies can play a direct role in high-precision document classification, schema mapping for data integration, generating superimposed information for the Semantic Web, and agent communication.

3.1 Information Extraction

Figure 1 shows part of a car-ads ontology for information extraction. The extraction ontology includes object and relationship sets and cardinality constraints (Lines 1-9) and a few lines of the data frames (Lines 10-20). Line 1 defines *Car* to be the main object of interest. Lines 2-9 define attributes of a car as they may appear in a car-ad (e.g. Line 2 declares that a *Car* in a car-ads listing may or may not have an associated *Year* and that a *Year* is for one or more cars). The data frames use Perl regular expression syntax to describe lexical constants and keywords that signal the presence of a particular object or relationship.

Given an extraction ontology, such as the car-ads extraction ontology in Figure 1, we can apply it to text such as the car ad in Figure 2. In Figure 2 the underlines denote text recognized by the data frames in the extraction ontology. Given the recognized text,

[2]Other approaches to writing wrappers include languages for wrapper development, HTML-aware tools, natural-language-based tools, and modeling-based tools (Laender, Ribeiro-Neto, da Silva & Teixeira 2002).

```
<html>
<head>
<title>The Salt Lake Tribute Classified's</title>
</head>
...
<hr>
<h4> '97 CHEVY Cavalier, Red, 5 spd, only 7,000 miles
on her. Previous owner heart broken! Asking only $11,995.
#1415 JERRY SEINER MIDVALE, 566-3800 or 566-3888
</h4>
<hr>
...
</html>
```

Figure 2: Sample Car Advertisement

we can use the ontology, its constraints and implied relationships, to extract the information and populate the ontology with the instance data. In this sense, we "understand" the car ad.

Our general approach to information extraction consists of the following steps. (See (Embley, Campbell, Jiang, Liddle, Lonsdale, Ng & Smith 1999*a*) for full details).

1. We develop the ontological model instance over the area of interest.
2. We parse this ontology to generate a database schema and to generate rules for matching constants and keywords.
3. Given an applicable web page with multiple records (like classified ads), we invoke a record extractor that separates an unstructured web document into individual record-size chunks (Embley, Jiang & Ng 1999*b*), gathers additional associated data linked on a separate page or factored into headers or footers (Embley & Xu 2000), removes markup-language tags, and presents them as individual unstructured record documents for further processing.
4. We invoke recognizers that use the matching rules obtained from the data frames to identify potential constant data values and context keywords in the cleaned records.
5. Finally, we populate the generated database by using heuristics to determine which constants populate which records in the database. These heuristics correlate extracted keywords with extracted constants and use cardinality constraints in the ontology to determine how to construct records and insert them into the database.

Once the data is extracted, we can issue queries using a standard database query language. To make our approach general, we fix the ontology parser, web record extractor, keyword and constant recognizer, and database record generator; we change only the ontology as we move from one application domain to another.

When we have applied our car-ads extraction ontology, we have attained recall ratios in the range of 90% and precision ratios near 98%. We have also applied extraction ontologies to many other domains: apartment rentals, books, campgrounds, cell phones, computer monitors, computer software, countries, course catalogs, digital cameras, games, gems, genealogy, jewelry, jobs, movies, music, obituaries, prescription drugs, personals, restaurants, and stocks (see (DEG 2000)). Perhaps the hardest domain we have tried is obituaries, but even for this application our recall and precision ratios were still in the 90% range, except for names of relatives, where the precision dropped to about 75%.

3.2 High-Precision Classification

The web contains abundant repositories of information in web documents—indeed so much that locating information of interest for an application becomes a huge challenge. Even sorting through a tiny subset of web documents is overwhelming. How can we automatically select just those documents that have the needed information for an application?

As a step toward solving this problem, we proposed a technique for high-precision recognition of web documents that apply to an ontologically specified domain (Embley, Ng & Xu 2001). High-precision classifiers determine not only whether a document, such as a listing of classified ads in a newspaper, contains items of interest for a predefined application ontology, but also whether particular elements of interest are present in the document.

It should be clear that if we can extract the basic information in a document relative to an application domain, then we can apply heuristic measures over these extracted values to determine whether the document is sufficiently similar to documents expected in the application domain and thus do high-precision classification. The heuristics we apply work particularly well for multiple-record web documents such as classified ads or other lists described objects. We apply three heuristics: (1) a density heuristic that measures the percent of the document that appears to apply to an application ontology, (2) an expected-value heuristic that compares the number and kind of values found in a document to the number and kind expected by the application ontology, and (3) a grouping heuristic that considers whether the values of the document appear to be grouped as application-ontology records. Then, based on machine-learned rules over these heuristic measurements, we determine whether a web document is applicable for a given ontology. Our experimental results show that we have been able to achieve over 90% for both recall and precision, with an F-measure of about 95%.

Although our extraction-ontology approach to classification differs fundamentally from most text classifiers (e.g. (McCallum 1996)), the work reported in (Riloff & Lehnert 1994) takes an approach similar to ours in that it also attempts to do high-precision classification for information extraction. Like most text classifiers, (Riloff & Lehnert 1994) uses machine learning, but to obtain the desired high precision, considerably more effort must be expended to establish the basis for machine learning. Not only must documents be marked as relevant and non-relevant, but each individual relevant element plus the context for each individual relevant element must also be marked. In addition, an application-domain-specific dictionary must be created. The basic trade-off in human effort between our approach and the approach in (Riloff & Lehnert 1994) is the effort to tag the elements in the document and create the domain-specific dictionary versus the effort to create the extraction ontology. Some more recent work has been reported that uses machine learning with less human effort for doing "high-precision" classification for domain-specific search engines (McCallum, Nigam, Rennie & Seymore 1999) and focused crawling (Chakrabarti, van den Berg & Dorn 1999). By mostly using unsupervised learning, human effort can be greatly reduced. The challenge, however, is to reach high accuracy, and it may not be possible to achieve the accuracy that can be obtained with an ontology-based approach. Ultimately, some combination of the approaches may be best.

Car	*Year*	*Make*	*Model*	*Mileage*	*Price*	*PhoneNr*
0001	1999	Ford	Mustang	42,130	$10,988	405-936-8666
0002	1998	Ford	Taurus	63,168	$7,988	405-936-8666
			...			
0111	1995	ACURA	INTEGRA LS		$14,5000	
0112	1988	ACURA	Legend		$4,600.00	
0113	1992	ACURA	Legend			
			...			

Car	*Feature*
0001	Yellow
0002	Black
	...
0111	Red
0111	Auto
0111	A/C
1112	Red
	...

Figure 3: Sample Tables for Target Schema

For more information call 405-936-8666.

Year	*Vehicle*	*Price*	*Miles*	*Exterior*
1999	Ford Mustang	$10,988	42,130	Yellow
1998	Ford Taurus	$7,988	63,168	Black
1995	Ford F150 Super Cab	$6,988	92,739	Red
1995	Ford Contour GL	$3,988	95,581	Green
	...			

Figure 4: Slightly Modified Excerpt of a Table from `www.bobhowardhonda.com`

3.3 Schema Mapping for Data Integration

The schema-mapping problem for heterogeneous data integration is hard and is worthy of study in its own right (Madhavan, Bernstein & Rahm 2001). The problem is to find a *semantic correspondence* between one or more *source schemas* and a *target schema* (Doan, Domingos & Halevy 2001). In its simplest form the semantic correspondence is a set of *mapping elements*, each of which binds an attribute in a source schema to an attribute in a target schema or binds a relationship among attributes in a source schema to a relationship among attributes in a target schema. Such simplicity, however, is rarely sufficient, and researchers thus use queries over source schemas to form attributes and relationships among attributes to bind with target attributes and attribute relationships (Miller, Haas & Hernandez 2000, Biskup & Embley 2003). As it turns out, we may even need queries beyond those normally defined for database systems. Thus, we more generally define the *semantic correspondence for a target attribute* as any named or unnamed set of values that is constructed from source elements, and we define the *semantic correspondence for a target n-ary relationship* among attributes as any named or unnamed set of n-tuples over constructed value sets. The sets of values for target attributes may be constructed in any way, e.g. directly taken from source values, computed over source values, or manufactured from source attribute names or from strings in table headers or footers.

Suppose, for example, that we are interested in viewing and querying web car ads through the target database in Figure 3, whose schema is

{*Car*, *Year*, *Make*, *Model*, *Mileage*, *Price*, *PhoneNr*}, {*Car*, *Feature*}.

Figures 4 and 5 show some potential source tables. The data in the tables in Figure 3 is a small part of the data that can be extracted from Figures 4 and 5.

It is easy to see that *Year* in the source table of Figure 4 and *Year* in the table of Figure 5 map to *Year* in the target table of Figure 3. But it is harder to see that both *Exterior* in Figure 4 and *Colour* in Figure 5 map to *Feature* in Figure 3 and even harder to see that the attributes *Auto* and *A/C* in Figure 5 should map as values for *Feature*, but only for "Yes" values.

3.3.1 Matching Problems

Many matching problems arise when trying to match source tables with a target schema.

- Merged values (e.g. *Make* and *Model* are separate attributes in Figure 3 but are merged as one attribute called *Vehicle* in Figure 4).
- Multiple subsets constituting a set (e.g. color, air conditioning, and transmission features in Figure 5 are each proper subsets of the set of features in Figure 4).
- Synonyms and homonyms (e.g. *Mileage* in Figure 3 and *Miles* in Figure 4 are synonyms; *Feature* in a source table, meaning only specific kinds of features, would be a homonym of *Feature* in Figure 3, which denotes general features).
- Externally factored information (e.g. the phone number in Figure 4 is for all vehicles).
- Internally factored information (e.g. the *Make* for the second and third cars in Figure 5 is *ACURA* and the *Model* for the third car is *Legend*).
- Missing information (e.g. there are no phone numbers in Figure 5 and there is no price for the *1992 ACURA*).
- Blanks fields that carry information (e.g. the empty strings in the *Auto* and *A/C* columns of Figure 5 are not missing values, but, instead, represent "No").
- Attributes appearing as values (e.g. in Figure 5 the features *Auto* and *A/C* are attributes rather than values).

This list is not exhaustive. It certainly illustrates, however, that there are many problems to solve.

3.3.2 Matching Solutions

Rather than directly try to find mappings from source schemas to target schemas as suggested in (Miller et al. 2000, Doan et al. 2001, Madhavan et al. 2001), our approach makes use of table understanding (Lopresti & Nagy 1999) and extraction ontologies (Embley et al. 1999*a*) and results in establishing a semantic correspondence between a source schema and a target schema. Our approach has four steps.

Make	*Model*	*Year*	*Colour*	*Price*	*Auto*	*A/C*
ACURA	INTEGRA LS	1995	Red	$14,5000	Yes	Yes
	Legend	1988	Red	$4,600.00		
		1992	grey		Yes	
AUDI	A4	2000	Blue	$34,500	Yes	Yes
...						

Figure 5: Slightly Modified Excerpt of a Table from `autoscanada.com`

1. *Form Attribute-Value Pairs.* Using table understanding techniques, we determine, for example, that ⟨*Year*: 1999⟩ and ⟨*Exterior*: *Yellow*⟩ are two of the attribute-value pairs for the first record in Figure 4.

2. *Adjust Attribute-Value Pairs.* We convert, for example, the recognized attribute-value pair ⟨*A/C*: *Yes*⟩ in the first row of Figure 5 to *A/C*, meaning that this car has air conditioning, and discard ⟨*A/C*: ⟩ in the second row.

3. *Perform Extraction.* The extraction ontology recognizes, for example, that *42,130* in the first row of Figure 4 should be extracted as the *Mileage* for the first car in Figure 3 and that the first part of *Ford Mustang* should be extracted as the *Make* while the second part should be the *Model*.

4. *Infer Mappings.* Given the recognized extraction (which need not be 100%), the system can infer general mappings from source to target. Based on the extraction examples above, the system would know, for example, that the first part of the *Make and Model* strings in Figure 4 map to *Make*, and that the remaining characters in the strings map to *Model*. It would also know that the *Price* values in Figure 4 map to *Price* in the target (Figure 3), and thus that $14,5000 is a *Price* even though it is an obvious typo[3] that does not match the form of a price.

We have implemented a system to process tables as just described. To test our system, we processed 46 tables containing car ads, which included 319 mappings. Our system correctly discovered 296 (92.8%) mappings, missed 23 (7.2%), and incorrectly declared 13 false mappings (4.2% of the 309 declared mappings). Of the 296 mappings we correctly discovered, 121 (40.9%) were direct, in the sense that attributes in the source and target schemas were identical, and 175 (59.1%) were indirect. Of the 175 indirect matches, 28 had synonyms, 1 had "Yes"/"No" values, 91 required collecting values from more than one table column, 2 had externally factored values, and 53 were merged—like *Make* and *Model* are merged under *Vehicle* in Figure 4—and needed to be split. (See (Embley, Tao & Liddle 2004) for a detailed discussion.)

Once the system is build, the cost for being able to do these mappings for any application is the cost of building an extraction ontology for the application. We were able to retarget our car-ads application to a cell-phone application in a few dozen person-hours. The results in this new application domain were similar (Embley et al. 2004).

We also implemented a similar system for mapping between attributes in ordinary relational database tables. For this system we tested three data sets obtained from the University of Washington (Doan et al. 2001). The applications for these data sets were university course schedules, university faculty, and real estate. Altogether our system correctly found 1335 of 1396 (96%) of the direct mappings and 479 of 510 (94%) of the indirect mappings while declaring only 98 (5% of 1906 mappings) incorrectly. (See (Xu 2003) for full details.)

[3]This typo actually appeared—was not made up just for illustration.

3.4 Semantic Web

Research is underway in universities and companies around the world to develop the *Semantic Web*—the next generation of the web. The idea of the Semantic Web is to add semantics to web content in order to make it easier to find and use for both humans and machines. Adding formal semantics to the web will aid in everything from resource discovery to the automation of all sorts of tasks (Koivunen & Miller 2002).

With all the advantages of the Semantic Web, what keeps it from reaching a critical mass where it will gain widespread acceptance and use? One reason is the newness of the area leading to a lack of necessary tools to help people publish and use Semantic Web content. To a large extent, however, it is simply the lack of useful content. For years people have been publishing web documents on nearly every topic imaginable and building systems to continually generate new content, so there is a vast amount of human readable information on the web. It is hard to imagine rewriting, by hand, the current web content to be accessible to Semantic Web agents.

Is it possible to automatically (or at least semi-automatically) rewrite the web content (or at least some of the web content) so that it will be accessible to Semantic Web agents? Perhaps the most daunting task in making web data accessible to Semantic Web agents is extracting the data from web pages. If we could automatically extract information from web pages and reconstitute it as metadata superimposed over the original web pages, we may be able to enable the Semantic Web as it is envisioned.

Currently, the basic description language for metadata on the Semantic Web is RDF (Resource Description Framework) (RDF 2003*b*). RDF is a domain-independent model for describing resources, where a resource is anything that can be represented by a Uniform Resource Identifier, including web pages, parts of web pages, or even physical objects. RDF helps give structure to web content, but does not describe the resources themselves. For this we need ontologies. RDF Schema (RDFS) is a simple ontology language written in RDF that allows the creation of vocabularies with classes, properties, and subclass/superclass hierarchies (RDF 2003*a*). DAML+OIL (DARPA Agent Markup Language + Ontology Inference Layer) is an extension of RDFS that allows finer-grained control of classes and properties with features such as cardinality constraints and inverses of properties (Fensel, van Harmelen & Horrocks 2002).

As an experiment to investigate the possibility of bridging the gap between the HTML web and the Semantic Web, we built a system to extract data from the web with respect to a DAML+OIL ontology and reconstitute it as RDF data superimposed over the original web site (Chartrand 2003). Figure 6 shows an overview of our RDF-extraction system. The first input to the system, on the left in Figure 6, is a