Automatically Extracting Ontologically Specified Data from HTML Tables with Unknown Structure

David W. Embley[1] Cui Tao[1] Stephen W. Liddle[2]

[1]Department of Computer Science
[2]School of Accountancy and Information Systems
Brigham Young University, Provo, Utah 84602, U.S.A.
{embley,ctao}@cs.byu.edu, liddle@byu.edu

Abstract

Data on the Web in HTML tables is mostly structured, but we usually do not know the structure in advance. Thus, we cannot directly query for data of interest. We propose a solution to this problem based on document-independent extraction ontologies. The solution entails elements of table understanding, data integration, and wrapper creation. Table understanding allows us to recognize attributes and values, pair attributes with values, and form records. Data-integration techniques allow us to match source records with a target schema. Ontologically specified wrappers allow us to extract data from source records into a target schema. Experimental results show that we can successfully map data of interest from source HTML tables with unknown structure to a given target database schema. We can thus "directly" query source data with unknown structure through a known target schema.

1 Introduction

The schema-mapping problem for heterogeneous data integration is hard and is worthy of study in its own right [MBR01]. The problem is to find a *semantic correspondence* between one or more *source schemas* and a *target schema* [DDH01]. 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 [MHH00, BE02]. Furthermore, as we shall see in this paper, we may also 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.

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
			...			
0011	1992	ACURA	legend		$9500	
0012	2000	AUDI	A4		$34,500	
0013	1985	BMW	325e		$2700.00	
			...			

Car	Feature
0001	Yellow
0001	Power Steering
	...
0002	Black
0002	Power Brakes
	...
0011	grey
0011	Auto
0011	AM/FM
	...

Figure 1: Sample Tables for Target Schema

We limit our discussion here to HTML tables found on the Web.[1] We consider these HTML tables to be our sources. Our target schema is an augmented conceptual-model instance (defined, explained, and illustrated in Section 2).

As a running example, we use car advertisements, which are plentiful on the Web and which often present their information in tables. Suppose, for example, that we are interested in viewing and querying Web car ads through the target database in Figure 1, whose schema is

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

Figures 2 [Bob02], 3 [Bob02], and 4 [Aut01] show some potential source tables. The data in the tables in Figure 1 is a small part of the data that can be extracted from Figures 2, 3, and 4.

1.1 Matching Problems

It is easy to see that *Year* in the source table in Figure 2 as well as *Year* in the source table in Figure 4 map to *Year* in the target table in Figure 1. It is not so easy, however, to see that both *Exterior* in Figure 2 and *Colour* in Figure 4 map to *Feature* in Figure 1 and even harder to see that we should map the attributes *Auto*, *Air Cond.*, *AM/FM*, and *CD* in Figure 4 as values for *Feature* in Figure 1, but only for "Yes" values.

In the following list we describe many *matching problems* that arise when trying to match source HTML tables with a target schema.

- *Merged Attributes.* *Make* and *Model* are separate attributes in Figure 1 but are merged as one attribute in Figure 2.

[1]The problems encountered in HTML tables are more than sufficient for this investigation. Table extraction within the broader context of images of paper tables and other types of electronic tables [LN99b] is also possible.

Year	Make and Model	Price ▼	Miles	Exterior	Photo
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	
1994	Ford F150	$4,488	147,588	Green/Tan	
1995	Ford Contour GL	$3,988	95,581	Green	
1994	Ford Probe	$3,988	90,370	White	
1994	Ford Taurus LX	$2,988	119,784	Blue	

Figure 2: Table from www.bobhowardhonda.com

- *Subsets.* *Exterior* in Figure 2 and *Colour* in Figure 4 contain colors. Colors in the target are a special kind of *Feature* and thus the set of colors in Figures 2 and 4 are subsets of the feature values we want for Figure 1. Indeed, these are proper subsets since there are also many other feature values in Figures 2, 3, and 4.

- *Synonyms.* *Mileage* in Figure 1 and *Miles* in Figure 2 have the same meaning, but the attribute names are not the same.

- *Extra Information.* The tables in Figure 1 make no request for photographs, which are present in Figure 2.

- *Information Hidden Behind a Link.* The values for the attribute *Make and Model* are linked to further information. Clicking on *Ford Taurus* in Figure 2 yields the information in Figure 3.

- *A List Rather than a Table.* A one dimensional table and a list are similar in appearance. *Features* in Figure 3 is a list, but could just as easily have been formatted as a table. Although it is a list, we nevertheless wish to match *Features* in Figure 3 with *Feature* in Figure 1.

- *Tables for Layout Rather than Information.* The structured information beginning with *Send Me More Information* in Figure 3 is formatted using HTML table constructs, but this "table" does not have the attribute-value characteristics of a table. Indeed, we do not consider it to be a table even though it is constructed as an HTML table.

- *Nested Tables.* The table beginning with *Price* then *Mileage* in Figure 3 is nested within the larger layout table (which could just have easily been a standard informational table).

- *Position and Composition of Attributes.* The nested table in Figure 3 has its attributes in the left column, rather than in the top row, which is more typical. The table in Figure 5

Pre-Owned Inventory

At Howard Auto Group we have created an Internet sales department to give our customers an alternative buying experience. Once you have found a vehicle you like we will be glad to **give you our lowest no haggle price right up front!** Then if you like that price you can complete the transaction with your Internet manager. He can also quote you a payment, interest rate and if you have a trade in give you an evaluation of your trade in. Remember the Internet department is designed to provide the fastest and friendliest service to the Internet user and to ensure a totally different buying experience! If you have any questions please feel free to give a call at **405-936-8666** or toll free **877-944-2842** or drop us an e-mail. Your Internet sales staff at Howard AutoNet is waiting to help you. Kyle, Brendon, James, Michelle, Rory and Mitesh.

Send Me More Information

1998 Ford Taurus $7,988

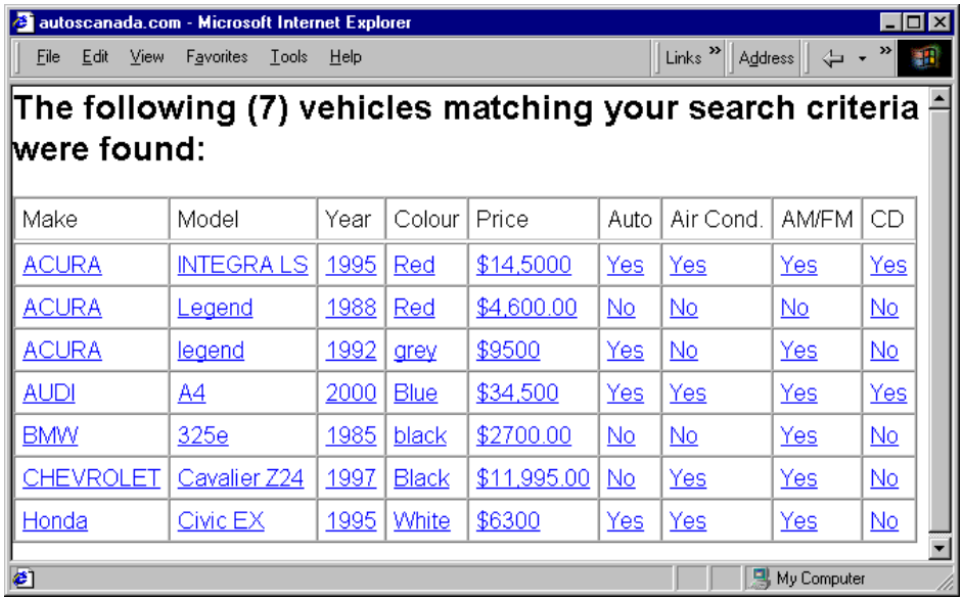
Features

- POWER STEERING
- POWER BRAKES
- POWER DOOR LOCKS
- POWER WINDOWS
- CASSETTE PLAYER
- TRIP ODOMETER
- TACHOMETER
- AIR CONDITIONING
- TILT STEERING WHEEL
- CRUISE CONTROL
- DRIVER SIDE AIR BAG
- PASSENGER SIDE AIR BAG
- DELUXE WHEEL COVERS
- REAR DEFROSTER
- DRIVER SIDE POWER MIRRORS
- SPLIT FRONT BENCH SEAT
- RECLINING SEATS
- CLOTH UPHOLSTERY
- CHILD-PROOF LOCK(S)

Click on photo to enlarge

Price	$7,988
Mileage	63,168 miles
Body Type	Car
Body Style	4 DR Sedan
Exterior	Black
Transmission	Automatic
Engine	3.0 L V-6
Doors	4
Fuel Type	Gas
Stock Number	22764
VIN	1FAFP52U2WA139879

Figure 3: Subtable Linked from www.bobhowardhonda.com



Make	Model	Year	Colour	Price	Auto	Air Cond.	AM/FM	CD
ACURA	INTEGRA LS	1995	Red	$14,5000	Yes	Yes	Yes	Yes
ACURA	Legend	1988	Red	$4,600.00	No	No	No	No
ACURA	legend	1992	grey	$9500	Yes	No	Yes	No
AUDI	A4	2000	Blue	$34,500	Yes	Yes	Yes	Yes
BMW	325e	1985	black	$2700.00	No	No	Yes	No
CHEVROLET	Cavalier Z24	1997	Black	$11,995.00	No	Yes	Yes	No
Honda	Civic EX	1995	White	$6300	Yes	Yes	Yes	No

Figure 4: Table from www.autoscanada.com

has compond attributes (e.g. *CITY* and *HWY* are subdivisions of *FUEL ECONOMY*). Furthermore, attributes may appear both in the top row and in the left column and may, at the same time, be compound. Sometimes it is even hard to tell: are the *Honda Civic* strings in Figure 5 attributes or values with implicit attributes?

- *Missing Information.* The tables in Figure 1 expect a phone number, but none of the tables in Figures 2, 3, or 4 contain a phone number.

- *External Factored Information.* Although no phone number appears in the tables in Figures 2 or 3, phone numbers do appear in the text above the tables in Figure 3. Further, although not shown, the footer information on the page below the table in Figure 2 also contains phone numbers (including a fax number). A value, such as a dealer phone number, that applies to all records in a table is often factored out, external to the table, and displayed only once.

- *Internal Factored Information.* Sometimes cars are grouped by years or by makes or by some other value, which is then factored out and written only once for each group. We may, for example, write *ACURA* in only the first row in Figure 4, leaving the *Make* entry blank in rows two and three. Double factoring and triple factoring (and more) is also possible. We may, for example, factor the model *Legend* for rows two and three in Figure 4 in addition to factoring *ACURA* for rows one through three.

- *Multiple Occurrences of the Same Attribute-Value Pair.* The price for the Ford Taurus in Figures 2 and 3 appears three times, once under *Price* in Figure 2, once as the value for the

	FUEL ECONOMY		PRICE		ENGINE
	HWY	CITY	INVOICE	RETAIL	
2001 Honda Civic DX	39mpg	33mpg	$NL	$12,810	1.7L I4 115HP
2001 Honda Civic HX	44mpg	36mpg	$NL	$13,610	1.7L I4 117HP
2001 Honda Civic LX	39mpg	33mpg	$NL	$14,910	1.7L I4 115HP
2001 Honda Civic EX	37mpg	32mpg	$NL	$16,510	1.7L I4 127HP

Figure 5: Table with a More Complex Attribute Structure

Price attribute in the nested table in Figure 3, and once at the top of the layout table in Figure 3. (Luckily, the values are all the same.) Other values also appear more than once. The number of miles, in fact, appears with two different attributes, once with *Miles* and once with *Mileage*.

- *Multiple Values where One is Expected.* The tables in Figure 1 expect at most one contact phone number for each vehicle, but there may be several as Figure 3 shows.

- *Attribute as Value.* In Figure 4, observe that the features *Auto*, *Air Cond.*, *AM/FM*, and *CD* are all attributes rather than values. Here, we must understand that *Yes* and *No* are not the values; rather they indicate whether the values *Auto*, *Air Cond.*, *AM/FM*, and *CD* should be included as *Feature* values in the tables in Figure 1.

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

1.2 Matching Solutions—Our Contribution

Rather than directly try to find mappings from source schemas to target schemas as suggested in [MHH00, DDH01, MBR01, BE02], our contribution in this paper is to argue for a different approach, show that it works, and explain why it may be superior. The approach requires table understanding [LN99b] and extraction ontologies [ECJ+99] and results in establishing a semantic correspondence between a source schema and a target schema. Our approach includes four steps.

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 2.

2. *Adjust Attribute-Value Pairs.* We convert, for example, the recognized attribute-value pair <*CD*: *Yes*> in row one of Figure 4 to *CD*, meaning that this car has a CD player, and the the pair <*CD*: *No*> in row two to a null string, meaning that this car has no CD player.

3. *Perform Extraction.* The extraction ontology recognizes, for example, that the *42,130* in the first row in Figure 2 should be extracted as the *Mileage* for the first car in Figure 1 and that the first part of the value *Ford Mustang* should be extracted as the *Make* while the second part should be extracted as the *Model*.

4. *Infer Mappings.* Given the recognized extraction (which, by the way, need not be 100%), the system can infer the general mapping from source to target. Based on the extraction examples above, the system would know, for example, that the *Miles* values in Figure 2 map to *Mileage* in the target (Figure 1), that the first part of the *Make and Model* strings map to *Make*, and that the remaining characters in the strings map to *Model*.

We present the details of our contribution in the remainder of the paper as follows. Section 2 describes extraction ontologies. Section 3 then provides the details for each of the four steps of our approach. In Section 4, we report the results of an experiment we conducted in which we derived source-target mappings for several dozen HTML tables for car advertisements, which we found on the Web. We summarize and point to future research work in Section 5.

2 Extraction Ontologies

An extraction ontology is a 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 over these objects and relationships, and descriptions of strings for lexical objects and keywords denoting the presence of objects and relationships among objects. When we apply an extraction ontology to a Web page, the ontology identifies the 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. The hard part of writing a wrapper for extraction is to make it 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. Wrappers based on extraction ontologies are robust.[2] Robust wrappers are critical to our approach: without them, we may have to create (by hand or at best semiautomatically) a wrapper for every new table encountered; with them, the approach can be fully automatic.

[2]Page-specific, handwritten wrappers (e.g. the early wrappers produced for TSIMMIS [CGMH+94]) are not robust. Machine-learning-based wrappers (e.g. [KWD97, Sod99]) are not robust since new and changed pages must be annotated and learned. Wrappers that automatically infer regular expressions for Web pages (e.g. [CMM01]) are robust in the sense that the regular-expression generator only needs to be rerun for new and changed pages; however, high page layout regularity is required, an assumption that often fails, but which we intend to consider in our future work with tables. Extraction ontologies (e.g. [ECJ+99]) are robust because they are based on conceptual-model specifications of a domain of interest, not on page layout. Although they are hand-crafted, as ontologies typically are, our experience shows that an expert can create a reasonably good extraction ontology for a narrow domain of interest such as car ads in a few dozen person-hours.

```
 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:*];
 8.  PhoneNr [1:*] is for Car [0:1];
 9.  Year matches [4]
10.      constant {extract "\d{2}";
11.                context "\b'[4-9]\d\b";
12.                substitute "^" -> "19"; },
13.                ...
14.  Mileage matches [8]
15.      ...
16.      keyword "\bmiles\b", "\bmi\.", "\bmi\b",
17.              "\bmileage\b", "\bodometer\b";
18.  ...
```

Figure 6: Car-Ads Extraction Ontology (Partial)

An extraction ontology consists of two components: (1) an *object/relationship-model instance* that describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets, and (2) for each object set, a *data frame* that defines the potential contents of the object set. A data frame for an object set defines the lexical appearance of constant objects for the object set and establishes appropriate keywords that are likely to appear in a document when objects in the object set are mentioned. Figure 6 shows part of our car-ads application ontology, including object and relationship sets and cardinality constraints (Lines 1-8) and a few lines of the data frames (Lines 9-18).

An object set in an application ontology represents a set of objects which may either be lexical or nonlexical. Data frames with declarations for constants that can potentially populate the object set represent lexical object sets, and data frames without constant declarations represent nonlexical object sets. *Year* (Line 9) and *Mileage* (Line 14) are lexical object sets whose character representations have a maximum length of 4 characters and 8 characters respectively. *Make*, *Model*, *Price*, *Feature*, and *PhoneNr* are the remaining lexical object sets in our car-ads application; *Car* is the only nonlexical object set.

We describe the constant lexical objects and the keywords for an object set by regular expressions using Perl syntax.[3] When applied to a textual document, the `extract` clause (e.g. Line 10) in a data frame causes a string matching a regular expression to be extracted, but only if the `context` clause (e.g. Line 11) also matches the string and its surrounding characters. A `substitute` clause (e.g. Line 12) lets us alter the extracted string before we store it in an in-

[3]Thus, for example, "`\b`" indicates a word boundary, "`\d`" indicates a numeric digit, and so forth.

termediate file. (For example, the *Year* data frame treats a year written "`'95`" as the constant "`1995`".) We also store the string's position in the document and its associated object set name in the intermediate file. One of the nonlexical object sets must be designated as the *object set of interest—Car* for the car-ads ontology, as indicated by the notation "`[-> object]`" in Line 1.

We denote a relationship set by a name that includes its object-set names (e.g. *Car has Year* in Line 2 and *PhoneNr is for Car* in Line 8). The *min:max* pairs in the relationship-set name are *participation constraints*. *Min* designates the minimum number of times an object in the object set can participate in the relationship set and *max* designates the maximum number of times an object can participate, with * designating an unknown maximum number of times. The participation constraint on *Car* for *Car has Feature* in Line 6, for example, specifies that a car need not have any listed features and that there is no specified maximum for the number of features listed for a car.

When we apply an extraction ontology to extract data, we first find record boundaries and divide a document into chunks of information, one for each record. (Although difficult in general, for tables whose tuples correspond to the object of interest, finding record boundaries is straightforward.) We then apply all regular expressions to each record, one at a time. For each record, the result is list of five-tuples: <*Recognized Character String*, *Object Set of String Recognizer*, *Whether the Recognized String is a Keyword or a Value*, *Recognized String Start Location*, *String Length*>. To this list, we apply five heuristics.

1. *Subsumed/Overlapping Constants.* We assume that no *Recognized String* and no overlapping part of a *Recognized String* corresponds with more than one *Object Set*. We thus discard the 5-tuples of *Recognized Strings* subsumed by other *Recognized Strings* and also discard the 5-tuples of *Recognized Strings* that overlap the tail-end of other *Recognized Strings*.

2. *Keyword Proximity.* For any *Recognized String* that is a possible *Value* for more than one *Object Set*, the closest *Keyword* (if any) disambiguates the possibilities. We thus discard all ambiguous *Value* 5-tuples except the closest ones to disambiguating *Keywords*. Having finished using *Keywords*, we discard all *Keyword* 5-tuples.

3. *Functional Relationships.* If only one *Value String* for an *Object Set* should appear and only one remains, it is selected as the value for the *Object Set* for the record.

4. *Nonfunctional Relationships.* If several *Value Strings* may appear for an *Object Set*, we take them all as values for the *Object Set* for the record.

5. *First Occurrence.* If only one *Value String* for an *Object Set* should appear but several remain, the first is selected as the value for the *Object Set* for the record.

See [ECJ+99] for details.

3 Derivation of Source-to-Target Schema Mappings

We assume that each tuple in the top-level table corresponds to a primary object of interest.[4] One consequence of this assumption is that we can simply generate an object identifier for each of these objects. Indeed, this is how we obtain the values under the attribute *Car* in Figure 1. Another consequence of this assumption is that we can easily group the source information into record chunks, one chunk of information for each object of interest. The information chunk for the *1998 Ford Taurus* in Figure 2, for example, is the second tuple in the table plus all the information in Figure 3. Having record chunks allows us to more easily build the atomic relationships between an object of interest and its associated data once we find the semantic correspondence for each target attribute. Hence, we are able to reduce the problem of finding a semantic correspondence to just finding the semantic correspondence for each target attribute A, which we defined earlier as the problem of finding the set of values constructed from source elements that corresponds to A.

We accomplish this objective using a "back-door" approach. Instead of directly searching for a mapping that associates each target attribute A with a value set in a source, we use our extraction ontology to search for values in the source that are likely to be found in the value set for A. Then, from the pattern of values we find, we infer what the mapping must be. This approach more easily allows us to recognize some of the unusual indirect mappings we are likely to encounter such as *Attribute as Value*, *External Factored Information*, and *Merged Attributes* as discussed and illustrated in the introduction. The following four subsections correspond to the four steps of our proposed approach.

3.1 Form Attribute-Value Pairs

The table understanding problem takes as input a table (for our work here, an HTML table) and produces standard records as output. Each record produced is a set of attribute-value pairs. A successful table-understanding system, for example, would produce the third record in the table in Figure 4 as

{<*Make*: *ACURA*>, <*Model*: *legend*>, <*Year*: 1992>, <*Colour*: *grey*>,
<*Price*: $9500>, <*Auto*: *Yes*>, <*Air Cond.*: *No*>, <*AM/FM*: *Yes*>, <*CD*: *No*>}

The hard part of table understanding is to recognize which cells contain the attributes and which contain the values and then to recognize which attributes go with which values. If we could depend on the attributes always being in column headers with one attribute for every column, attribute/value identification would be much simpler; but this is not always the case even though

[4] We do not address fundamental mismatches of primary objects in our work here. Whether this approach extends to cases where we cannot easily find an alignment for the main objects of interest (cars in our example here) is a question for future research.

Year	Make	Model	Price
1995	Ford	F150 Super Cab	$6,988
		Contour GL	$3,988
	ACURA	INTEGRA LS	$14,500
	Honda	Civic EX	
1994	Ford	F150	$4,488
		Probe	$3,988
		Taurus LX	$2,988

(a)

Year	Make	Model	Price
1995	Ford	F150 Super Cab	$6,988
1995	Ford	Contour GL	$3,988
1995	ACURA	INTEGRA LS	$14,500
1995	Honda	Civic EX	
1994	Ford	F150	$4,488
1994	Ford	Probe	$3,988
1994	Ford	Taurus LX	$2,988

(b)

Figure 7: Internal Factoring in Tables

it is common. Furthermore, since HTML table creators do not always use the tags <*th*>, <*td*>, and <*tr*> as they were intended, we cannot use HTML tags to solve this problem. [LN99a] has a solution for a common subclass of HTML tables, but not for all HTML tables.[5]

Once we have identified attributes, we can immediately associate each cell in the grid layout of a table with its attribute. If the cell is not empty, we also immediately have a value for the attribute and thus an attribute-value pair. If the cell is empty, however, we must infer whether the table has a value based on internal factoring or whether there is no value. Figure 7a shows an example of internal factoring. The empty *Year* cell for the *Contour GL*, for example, is clearly *1995*, whereas the *Price* for the *Honda* is simply missing. We recognize internal factoring in a two-step process:[6] (1) we detect potential factoring by observing a pattern of empty cells in a column, preferrably a leftmost column or a near-leftmost column; (2) we check to see whether adding in the value above the empty cell helps complete a record by adding a value that would otherwise be missing.

Once we recognize the factoring in a table, we can rewrite the schema as a nested schema that reflects the factoring and then unnest the table to distribute factored values and to return the schema for the nested table to an unnested schema. Textually, we represent a nested component of a schema by $(A_i, ..., A_n)$* where the A_i's are attribute names. In general, nested components may appear inside of and along side of other nested components. The nested schema that defines the internal factoring for the table in Figure 7a is $Year$, ($Make$, ($Model$, $Price$)*)*.

To unnest a table with a nested schema, we use a μ operator whose definition is based on the unnest operator in [KS91]. We write $\mu_N t$ to unnest the nested component N of nested table t.

[5] In our work here, we accept this simpler solution for now, but in some of our other work [Haa98, Tub01] we have explored the attribute-recognition problem, and in future work, we intend to provide a general solution. Once we have a general solution, the approach we propose here carries over without change.

[6] This procedure is more fully described in [EX00], where we explain how we use a multi-dimensional cosine measure and a hill-climbing procedure to recognize factored values and appropriately distribute them to their proper records.

To unnest table T_a in Figure 7a, for example, we can apply $\mu_{(Make,\ Model,\ Price)^*}\mu_{(Model,\ Price)^*}T_a$, which yields the table in Figure 7b. Here, we start with $Year$, ($Make$, ($Model$, $Price$)*)*. After the first operation $\mu_{(Model,\ Price)^*}$, the schema is $Year$, ($Make$, $Model$, $Price$)* and the *Make*s have been distributed to the *Model*s, i.e. *Ford* appears in the empty cells in the *Make* column in Figure 7a in our example. Then after the second operation $\mu_{(Make,\ Model,\ Price)^*}$, which distributes the *Year*s to each empty cell in the *Year* column, we have the table in Figure 7b. Alternatively, we could have achieved the same result by applying $\mu_{(Model,\ Price)^*}\mu_{(Make,\ (Model,\ Price)^*)^*}T_a$, which first distributes the years to each make and then distributes *Year-Make* pairs to each *Model*. In either case, once we have resolved internal factoring with the μ operator, we immediately have the table's attribute-value pairs.

3.2 Adjust Attribute-Value Pairs

After discovering attribute-value pairs, we make some adjustments to prepare each record for data recognition by means of an extraction ontology. We format attribute-value pairs for easy recognition by the extraction ontology. We add in linked sub-information for each record (i.e. we add the information in Figure 3 to the ordered pairs for the second record in Figure 2). If we wish to process nontext items such as icons, we could replace them with text; for example, we could replace a color-swatch icon with the name of the color.[7] Finally, we process Boolean indicators, such as "*Yes/No*" in Figure 4, by replacing them with attribute-name values.

For our running example, the adjusted attribute-value pairs in the third record in Figure 4 become

Make: *ACURA*; *Model*: *legend*; *Year*: 1992; *Colour*: *grey*; *Price*: $9500; *Auto*; *AM/FM*;

Here, the Boolean-valued attribute-value pair <*Auto*: *Yes*> has become simply *Auto*, meaning that the car has an automatic transmission, and the pair <*AM/FM*: *Yes*> has become *AM/FM*, meaning that the car has an AM/FM radio. Further, the attributes for the *No* values, *Air Cond.* and *CD* have disappeared altogether, meaning that the car has neither air conditioning nor a CD player.

When attribute names are the values we want and the values are some sort of Boolean indicator (e.g. Yes/No, True/False, 1/0, cell checked or empty), we transform the Boolean indicators into attribute-name values with the help of a β operator which we introduce here. Syntactically we write $\beta^A_{T,F}r$ where A is an attribute of relation r and T and F are respectively the Boolean indicators for the *True* value and the *False* value given as A values in r. The result of the β operator is r with the *True* values of the A column replaced by the string A and the *False* values of A replaced by the null string. As an example, consider $\beta^{Auto}_{Yes,No}\beta^{Air\ Cond.}_{Yes,No}\beta^{AM/FM}_{Yes,No}\beta^{CD}_{Yes,No}T$ which transforms the table T in Figure 4 to the table in Figure 8.

[7] Our current implementation does not replace nontext items with text. We leave this for future work.

Make	Model	Year	Colour	Price	Auto	Air Cond.	AM/FM	CD
ACURA	INTEGRA LS	1995	Red	$14,5000	Auto	Air Cond.	AM/FM	CD
ACURA	Legend	1988	Red	$4,600.00				
ACURA	legend	1992	grey	$9500	Auto		AM/FM	
AUDI	A4	2000	Blue	$34,500	Auto	Air Cond.	AM/FM	CD
BMW	325e	1985	black	$2700.00			AM/FM	
CHEVROLET	Cavalier Z24	1997	Black	$11,995.00		Air Cond.	AM/FM	
Honda	Civic EX	1995	White	$6300	Auto	Air Cond.	AM/FM	

Figure 8: Table in Figure 4 Transformed by the β Operator

3.3 Perform Extraction

Once we have adjusted attribute-value pairs as just discussed, we apply our extraction ontology. For our running example, the extraction for the third record in Figure 4 yields

{<*Car*: 0011>, <*Year*: 1992>, <*Make*: *ACURA*>, <*Model*: *legend*>, <*Mileage*: >,
<*Price*: $9500>, <*PhoneNr*: >},
{<*Car*: 0011>, <*Feature*: *grey*>},
{<*Car*: 0011>, <*Feature*: *Auto*>},
{<*Car*: 0011>, <*Feature*: *AM/FM*>}.

We emphasize that our extraction ontology is capable of extracting from unstructured text as well as from structured text. Indeed, we can directly extract the phone numbers and features from the text, list, and horizontal table in Figure 3.[8]

For direct extraction we introduce the ϵ operator, which is based on a given extraction ontology. We define $\epsilon_S t$ as an operator that extracts a value, or values, from unstructured or semistructured text t for object set S in the given extraction ontology O according to the extraction expression for S in O. The ϵ operator extracts a single value if S functionally depends on the object of interest x in O, and it extracts multiple values if S does not functionally depend on x. As an example, $\epsilon_{PhoneNr} P$ extracts *405-936-8666* from the unstructured text in page P in in *Figure* 3 and returns it as the single-attribute, single-tuple, constant relation {<*PhoneNr*: *405-936-8666*>}.

We can use the ϵ operator in conjunction with a natural join to add a column of constant values to a table. For example, assuming that the phone number *405-936-8666* appears in page P with the table in Figure 2, which indeed it does, we could apply $\epsilon_{PhoneNr} P \bowtie T$ to add a column for *PhoneNr* to table T in Figure 2.

[8]For the results we obtain in Section 4, our implementation uses this technique. We process top-level tables using table understanding as explained here, but we process pages linked from individual records only by applying our extraction ontology to the raw semistructured information found in a linked page.

At this point we could take a data-warehousing approach and directly insert this extracted information into a global database as Figure 1 implies. Alternatively, instead of populating the global database, we can use this information to infer a mapping from the source to the target and extract information from sources whenever a query is posed against the global database schema.

3.4 Infer Mappings

We record the sequence of transformations produced when we form attribute-value pairs and when we adjust attribute-value pairs in preparation for extraction, and we observe the correspondence patterns obtained when we extract tuples with respect to a given target ontology. Based on this sequence of transformations and these correspondence patterns, we can produce a mapping of source information to a target ontology. As a simple example, consider mapping the table T_a in Figure 7a to the target schema for the tables in Figure 1. We first apply the μ operator to do the unnesting and obtain table T_b in Figure 7b. We then observe that objects extracted for the *Year* object set in the target come from the *Year* column in T_b. Similarly, for *Make*, *Model*, and *Price*, we also observe a direct correspondence. Hence, we can record the semantic correspondence of T_a and the target schema as the mapping $Year = \pi_{Year}\mu_{(Make,\ Model,\ Price)^*}\mu_{(Model,\ Price)^*}T_a$, $Make = \pi_{Make}\mu_{(Model,\ Price)^*}T_a$, $Model = \pi_{Model}T_a$, and $Price = \pi_{Price}T_a$.

Creating an inferred mapping has two important advantages. (1) The global view can be virtual. Since we have a formal mapping, we can translate any query applied to the global view to a query on the source, optimize it, execute it, and return the results from the source for the global query.[9] (2) We can obtain additional values not recognized by the ontology, but which are nevertheless valid values in the source. For example, *Super Cab* may not be technically part of the model for the *1995 Ford F150* in Figure 2, and the ontology may therefore not recognize it as part of the model. Nevertheless, someone declared *Super Cab* to be part of the model, and we should therefore extract it as such. Using the mapping, we extract full strings under *Model* in Figure 7a and thus we obtain *F150 Super Cab* as the model for the *1995 Ford* even though the extraction ontology may only pick up *F150* as the model. As another example, the mapping approach would obtain *all* the *Features* in the list in Figure 3 even though the ontology may not recognize all of them as features. When we use the mapping, we generalize over the structure and infer additional information not specifically recognized by the ontology.[10]

[9]Since the main contribution of this paper is the derivation of the mappings, not local query rewriting [LRO96, Ull97], we leave for future research a full explanation of this procedure. The idea, however, is quite straightforward once we have the semantic correspondence. Since we know the record structure of the source, we can add object identifiers for each value in the value sets. We can then join over these OID-augmented value sets to obtain a universal relation. Since the mappings result in sets associated with target attributes, if we add columns of null strings for each target attribute with no correspondence, we will have a universal relation over the target attributes. We can now project onto the schema for each of the target tables. We thus have a standard view definition, which we can substitute for each of the table references in a global query. We can then optimize the query and execute it on the source, returning only those values from the source that contribute to the global query result.

[10]For future research, we are considering the possibility of strengthening recognizers for extraction ontologies

Make	Model
Ford	Mustang
Ford	Taurus
Ford	F150 Super Cab
Ford	F150
Ford	Contour GL
Ford	Probe
Ford	Taurus LX

Figure 9: Columns Added to the Table in Figure 2 by the δ Operator

To complete our task, we now define a few more operators. These operators, together with the ones we defined earlier, provide the complete set of operators we need for mapping all the HTML tables we have encountered.[11]

For merged attributes we need to split values. We can divide values into smaller components with a δ operator which we introduce here. We define $\delta^{A}_{B_1,...,B_n} r$ to mean that each value v for attribute A of relation r is split into $v_1, ..., v_n$, one for each new attribute $B_1, ..., B_n$ respectively. Associated with each B_i is a procedure p_i that defines which part of v becomes v_i. In this paper we specify each procedure p_i by regular expressions with `extract` and `context` phrases similar to those defined for extraction ontologies discussed earlier in Section 2. The result of the δ operator is r with n new attributes, B_1, ..., B_n, where the B_i value on row k is the string that results from applying p_i to the string v on row k for attribute A. As an example, consider $\delta^{Make\ and\ Model}_{Make,Model} T$, where T is the table in Figure 2, the expression associated with *Make* is `extract "\S+" context "\S+\s"` which extracts the characters of the string value up to the first space, and the expression associated with *Model* is `extract "\S.*" context "\s.+"` which extracts all the remaing characters in the string after the first space.[12] This operation adds the two columns in Figure 9 to the table in Figure 2.

For split attributes we need to merge values. We can gather values together and merge them with a γ operator which we introduce here. Syntactically, we write $\gamma_{B \leftarrow A_1+...+A_n} r$ where B is a new attribute of the relation r and each A_i is either an attribute of r or is a string. The result of the γ operator is r with an additional attribute B, where the B value on row k is a sequential concatenation of the row-k values for the attributes along with any given strings. As an example, consider $\gamma_{Model\ with\ Trim \leftarrow Model+\texttt{" "}+Trim} T_1$ which converts Table T_1 in Figure 10 to Table T_2.

We can use standard set operators to help sort out subsets, supersets, and overlaps of value sets.

as they encounter additional values recognized through mappings, but not recognized directly through regular expressions. Strengthening ontologies in this way is similar to the work on learning reported in [JMNR99, RJ99].

[11] In future research, we would like to obtain a general completeness result.

[12] In Perl, "`\s`" matches a white space character, "`\S`" matches a non-space character, "`.`" matches any character, "`+`" indicates one or more repetitions, and "`*`" indicates zero or more repetitions.

T_1

Make	Model	Trim
Ford	Contour	GL
Ford	Taurus	LX
Honda	Civic	EX

T_2

Make	Model	Trim	Model with Trim
Ford	Contour	GL	Contour GL
Ford	Taurus	LX	Taurus LX
Honda	Civic	EX	Civic EX

Figure 10: Application of the γ Operator to Table T_1 Yielding Table T_2

Target Attribute	Source Derivation Expression for Value Sets
Year	$\pi_{Year}T$
Make	$\pi_{Make}T$
Model	$\pi_{Model}T$
Price	$\pi_{Price}T$
Feature	$\rho_{Colour \leftarrow Feature}\pi_{Colour}T$ $\cup\ \rho_{Auto \leftarrow Feature}\pi_{Auto}\beta^{Auto}_{Yes,\ No}T$ $\cup\ \rho_{Air\ Cond. \leftarrow Feature}\pi_{Air\ Cond.}\beta^{Air\ Cond.}_{Yes,\ No}T$ $\cup\ \rho_{AM/FM \leftarrow Feature}\pi_{AM/FM}\beta^{AM/FM}_{Yes,\ No}T$ $\cup\ \rho_{CD \leftarrow Feature}\pi_{CD}\beta^{CD}_{Yes,\ No}T$

Figure 11: Inferred Mapping from Source Table T in Figure 4 to Target Table in Figure 1

We can, for example, take a union of the exterior colors in Figure 2 and features in Figure 3 to form part of the set for *Feature* in Figure 1. After adding needed projection and renaming operations, this union is $\rho_{Exterior \leftarrow Feature}\pi_{Exterior}T \cup \rho_{Features \leftarrow Feature}T$*/Make and Model/Features*, where T is the table in Figure 2 and *T/Make and Model/Features* is a path expression that follows the link under *Make and Model* in table T to the *Features* list in Figure 3. When we need subsets of a set, we can extend the standard selection operator $\sigma_C r$ to allow C to be a regular expression that identifies the subset of values we wish to include. Given that we can also apply set-difference operations, we can resolve overlapping sets by operator combinations.

Now that we have the operators we need, we can give examples. Figure 11 gives the mapping from the source table in Figure 4 to the target table in Figure 1. Observe that we have transformed all the Boolean values into attribute-name values and that we have gathered together all the features as *Feature* values. Figure 12 gives the mapping for the car ads from the site for Figures 2 and 3. For purposes of illustration, we assume for the mapping in Figure 12 that we have recognized the list and the horizontal table in Figure 3 as tables. Observe that we have split the makes and models as required, matched the synonyms *Miles* and *Mileage*, extracted the *PhoneNr* from the free text, and gathered together all the various features as *Feature* values.

Target Attribute	Source Derivation Expression for Value Sets
Year	$\pi_{Year} T$
Make	$\pi_{Make} \delta^{Make\ and\ Model}_{Make,\ Model} T$
Model	$\pi_{Model} \delta^{Make\ and\ Model}_{Make,\ Model} T$
Mileage	$\rho_{Miles \leftarrow Mileage} \pi_{Model} T$
Price	$\pi_{Price} T$
PhoneNr	$\epsilon_{PhoneNr} T/Make\ and\ Model$
Feature	$\rho_{Exterior \leftarrow Feature} \pi_{Exterior} T$ $\cup\ \rho_{Features \leftarrow Feature} T/Make\ and\ Model/Features$ $\cup\ \rho_{Body\ Type \leftarrow Feature} T/Make\ and\ Model/Body\ Type$ $\cup\ \rho_{Transmission \leftarrow Feature} T/Make\ and\ Model/Transmission$ $\cup\ \rho_{Engine \leftarrow Feature} T/Make\ and\ Model/Engine$

Figure 12: Inferred Mapping from the Source Tables T in Figure 2 and *T/Make and Model* in Figure 3 to the Target Table in Figure 1

4 Experimental Results and Discussion

We gathered tables of car advertisements from many more than a hundred different English-language sites (several dozen were from non-U.S. sites). Because of human resource limitations, however, we analyzed only 60. In gathering tables, we encountered very few our implemented system could not process. Our system does not (yet) (1) convert color-swatch icons to color names, (2) recognize check marks rendered from images, (3) read legends for abbreviated attribute names in column headers, and (4) handle embedded links to subpages describing more than one car. Every car-ads table we encountered had simple attributes in the top row; thus we discarded no tables for structural reasons.

Of the 60 car-ads tables we analyzed, 40 included links to other pages containing additional information about an advertised car (Figures 2 and 3 show a typical example). For all 60 tables, we first applied our system to identify and list attribute-value pairs for tuples of top-level tables, and then for the 40 tables with links, we appropriately associated linked information (without alteration) with each tuple. We then applied our extraction step and looked for mapping patterns.

Since our objective was to obtain mappings (rather than data), it was not necessary for us to process every tuple in every table. Hence, from every table, we processed only the first 10 car ads. As a threshold, we required six or more occurrences of a pattern to declare a mapping. A human expert judged the correctness of each mapping.[13] We considered a mapping declaration for a target attribute to be completely correct if the pattern recognized led to exactly the same mapping as the human expert declared, partially correct if the pattern led to a unioned (or

[13] Although expert judgement for tables can sometimes be hard [HKL+01], establishing correctness results for car-ads for our target table was not difficult.

intersected) component of the mapping, and incorrect otherwise. For data outside of tables, the system mapped an individual value to either the right place or the wrong place or did not map a value it should have mapped. Because of differences in granularity, we separate table mappings from individual-value mappings in reporting our results.

4.1 Results

We divided the 60 car-ads tables into two groups: 10 "training" tables and 50 "test" tables. We used the 10 "training" tables to adjust our car-ads extraction ontology to recognize ordered pairs derived from our table-understanding procedure and also to fine-tune our ontology and update it with the latest makes, models, and features.[14] Adjusting for recognizing ordered pairs was straightforward—we simply added the various attribute names we found in the training set as keywords and sometimes as context identifiers for values (particularly for *Mileage* and *Price*, which both need something other than standard numbers to correctly identify them). Updating our ontology with the latest makes, models, and features was also straightforward, although it was a bit tedious especially for features, which tend to be more prolific in dealer sites on the Web than in classified ads posted by individuals.

For the 10 training tables, we were able to identify 100% of the 57 mappings while declaring no false mappings. Furthermore, we correctly found 94.6% of the values in the linked data, while incorrectly declaring only 5.4%. These numbers decreased for the 50 test tables. For these 50 test tables, we were able to completely identify 94.7% of the 300 mappings and partially identify 1.3%, while declaring no false mappings and failing to declare only 4% of the mappings. Based on a sample of nearly 3,000 values found in unstructured secondary pages, we found that the precision and recall ratios for the 50 test tables were approximately 86% and 97% respectively. This corresponds well with our previous car-ads experiments [ECJ+99].

Of the 357 mappings we discovered, 172 were direct, in the sense that the attributes in the source and target schemas were identical. Of the 185 indirect matches, 29 used synonyms and thus required only renaming with a ρ operator, 5 had Boolean values and thus required a β operator, 68 included features scattered under various attributes and in raw text and thus required $\cup$ and ϵ operators, 19 provided only factored telephone numbers and thus required ϵ and $\bowtie$ operators, 89 needed to be split and thus required a δ operator, and some required combinations of these operators (e.g. synonyms and union). The values we needed to split came in a variety of different combinations and under a variety of different names. We found, for example, *Description* as an attribute for the combination *Year+Make+Model+Feature*, *Model Color* as an attribute for *Make+Model+Color*, and *Model* as an attribute for *Year+Make+Model*.

[14] Our car-ads extraction ontologies had originally been constructed to recognize free-form car-ads, written as they usually appear in the classified ads of newspapers [ECJ+99].

4.2 Discussion

As mentioned in our earlier discussion, discovering correct mappings can lead to an increase in values extracted compared to values that would have been extracted by the extraction ontology alone and can also therefore lead to the acquisition of additional knowledge for the extraction ontology. In our experiments, we required a 60% or greater match to declare a mapping match. Overall, we actually achieved roughly 90–95%, a much higher percentage. This, however, leaves about 5–10% of the approximately 3,000 values encountered in tables as being unrecognized by the extraction ontology (and potentially many more since we processed only 10 car ads per site). Examples include non-U.S. models such as the Toyota Starlet or Nissan Presea; elaborately described features such as "telescoping steering wheel"; abbreviations not encountered previously such as "leath int" for "leather interior"; and features simply not encountered before, such as "trip computer".

We missed 12 mappings and only partially identified 4 mappings. Our system missed 6 mappings of car model because the extraction ontology was targeted to U.S. car-ads, and so non-U.S. car-ads introduced models that our system did not recognize. Two more mappings of car model were missed because the extraction ontology was not sufficiently robust (for Jaguar and SAAB models). The system missed 1 price mapping and 1 mileage mapping because the extraction ontology was overly restrictive. All of these problems can be corrected by minor adjustments to the extraction ontology. The final 2 missed mappings require more work. In both of these cases a cell contained two dollar-amount values, one for list price and another for sale price. Our system picked up the list price instead of the sale price.

The 4 partial mappings were for car features. Two of these cases could also be corrected by minor adjustments to the extraction ontology. A more interesting case was a table that included a "Description" column that contained an unstructured paragraph of text that would be more appropriately treated as if it were a linked page (where we expected to find unstructured text). Another interesting case was a table that included listings for trailers mixed in with car ads (e.g. "1999 Load Rite Trailer").

5 Conclusion

In this paper, we suggested a different approach to the problem of schema matching, one which may work better for the heterogeneous HTML tables encountered on the Web. In essence, we transformed the matching problem to an extraction problem over which we could infer the semantic correspondence between a source table and a target schema. We then showed how to discover the appropriate queries for source-to-target mapping rules. We gave experimental evidence to show that our approach can be successful. In particular, we correctly inferred 94% of the appropriate

mappings to our target car-ads ontology from 60 HTML car-ads Web tables with a precision of 98%.

As a next step in our work on extraction from HTML tables, we intend to implement the ideas we have on forming attribute-value pairs for tables in linked information, for nested tables, and for less common cases—where attributes for multiple records appear on the left, where attributes appear both on top and on the left, and where attributes are nested and compound. Once we have attribute-value pairs, we can directly apply the mapping techniques discussed here.

Many tables are behind forms, in the so-called "hidden Web" [RGM01], and we are currently working on extracting data from the hidden Web [LYE01]. Once extracted, if the result is a table, we can use the techniques presented here to extract the data into a target view. If the result is not a table, we use previous techniques we have developed [ECJ+99] to extract the data. Further, we also plan to piece together all the components we have developed in our data-extraction work [DEG] into a comprehensive extraction tool.

Acknowledgements: This material is based upon work supported by the National Science Foundation under grant No. IIS-0083127.

References

[Aut01] autoscanada.com, Summer 2001.

[BE02] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 2002. (to appear).

[Bob02] www.bobhowardhonda.com, January 2002.

[CGMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *IPSJ Conference*, pages 7–18, Tokyo, Japan, October 1994.

[CMM01] 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)*, Rome, Italy, September 2001.

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD 2001)*, pages 509–520, Santa Barbara, California, May 2001.

[DEG] Homepage for BYU data extraction research group. URL: http://osm7.cs.byu.edu/deg/index.html.

[ECJ+99] 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*, 31(3):227–251, November 1999.

[EX00] D.W. 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.

[Haa98] T.B. Haas. The development of a prototype knowledge-based table-processing system. Master's thesis, Brigham Young University, Provo, Utah, April 1998.

[HKL+01] J. Hu, R. Kashi, D. Lopresti, G. Nagy, and G. Wilfong. Why table ground-truthing is hard. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 129–133, Seattle, Washington, September 2001.

[JMNR99] R. Jones, A. McCallum, K. Nigam, and E. Riloff. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques, and Applications*, pages 52–63, Stockholm, Sweden, 1999.

[KS91] H.F. Korth and A. Silberschatz. *Database System Concepts*. McGraw-Hill, Inc., New York, New York, second edition, 1991.

[KWD97] N. Kushmerick, D.S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, pages 729–735, 1997.

[LN99a] S. Lim and Y. Ng. An automated approach for retrieving heirarchical data from HTML tables. In *Proceedings of the Eighth International Conference on Informaiton and Knowledge management (CIKM'99)*, pages 466–474, Kansas City, Missouri, November 1999.

[LN99b] D. Lopresti and G. Nagy. Automated table processing: An (opinionated) survey. In *Proceedings of the Third IAPR Workshop on Graphics Recognition*, pages 109–134, Jaipur, India, September 1999.

[LRO96] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Data Bases*, Mumbai (Bombay), India, 1996.

[LYE01] S.W. Liddle, S.H. Yau, and D.W. Embley. On the automatic extraction of data from the hidden web. In *Proceedings of the International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, pages 106–119, Yokohama, Japan, November 2001.

[MBR01] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.

[MHH00] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 2000.

[RGM01] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.

[RJ99] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth national Conference on Artificial Intelligence (AAAI-99)*, pages 474–479, Orlando, Florida, July 1999.

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

[Tub01] K. Tubbs. Recognizing records from the extracted cells of genealogical microfilm tables. Master's thesis, Brigham Young University, Provo, Utah, December 2001. http://www.deg.byu.edu.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In Foto N. Afrati and Phokion Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40, Delphi, Greece, January 1997. Springer-Verlag.